CHALMERS
UNIVERSITY OF TECHNOLOGY

Department Signals and Systems

Real-time camera based lane detection for autonomous driving

J.J.A. (Jasper) Spanjaards

Supervisors:
Prof. Jonas Sjöberg
Robert Hult

Eindhoven, September 2016

Abstract

In this report a new designed real-time lane detection algorithm based on monocular vision and inertial measurement unit (IMU) is explained and evaluated. The acquired image from the camera is undergoing an inverse perspective mapping to create a bird-view perspective of the road. Local maxima in this image determine the lane-markers. Faulty local maxima are removed and constrained line fitting over these local maxima is applied using Quadratic Programming. This fit represents the current drive-lane existing out of the left and right lane marker. The lane-width and vehicle distance to the center of the lane are calculated with the use of these fits. The vehicles IMU is used to predict the lane-marker in the next time-frame. The result is a lane detection algorithm which is working in real-time at a frequency of 10Hz. The algorithm is validated and is able to successfully detect the lane-width and vehicle distance to the center up to 94% of the time in highway scenarios. For almost every occurrence where the algorithm performed poorly, an error signal is send. In most of these situations the algorithm is able to minimize the error via logical statements and rely on previous measurements. It can be concluded that this algorithm can be used to maintain the vehicle position inside the lane for highway conditions.

Preface

Part of my master program at the Eindhoven University of Technology in Eindhoven, The Netherlands, is to do an internship. I've been granted the opportunity to do this internship at Chalmers University of Technology in Gothenburg, Sweden. The aim of the internship is to develop, implement and test a lane-detection algorithm for the Grand Cooperative Driving Challenge, held in May 2016 in the region of Eindhoven. Chalmers University of Technology is participating in this challenge, together with 9 other participants, of which one is the Eindhoven University of Technology.

I want to thank my Professor at the TU/e, Henk Nijmeijer, who introduced me to Professor Jonas Sjöberg, my main-supervisor at Chalmers. Further I want to thank Robert Hult, my daily supervisor, for his help and knowledge during the internship. To all my team members, you did a great job and worked hard to obtain a nice result. I'm grateful for being part of this amazing team and participating in this event. Finally I want to thank all my new friends I got to know during my stay here. I had a great time in Gothenburg during my internship, as during my spare time.

Contents

List of Figures

List of Tables

1. Introduction

In the automotive industry the focus nowadays lays more and more in providing semi and fully autonomous vehicles. In order to drive safely in autonomous mode, it is necessary that the car is able to interact with other vehicles which are, or are not autonomous. The system has to have knowledge of the surrounding, such as pedestrians as well as the road itself [1] [4] [23]. Having knowledge of the road is one of the fundamentals in autonomous driving. This can be achieved via various methods. One of these methods is vision, where one [1] [20] or multiple camera's [3] acquire an image of the road which is then processed to obtain the lane-markers from it. This is vital information in order to drive autonomous as it gives the position of the vehicle within the lane. During this project a new algorithm is created which detects the lane markers and the vehicle position using a camera.

In this report it is first described which methods are available to achieve the position of the vehicle within the lane [13]. Hereafter a choice is made between these modalities. This modality is then used to create a real-time lane detection algorithm, and is explained in this report. One of the methods is by using vision [2], which is based on the same perceptual cues that humans do. Humans look at the road and lane markings to drive correctly on their lane, or to determine their steering input when they decide to overtake. This modality is used in this algorithm, which feeds the lane-width and vehicle distance relative to the center of the lane to a lateral controller. Possible alternative modalities have been investigated, and are briefly explained in the first chapter.

In the second chapter the working of this real-time lane detection algorithm is explained. The basis is applying an inverse perspective mapping of the road [5], obtained by a camera. Combined with finding local maxima to obtain the lane markers, the position of the car as well as the road ahead is determined. The algorithm is designed for use during the Grand Cooperative Driving Challenge (GCDC, http://www.gcdc.net), held in Helmond, The Netherlands, in May 2016. It is an event where student teams from all over Europe work are challenged to drive autonomous in a cooperative way. As this algorithm is designed for the GCDC, the focus is that it works in highway scenarios, meaning straight to low curved roads, good conditioned roads with lane markings on both side of the lane and a low road-slope. In the third chapter the algorithm is evaluated.

The lane detection algorithm has to be fast enough to operate in real-time. It also has to be robust so that it works in various conditions, therefor it must satisfy the following conditions:

- Operate real-time at a sufficient frequency to make sure errors stay within bounds and do not result in any dangerous situations.
- The quality of lane detection should not be affected by shadows, which can be cast by trees, buildings, and so forth.
- It must be able to handle low curved roads (highway conditions).
- It must be able to handle with noise, coming from other objects such as cars and so forth.
- It should produce an explicit measurement of the reliability of the results obtained.

A few assumptions are made in order to simplify the problem, and to improve the algorithm:

- The road is assumed to be flat. This is vital for the inverse perspective mapping to be valid.
- Road markings are assumed to be parallel, as this is mostly the case in highway conditions. Exceptions such as merging and splitting of lanes do not occur.
- Lane-markers have a significant higher intensity than the road itself.

In chapter 5 a conclusion is given on the performance of the algorithm, followed by recommendations for future work.

2. Modalities for lane detection

There are various methods to successfully detect, or help to detect, roads and lane-markers. Aharon Bar Hillel et al [9] describe in their article the most relevant methods. Since their publish date no relevant new techniques were introduced. These techniques will be briefly explained and discussed below. In table 2.1 an overview is given of the modalities with their pros and cons.

2.1 Monocular vision

By obtaining the same image of the road as we humans see the road while driving, using a camera, it is possible to detect the road via various methods. Edge detection [15] works basically by identifying where the object changes color (its HSV or RGB code) significantly. Via this method lane-markers, and edged of the road are easily identified. Yoo et all [22] and Assidiq et all [1] use this method in their algorithm. Both of them include the use of Hough Transform [16], which is used to identify participle shapes, such as circles or lines. Another method is to use difference in intensity. Well prepared roads have lane markers with a higher intensity than the road itself. Edge detection works well in this scenario, as well as representing the lane-markers using local maxima points. However, the smaller the resolution of the image, the less edge detection performs. High resolution images however increase the computation time, which is a disadvantage when wanting to perform in real-time. Edge detection also needs more pro-processing to obtain the lane-markers, as more edges than only the lane-markers itself are often found. The algorithm has to distinguish the difference between lane-markers and an other object. Finding lane-markers based on local maxima is therefore found a more suitable method for real-time operation.

Once the previous and current lane are detected and the vehicle position within it is known, this information can be used to predict the state of the next frame. The next time-frame is then a control factor to determine if the calculations are correct or not.

Advantage

- Easy to implement and allows for good cost-effective solutions on a consumer-level.

Disadvantage

- Due to the high-reliability demands, building a vision-based system, even for the simplest applications, is a big development effort. Many functional blocks are required for a stable system, and many different conditions and assumptions have to be identified and handled. In addition a large validation effort is required, as many of the failure cases are rare and hard to predict.

2.2 Stereo vision

Stereo imaging is done by using two cameras in order to obtain 3D information, which can be compared to the human vision. The range accuracy is a function of the stereo baseline (the distance between the two cameras). A larger baseline stereo system will provide better range accuracy, but often at the cost of lower reliability and higher computation cost (a tougher correspondence problem to solve). Generally speaking, stereo imaging poses a greater processing challenge compared with LIDAR system, with an increased probability of errors. Still, stereo imaging can be used for the same basic tasks as LIDAR, including obstacle identification, host to road pitch angle estimation, curb detection and 3D road geometry and slope estimation. Bertozzi and Broggi developed an algorithm (GOLD) [3] that make use of stereo vision

Advantages

- Easy to implement and allows for good cost-effective solutions on a consumer-level.
- Able to compute distances of objects relative to the car.

Disadvantages

- Range accuracy and reliability lower than LIDAR because depth measurements is texture dependent.
- Even more than for monocular vision, there is a large development effort in order to create a high-reliable system.

2.3 Light detection and ranging (LIDAR)

LIDAR is a surveying technology that measures distance by illuminating a target with a laser light. Sren and Benjamin [12] created a lane marker detection and mapping algorithm purely based on LIDAR alone. Albert et all [11] combine LIDAR and monocular vision in order to find lane markers. Basically, LIDAR is used to:

- Identify objects and boundaries of the road by using its 3D detection.
- Identify the type of road by determining the road roughness. This can also be used to identify the edge of the road.
- Identify potholes and bumps in the road surface

Advantage

- Most LIDARs can report reflected intensity as well, providing a substitute to a visual camera with an advantage of being an active light source and thus independent of natural light sources. This specifically helps in coping with shadows and darkness. Since lane marks have only intensity information and no 3D structure, intensity measurement is required if the LIDAR is to be used as the only modality.

Disadvantages

- LIDAR equipment is still relatively very expensive.
- Large development effort

2.4 Geographic information systems (GIS), Global positioning system (GPS) and inertial measurement unit (IMU)

GIS is basically a geographic mapping of the reality, containing information about roads, fixed obstacles (like houses e.g.), altitudes, lakes and more. Possible sources for map information are high-resolution aerial (or satellite) images or GPS measurements collected on the ground, converted to digital maps and Digital Elevation Models. The resolution obtainable from aerial imagery can go up to 2.5cm [6], which is enough to detect lane marks.
GPS is a satellite-based navigation system made up of a network of satellites placed into the earth's orbit. Satellites circle the earth twice a day in a very precise orbit and transmit signal information to earth. GPS receivers take this information and use trilateration to calculate the user's exact location. Essentially, the GPS receiver compares the time a signal is transmitted by a satellite with the time it is received. The time difference tells the GPS receiver how far away the satellite is. Now, with distance measurements from a few more satellites, the receiver can determine the user's

position and display it on the unit's electronic map. A GPS receiver must be locked on to the signal of at least 3 satellites to calculate a 2-D position (latitude and longitude) and track movement. With four or more satellites in view, the receiver can determine the user's 3-D position (latitude, longitude and altitude). Once the user's position has been determined, the GPS unit can calculate other information, such as speed, bearing and more. The position accuracy for commercial GPS systems is between 3 and 5 meters [7] . However, using IMU (which determines the roll, pitch and heading of the wearer) may improve the accuracy of the GPS system up to 1m. To increase the accuracy even further, corrections can be applied using real time kinematic (RTK). It uses measurements of the phase of the signal's carrier wave, rather than the information content of the signal, and relies on a single reference station or interpolated virtual station to provide real-time corrections, providing up to centimeter-level accuracy [17] at a frequency of 10Hz. This is enough to use GPS as a reliable system to determine the vehicles position within the (prerecorded) lane. Sebas Peters uses GPS with RTK correction, IMU and GIS in his algorithm [18].

Advantage

- High accuracy and reliability with RTK when connection with satellites are good.

Disadvantages

- A base station is needed to provide the RTK signal.
- RTK is still relatively expensive, costing several thousands of euros.
- Signals can be obstructed by objects such as threes and buildings.
- Not able to work without RTK.
- When GPS signal is lost the system has to rely on the cars IMU, which is only valid for a small period of time (few seconds).

2.5 Radar

Radar is an object-detection system that uses radio waves to determine the range, angle, or velocity of objects. A radar transmits radio waves or microwaves that reflect from any object in their path. A receive radar, which is typically the same system as the transmit radar, receives and processes these reflected waves to determine properties of the object(s). It is able to detect obstacles like other vehicles or buildings, and able to discriminate between road an off-road regions based on their large reflective difference. The advantage of a radar is the it is relatively cheap, and can used well in combination of more than one modality. The disadvantage is that the properties formed are only a limited subset of the capabilities a LIDAR has.

2.6 Overview of modalities

LIDAR is most likely giving the best results when it comes to road detection and also other object detection, like cars for example. However, the current price of well performing LIDAR makes it too expensive for the use in a car. However, relative cheaper LIDARs may be used in addition to other modalities. Monocular vision is a cost-effective method, which is not fully proven yet, it has been proven to work fairly decent in highway scenario's and there are a lot of working prototypes.

Table 2.1: Overview of modalities

Name	Main features	Advantages	Disadvantages	Price [EUR]
Monocular vision (camera)	Recording visual image of area in front of car	Cost-effective, easy to implement and similar to human driving	Development effort, failure cases are rare & hard to predict, solving lightning variation and illumination at night	100 (webcam based)
Stereo vision (two camera's)	Same as monocular vision	Same as monocular vision including the ability to see depth	Interaction between cameras needed, increased computation time	200 (webcam based)
LIDAR	3D-mapping around vehicle	Emits light $\rightarrow$ independent of ambient light and creates a map around whole vehicle	Expensive, but prices tend to drop in upcoming years, development effort	25.000 (for 360° FOV)
GIS + GPS + IMU	GIS: Map of area; GPS: determines geographic position; IMU: determine roll, pitch and heading	Knowing driving area beforehand on large range, high accuracy	Only accurate with (costly) RTK corrections, Base station needed for RTK, GPS connection can fail	10.000 (including RTK)
Radar	Detecting objects (range, angle and velocity)	Can detect obstacles and their (relative) speeds	Low accuracy, only assistance in autonomous driving	500 (based on long range radar 77Ghz)

2.7 Choice of modality

Various modalities are investigated and described in this chapter. Monocular vision is chosen as the modality in this algorithm to detect the road/lane. It will be combined with the GIS+GPS+IMU system, however this will be outside the scope of this report. The GIS system will be used to determine additional lanes and to average the calculated road width and distance to center of the vehicle, based on the reliability of the signal.

3. Methodology

There are lots of studies done on lane detection using vision. Overviews of articles [2] [13] give a good insight which methods are used. Inverse perspective method is used in about 50% of the programs. It simplifies the computations afterwards as it creates a linear scale and parallel lines, making calculations easier. Most algorithms involve techniques as 'edge detection' (searching for a marking with a positive peak in the intensity, followed by a negative peak in the intensity) [19] [20] or 'Hough transformation' [21] (detect patterns such as straight lines or circles). A combination of these two is also often used. A Kalman filter is regularly used [21] [8] to predict the next frame, improving the results and be more resistant to noise.

The designed lane detection algorithm in this report is based on different basic principles in image processing, combined with the sensors data of the car. The overview of the processes in the algorithm is shown in Figure 3.1 below. A more detailed explanation of every process is given further in this chapter. The algorithm is designed to work in real-time and thus computation time is an important factor in designing it. Matlab and Simulink is used to build and run the algorithm.

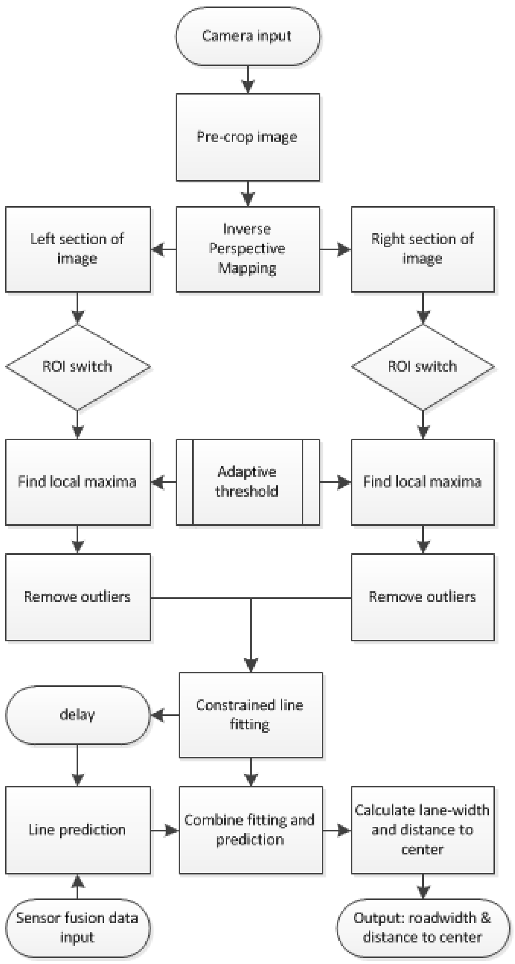


Figure 3.1: Overview of the processes in the lane detection algorithm

3.1 Camera input

In previous articles [1] [20] [3] [22] concerning lane detection using monocular vision, only a few state which camera they used, and non explained why. Webcams were often used, as they are easy to handle, relatively cheap and easily implementable. The most important camera features are resolution, frames per second (FPS), Field of view (FOV) and focus-type.

To determine which resolution is needed, we must consider how far ahead we want to detect the lines, but also take into account that a higher resolution means a larger computation time. As the algorithm is only used to determine the lane-width and distance to center of the car, it is reasonable to detect lines up to 10m in front of the car. A camera with a resolution of 360 x 640 has been proved via testing to accurately detect the lane-markers. A lower resolution is used in previous articles but gives a slightly less accurate measured distance. Camera's basically work on 30Hz/FPS, it is preferred to run on the highest possible frequency to obtain a smooth and up-to-date measurement. This is necessary in order for the lateral controller, which determines the steering input in the car, to operate fast. However, the computation time is too long to operate at this frequency. Therefore the frequency of obtaining an image from the camera is lowered to 10Hz. At 70km/h this gives the system 0.1s to respond, meaning a distance of about 2m is traveled. For low-curved roads this means that the real position does not deviate significant with the calculated position. The FOV has to be large enough to detect a single lane, which is the case for almost all webcams.

Automatic light correction is necessary to cope for when driving through a tunnel, or change in weather conditions. It should be noted that it does not always work, and in order to work properly, the focus has to be approximately 5 meters in front of the car. This is also necessary to obtain a clear and sharp image for the lane markers. Therefore a camera with a fixed (with the correct focus length) or manual focus is needed.

A Logitech C920 webcam [14] is used, it satisfies the most important features and is easy to implement (webcam is supported in Matlab). An alternative could be a car DVR camcorder such as the MINI 0805. These are specially designed to record the road ahead and have very good specifications for doing so. However, this cam is probably (not yet) supported by Matlab. It is preferred to mount the camera on top of the car to have a clear vision. However, due to absence of proper mounting material it is chosen to mount the camera using a suction cup on the front windshield at the inside of the car. The camera's focus is about 5 meters in front of the car to obtain a clear and sharp image of the most important lane markers.

The camera is able to detect multiple lanes, this is beneficial to determine if there are multiple lanes. Also it is able to determine a more accurate distance to the lane center of the lane on either side. However, larger images have to be acquired, not only in width, but also in height as the camera has to look further ahead due to the FOV. This results into a longer computation time, which is a disadvantage if the algorithm has to run in real time. Another disadvantage is that traffic passes the view of the camera much more often than it does when we only look at the lane we drive in for a distance of 10 meters. This means that the chance of obtaining false outcomes due to noise is much larger. Due to the disadvantages it is chosen to look forward to only the lane which the car drives on. Assumed is that the lanes on the side are of equal width in order to predict a reasonable distance of the vehicle to the center of a parallel lane. Lanes on the side are detected using road mapping.

3.2 Inverse perspective mapping (IPM)

The camera in the car captures the road under an angle of about 30 $^\circ$ relative to the road. Due this perspective, the lane markers are not parallel on the input picture, but moving inwards as can be seen in Figure 3.4a. Using an inverse perspective mapping, a 'bird-view' image is obtained as shown in Figure 3.2. This results in parallel lane-markers with a linear distance scale, which increases the easiness of calculating the lane-width. The transformation matrix is created using the rotational matrices below, shown in (3.1). Figure 3.3 explains the virtual rotations of the camera image.

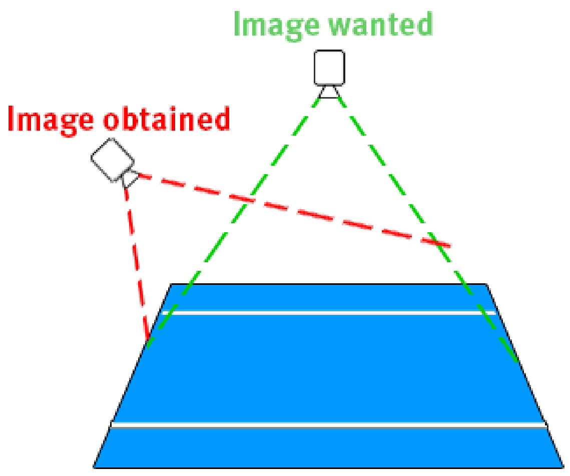


Figure 3.2: Principle of IPM

$$R_\omega = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\omega) & sin(\omega) & 0 \\ 0 & sin(\omega) & cos(\omega) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; R_\phi = \begin{bmatrix} cos(\phi) & 0 & -sin(\phi) & 0 \\ 0 & 1 & 0 & 0 \\ sin(\phi) & 0 & cos(\phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; R_\kappa = \begin{bmatrix} cos(\kappa) & sin(\kappa) & 0 & 0 \\ -sin(\kappa) & cos(\kappa) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

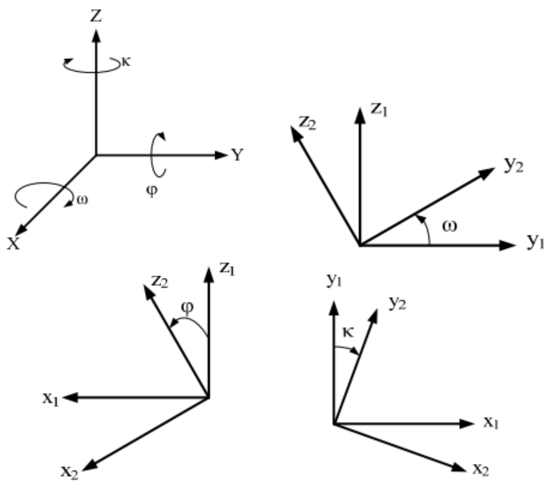
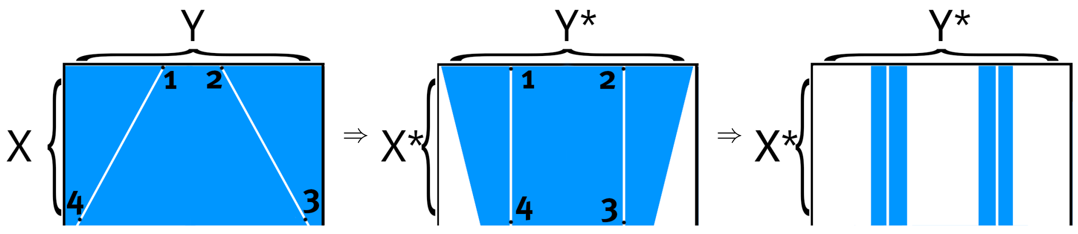


Figure 3.3: Rotating of camera body in an alternative frame using inverse perspective mapping

Clearly for practical reasons the Matlab function *fitgeotrans* is used to determine the projective transformation matrix [5] [10]. Here 4 points within the input image (pre-cropped) have to be selected and 4 points which indicate their true position. The true positions should form a rectangle. The transformation matrix given is feed into the Simulink block 'Warp', which gives the transformed image. A representation of the result is shown in Figure 3.4. The redundant part of the image is removed to increase speed. To reduce noise and increase the speed of the algorithm a region of interest (ROI) is selected after finding reliable line-fits. This is shown in Figure 3.4c. When the camera position is changed, a new transformation matrix has to be determined.



(a) Cropped input image (b) Applied IPM (c) Region of interest

Figure 3.4: Schematic representation of how inverse perspective mapping works. The white lines represent the lane-markers and the blue represents the tarmac.

3.3 Image splitting

The image is split into a left half and a right half. This is done to increase the easiness of coding for the next processes, as they do not have to take into account that the image consists of two different lines. Another benefit is that the adaptive threshold value for selecting the local maxima is more accurate. Often one side of the image is covered (partly) in shades, while the other half is not, this is now of less influence in the outcome.

3.4 Region of interest (ROI) switch

A ROI is defined as a small area around the previous found linefit, as shown in Figure 3.4c. When a detection of a lane markers seems unreliable, it is possible that the lane markers will be outside the ROI and can not be seen. To prevent this, a safety process is build in so that the algorithm will look at the complete picture again in the next frame. The switch is based on the following logical statements:

- If the number of local maxima found after removing the outliers (explained hereafter) is less than 20% of the maximum number of local maxima to be found, it assumes that no line is found. In this case the previous found local maxima are taken. This is mostly the case when no lane marker can be found or when sudden large change in lightning is occurring (e.g. driving in or out a tunnel).
- If $|y_{0,current} - y_{0,old}| > \frac{Image\ Width}{30}\ AND$ $\left(y_{0,current} < 0.05\frac{Image\ Width}{2}\ OR\ y_{0,current} > 0.95\frac{Image\ Width}{2}\right)$. Meaning that if the current line ($x = \alpha y + y_0$) moved more than the threshold value ($\frac{Image\ Width}{30}$) and is situated on the edges of the image, the ROI is not valid. This is mostly the case during lane-switching.
- If $\Delta Lanewidth > 0.1\ AND\ 3.0 < Lanewidth < 4.0$. Meaning that the lane-width change in a single time frame is larger than a certain threshold value (0.1 meter). In this case it takes mean of the current lane-width and the previous lane-width. This is mostly the case when road markings or other vehicles are falsely detected as a line.

- If $3.0 > Lanewidth\ OR\ Lanewidth > 4.0$. Meaning that if the lane-width is outside of its estimated bounds (3 - 4 meters), the ROI is not valid. In this case the previous lane-width is taken. This is mostly the case when road markings or other vehicles are falsely detected as a line.

If the ROI is inactive, also a message is send to the central controller that the measurement is unreliable.

3.5 Finding local maxima

The key function on which the algorithm is based on is finding the local maxima of the image. The input image is a gray-scale image of type uint16 (unsigned 16-bit integers), which mean that every pixel has a value ranging between 0 and 2^{16}-1 (65535). 0 Indicates a black pixel, and 65535 indicates a white pixel. A local maximum is a point in a region, where the value is the highest, and thus the brightest. It is assumed that a lane marker is the brightest object on the road, every found local maximum should thus lay on an lane marker. Important settings are the maximum number of local maxima (MAX_LM), neighborhood size (the area around the pixel which all have a lower value) and the threshold (minimum value of the pixel to be seen as a local maximum). The maximum number of local maxima is chosen to be the height in pixels of the image (the cropped IPM image) divided by 10. The width of the neighborhood size is chosen to be equal to the width of a section and the height of the neighborhood size is chosen to be the section height divided by the maximum number of local maxima. The threshold is an adaptive threshold, which is based on the mean intensity of the section and the maximum intensity. The threshold value is set as $intensity(max) - (intensity(max) - intensity(mean))/2$.

3.6 Histogram And Fit Based Outlier Removal (HAFBOR)

It can be that local maxima are found that are not on a lane marker. Random Sample Consensus (RANSAC) is initially used to separate the inliers from the outliers. However, RANSAC does not always chose the correct inliers, therefore a new outlier removal script is written. To remove the potential outliers, first a horizontal histogram is made of the found local maxima. A section is divided into bins (in this case 20). If more than the maximum number of local maxima divided by the number of bins are in a single bin, it is checked for how many bins the number of local maxima is higher than this value. If the mean number of local maxima found over these bins is higher than a certain threshold value, given by (3.2), it decides that the line is valid.

$$Count\ density > 0.5\frac{Image\ Width}{MAX\_LM} * number\ of\ valid\ bins \tag{3.2}$$

Starting from the local maximum found closest to the car, the next local maximum is searched for within a forward region (the initial search frame). This is done for every new found local maximum. When the 7th local maximum is found, a fit is made through the previous found local maxima, the new search region is based on this fit to increase the accuracy of finding the correct next local maximum. Figure 3.5 shows a schematic representation of how these steps are taken.

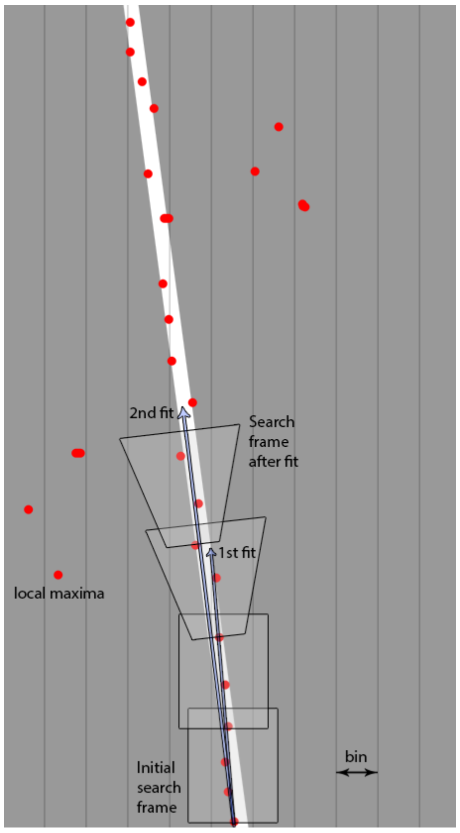


Figure 3.5: Schematic representation of how local maxima outliers are removed

3.7 Constrained line fitting

For both lines the filtered local maxima are obtained, a fit through these points should represent the lane-markers. Both linear fitting as well as 2nd order fitting has been explored. In real time there is no solver that is able to solve the simple 2nd order constrained line ($x = \alpha_1 y^2 + \alpha_2 y + y$) fitting. Offline the 2nd order fitted lines do not show a significant improvement on the result and is more prone to errors. Therefore it is chosen to work with a linear fitting. It gives a good representation of a straight to low curved road. It is assumed that the lane-markers are parallel to each other within a boundary of ϵ. This quadratic programming problem can be formulated as in (3.3). The objective of quadratic programming is to find a vector Z of the same dimension as f that

$$\begin{aligned} minimizes & \left(\frac{1}{2} Z^T H Z + f^T Z\right) \\ s.t. \quad & AZ \leq b \end{aligned} \tag{3.3}$$

The matrices H, f, A and b, depend on the minimization problem that has to be solved. The fit for a single line is obtained by solving

$$min \sum_{i=1}^{N} (ky_i + m - x_i)^2 \tag{3.4}$$

where (x_i, y_i) are the coordinates of the local maxima. N is the amount of local maxima. (3.4) can be rewritten as

$$min\sum_{i=1}^{N}\left(\begin{bmatrix}k & m\end{bmatrix}\begin{bmatrix}y_i^2 & y_i\\ y_i & 1\end{bmatrix}\begin{bmatrix}k\\ m\end{bmatrix}+x_i^2-2ky_ix_i-2mx_i\right)=min\sum_{i=1}^{N}\left(Z^TH_iZ-2f_i^TZ+x_i^2\right) \quad (3.5)$$

with

$$Z=\begin{bmatrix}k\\ m\end{bmatrix} \quad H_i=\begin{bmatrix}y_i^2 & y_i\\ y_i & 1\end{bmatrix} \quad f_i=\begin{bmatrix}y_ix_i\\ x_i\end{bmatrix}$$

the last term x_i^2 can be neglected as it does not influence Z. The equation obtained has only Z as variable. As only the matrix H_i and f_i change with i, thus the coordinates, we can write these as $\sum_{i=1}^{N} H_i = H$ and $\sum_{i=1}^{N} f_i = f$. That results in the following equation:

$$min(Z)\left(\frac{1}{2}Z^THZ+f^TZ\right) \quad (3.6)$$

As the fits of both lane-markers are dependent on each other, and they are assumed to be parallel with only a small variation, the combined fit has to be minimized, thus:

$$min(Z_1,Z_2)\left(\frac{1}{2}Z_1^TH_1Z_1+f_1^TZ_1+\frac{1}{2}Z_2^TH_2Z_2+f_2^TZ_2\right) \quad (3.7)$$

where the subscript denotes a local maximum on the left lane marker with a 1, and one on the right with a 2. This can again be written as a matrix form:

$$min(Z_1,Z_2)\left(\begin{bmatrix}Z_1^T & Z_2^T\end{bmatrix}\begin{bmatrix}H_1 & 0\\ 0 & H_2\end{bmatrix}\begin{bmatrix}Z_1\\ Z_2\end{bmatrix}+\begin{bmatrix}f_1\\ f_2\end{bmatrix}^T\begin{bmatrix}Z_1\\ Z_2\end{bmatrix}\right)=min(\bar{Z})\left(\frac{1}{2}\bar{Z}^T\bar{H}\bar{Z}+\bar{f}^T\bar{Z}\right) \quad (3.8)$$

Now the quadratic programming problem as in (3.3) is obtained, with $\bar{Z}=\begin{bmatrix}Z_1\\ Z_2\end{bmatrix}$, $\bar{H}=\begin{bmatrix}H_1 & 0\\ 0 & H_2\end{bmatrix}$ and $\bar{f}=\begin{bmatrix}f_1\\ f_2\end{bmatrix}$. As the slope of both fitted lines are assumed to be bounded parallel, this problem is subjected to $|k_1-k_2|\leq\epsilon$. Thus both $k_1-k_2\leq\epsilon$ and $-k_1+k_2\leq\epsilon$ have to be satisfied, where ϵ is the constraint. As $\bar{Z}=\begin{bmatrix}k_1\\ m_1\\ k_2\\ m_2\end{bmatrix}$, the linear coefficients in the constraints become:

$$A=\begin{bmatrix}1 & 0 & -1 & 0\\ -1 & 0 & 1 & 0\end{bmatrix}$$

$$b=\begin{bmatrix}\epsilon\\ \epsilon\end{bmatrix}$$

A solver is used to solve this problem. The output is the matrix $\bar{Z}$ which give the coefficients for the two bounded parallel lines which represent the lane-markers.

3.8 Line Prediction

The car inertial measurement unit (IMU) and GPS can be used to predict the state of the car for the next frame. Using the line-fit of the current frame and data from the IMU and GPS, an estimation can be made of the line-fit for the next frame. Corrections are applied to the GPS using the IMU and RTK system. An accurate value for the distance traveled (ds) can be obtained between two frames. The yaw rate is obtained from the IMU. The prediction is calculated using

$$\begin{aligned} \alpha_{prediction} &= \alpha_{measurement} - yaw\ rate \\ y_{0,prediction} &= y_{0,measurement} - ds * sin(yaw\ rate) \end{aligned} \tag{3.9}$$

with $ds = \sqrt{(dx^2 + dy^2}$. This gives an estimated linefit for the next frame: $x_{prediction} = \alpha_{prediction} y + y_{0,prediction}$. This estimation is used together with the next found frame in order to give a more reliable result.

3.9 Lane-width and distance to center calculation

First the fitted line and the predicted line are combined by taking the mean of the parameters α and y_0. For the line on the right, a value of $\frac{Image\ Width}{2}$ is added to y_0 in order to rectify for the shift in coordinates for using splitting the image. The lane-width is then calculated using.

$$mean\left[(k_1 y + m_1) - (k_2 y + m_2)\right] * P2M \tag{3.10}$$

Meaning, the mean distance between the right and left lane-marker determines the lane-width. $P2M$ is the pixel to meters conversion, this is calibrated initially to determine how many pixels equal 1 meter.

To determine the vehicle distance to the center of the lane, the found lane-width is used. In situations where the lane-width is potentially faulty, the lane-width is used as depicted in section 'Region of interest (ROI) switch'. The distance to the center is determined by.

$$\begin{aligned} Distance\ to\ left\ lane &= \left[\frac{Image\ Width}{2} - \left[(k_1 y + m_1) - (k_2 y + m_2)\right]\right] * P2M - \frac{Lanewidth}{2} \\ Distance\ to\ right\ lane &= \left[\frac{Image\ Width}{2} - \left[(k_1 y + m_1) - (k_2 y + m_2)\right]\right] * P2M + \frac{Lanewidth}{2} \\ Distance\ to\ center &= \frac{Distance\ to\ left\ lane + Distance\ to\ right\ lane}{2} - Camera\ offset \end{aligned} \tag{3.11}$$

3.10 Result

An algorithm is successfully build within a Matlab and Simulink environment. The processes shown in Figure 3.1 and described in this chapter are all implemented. The algorithm is able to run at a frequency of 10Hz.

4. Implementation and validation

4.1 Hardware

For real-time testing within a vehicle, a Volvo S60 is used. The algorithm is running on an industrial PC with the following configurations:

PC hardware

- Processor: Intel I7 (clockspeed: 1.7Ghz)
- Memory (RAM): 4GB DDR-III
- Graphics: Intel HD graphics 4000
- Hard disk: Intel 535 120Gb SSD

PC software

- Operation system: Windows 7 professional 64bit
- Matlab version: 2015b 64 bit

The PC communicates via UDP with a Micro-Auto-Box running the dSPACE software containing the lateral and longitudinal controllers. The vision is obtained using a Logitech c920 webcam, plugged into the PC using USB.

4.2 Performance

The algorithm is validated and has been tested on several roads in various weather conditions (sunny/clouded/rain/evening). It works in real time at 10Hz using the hardware and software described above. The validation method is discussed below, followed by successful and failure scenarios.

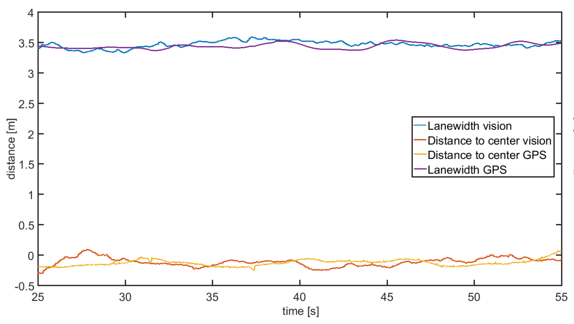
4.2.1 Validation

The algorithm is validated by measuring and estimating the lane-width and the vehicle distance to the center. A test road is mapped using GPS with RTK corrections. The test vehicle records the lane and calculates the lane-width (real width is 3.5) and distance to the center both with GPS as with vision. This gives a good representation for the functioning of the vision algorithm. As the lane is mapped by driving over the lane-marker, there is a small deviation due to not driving perfectly on the lane-marker. The comparison is shown in Figure 4.1.

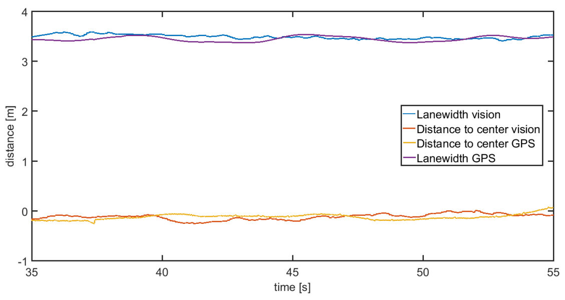
Figure 4.1 shows that the lane-width and vehicle distance to center are fairly equal. The maximum difference in measurements is 30cm, which could be due to the deviation of mapping the lane markers. As the data of the GPS and vision algorithms is combined, it has a maximum error of about 15cm. This is acceptable for a vehicle of 2m wide on a lane of 3.3m wide.

4.2.2 Successful scenarios

Various test drives have been made on the E6 and E20 highway near Gothenburg, Sweden. The velocity of the vehicle during testing was varying between 80 - 100km/h. The results of three of these test drives can be seen in Figure 4.2 and Table 4.1. Figure 4.3 shows a frame with successful determined lane-markings. Overall the algorithm works within a good margin of error in highway scenarios.



(a) Comparison 1

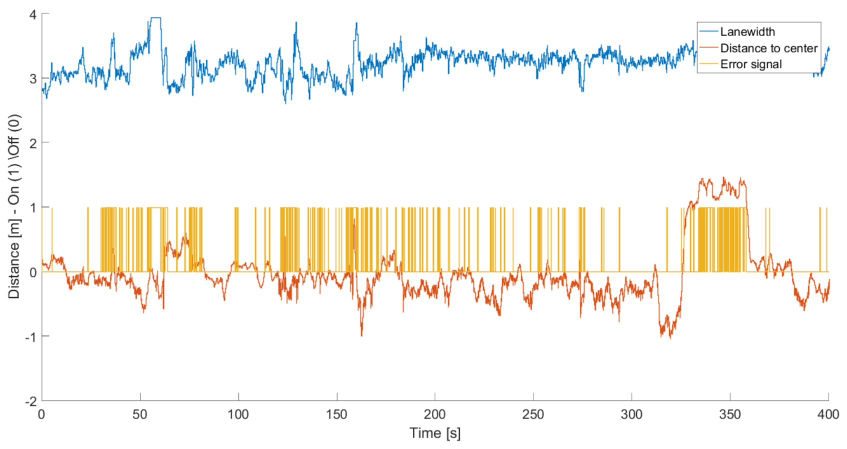


(b) Comparison 2

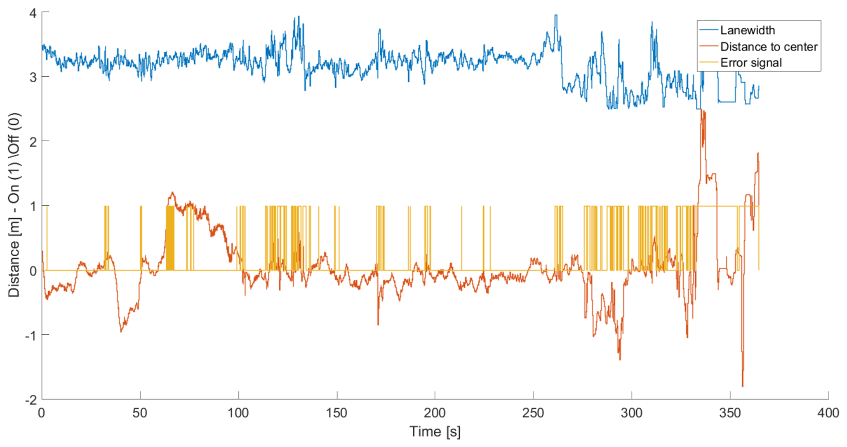
Figure 4.1: Comparison of calculated lane-width and vehicle distance to center between vision and GPS algorithm. The distance traveled is circa 400m while the vehicle stays in the same lane.

Table 4.1: Result of test drives on highway. The results are corresponding to the results of Figure 4.2.

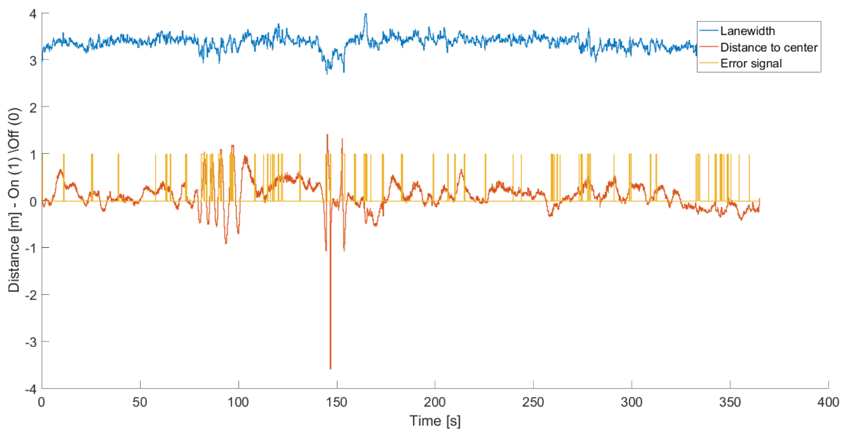
Testdrive #	# Frames	Average lane-width [m]	Standard deviation [m]	Real lane-width [m]	# Error-signals
1	3643	3.17	0.27	3.0 - 3.3	801
2	4001	3.24	0.20	3.0 - 3.3	595
3	3647	3.35	0.15	3.3 - 3.5	213



(b) Record 2

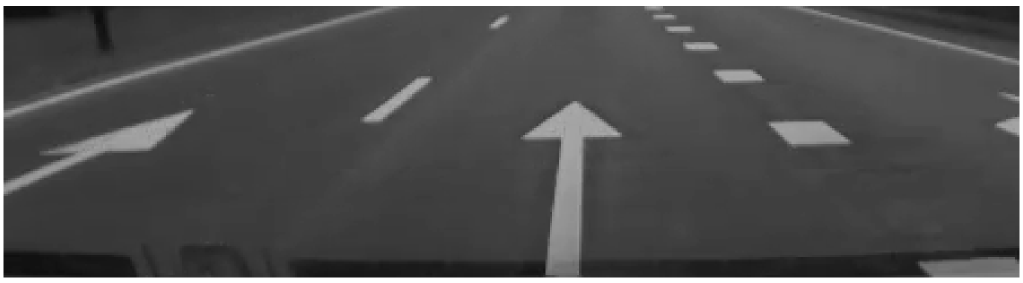


(a) Record 1

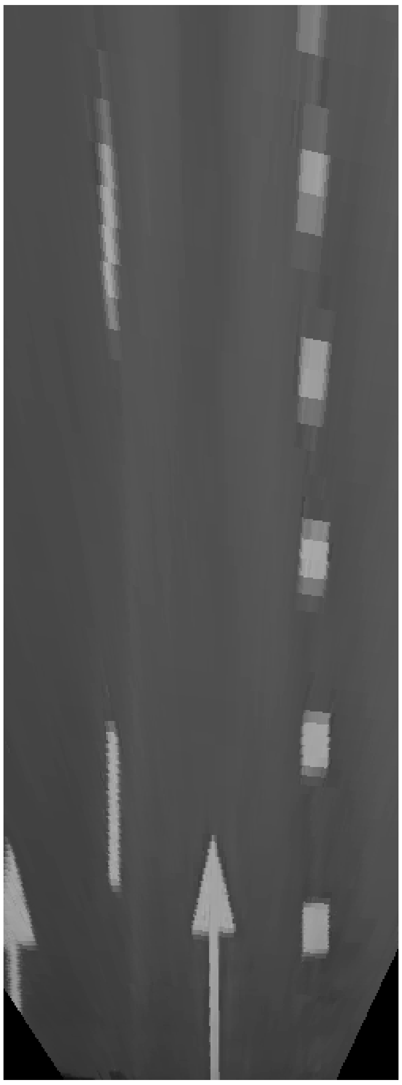


(c) Record 3

Figure 4.2: Results of lane-width, vehicle distance to center and error signal during real-time testing on highway. Vehicle was not in autonomous mode during this recordings.



(a) Cropped input image


(b) Applied IPM

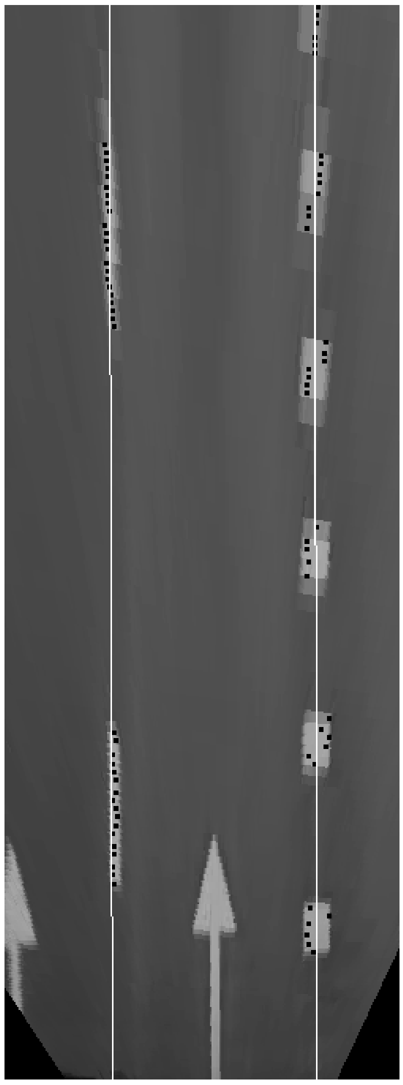

(c) Result of the linefits which represent the lane-markers

Figure 4.3: Snapshot of the result during a real-time test of the algorithm.

4.2.3 Failure scenarios

There are several cases where the algorithm fails, these failures can be either small or large. In most cases they are detected and the sent signal to the micro-autobox is partly corrected with the use of logical statements as described in the section 'Region of interest (ROI) switch'. Also by combining the signal with the signal of the algorithm based on GPS, the error is reduced even more (or may even rely completely on the GPS algorithm signal). The lateral controller should not be too sensitive in order to coop with the noisy (combined) data received from both algorithms.

Figure A.1 and Figure A.2 correspond to certain frames of the graphs in Figure 4.2. During the test drives, a couple of scenarios occurred where the algorithm failed to correctly determine the lane-width and vehicle distance to the center of the lane. Table 4.2 shows what is the cause for the failure scenario and what the impact is on the output side of the algorithm.

Table 4.2: Causes of failure scenarios with the corresponding impact on the output of the algorithm.

Frame	**Cause**	**Lane-width error [m]**	**Distance to center error [m]**	**Error Signal**
a (117s of Figure 4.2a)	Lane at left side exists of brighter tarmac.	< 0.2	< 0.2	On
b (310s of Figure 4.2a)	Lane splitting occurred	$0.2 - 0.5$	$0.2 - 0.5$	On
c (323s of Figure 4.2a)	Other lane-markers on the road	$0.2 - 0.5$	$0.2 - 0.5$	On
d (333s of Figure 4.2a)	Overexposure of camera due to the sun	$0.5 - 1.0$	$0.5 - 1.0$	On
e (337s of Figure 4.2a)	Effect of wrong choice of ROI in frame d	$0.5 - 1.0$	> 1.0	On
f (34s of Figure 4.2b)	Car in front of vehicle blocking sight	< 0.2	< 0.2	On
g (58s of Figure 4.2b)	Lane splitting occurred	$0.5 - 1.0$	< 0.2	On
h (123s of Figure 4.2b)	White car in front with higher intensity than lane-markers	$0.5 - 1.0$	$0.5 - 1.0$	On
i (273s of Figure 4.2b)	Light gradient due to passing underneath bridge	$0.2 - 0.5$	$0.2 - 0.5$	On
j (357s of Figure 4.2b)	Error signal while driving over lane-marker	< 0.2	< 0.2	On
k (144s of Figure 4.2c)	Lane switching	< 0.2	$0.2 - 0.5$	On
l (162s of Figure 4.2c)	Other road markings	$0.2 - 0.5$	$0.2 - 0.5$	On

4.3 Result

The algorithm is successfully implemented in the test vehicle and it's results are validated. Failure scenarios make it hard to implement autonomous driving based on just a single modality, as they don't occur often and are sometimes hard to predict. Including all scenarios in a single real time algorithm based on one modality is an impossible task. The solution is combining different modalities in order to obtain a more reliable result. Most scenarios can then be prevented. For example driving too close behind another vehicle can often be prevented using a radar to keep distance, or to neglect part of the image during processing. An instant change in intensity due to overexposure can be detected, where after the algorithm can rely on the vehicle IMU for a short period of time, until the overexposure is compensated. Or the output signal could rely only on a signal provided by an algorithm based on LIDAR, which relies on its own emitted laser beam. With the use of GIS and GPS, it can be exactly known where and when e.g. lane splitting or merging occurs. The algorithm can adapt itself to that specific situation for that time so that its output signals are reliable.

5. Conclusion and recommendations

5.1 Conclusion

An algorithm is developed that is able to detect lane markings in real-time based on finding local maximums using monocular vision. It is able to operate in real-time at a frequency of $10Hz$ using a mid-level computer, which is enough to autonomously maintain the vehicle in the center of the lane. The outcome is most of the time not affected by low-contrast shadows, other vehicles on the road or small markings on the road itself. The algorithm is designed specifically for highway conditions and does not perform well in different conditions. Highway conditions mean lane markings on both side of the lane, with a good contrast with the tarmac, low-curved and flat roads. The algorithm is able to remove outliers caused by noise of e.g. shadows casted by trees or small raindrops. It uses a ROI to decrease computational time and increase the reliability of the output signal by decreasing the noise in the input image. It uses the vehicles IMU to predict the next frame to increase smoothness and reliability of the signal. The algorithm is validated by comparing its output signals with an output signals of an algorithm based on GPS with RTK correction. It is successfully able to calculate the lane-width and vehicle distance to the center of the lane. Over several test drives of approximately 6 minutes (circa 8km), the lane-width is calculated within its bounds ($\pm 0.25m$) with an average standard deviation of $0.21m$. Under the following circumstances the algorithm works poorly or fails:

- Absence of lane markers, leading to false detection and tracking
- Driving during wet road conditions towards the sun. This creates an intense reflection on most roads, leading to find local maximums on other places than the lane-markers.
- Driving over a high-contrast shaded road by objects such as trees. Spots where the sun-rays shine can be falsely seen as lane-markers.
- Driving on medium to highly curved roads or corners. The inverse perspective mapping fails. This is probably due to the rolling of the vehicle itself where after the given transformation matrix no longer is valid.
- Driving on bumpy roads will also cause the inverse perspective mapping to fail, as the camera sees the road under a varying angle.

5.2 Recommendations

For future work it is recommended to

- Combine the output with outputs based on different modalities.
- Vary the region of interest size according to the error instead of taking the complete image.
- Apply Kalman filter to predict next lane marker using the current and previous frames. This will smooth the found lane markers.

Bibliography

[1] Abdulhakam.AM.Assidiq, Othman O. Khalifa, Md. Rafiqul Islam, and Sheroz Khan. Real time lane detection for autonomous vehicles. *Computer and Communication Engineering, 2008. ICCCE 2008*, 2008. 1, 2, 7

[2] M. Bertozzi. *Vision-based lane/road detection methods*, volume 90. Proceedings of the IEEE, 2002. 1, 6

[3] Massimo Bertozzi and Alberto Broggi. Gold: A parallel real-time stereo vision system for generic obstacle and lane detection. *IEEE transactions on image processing*, 7, 1998. 1, 2, 7

[4] K. Chu, M. Lee, and M. Sunwoo. Local path planning for off-road autonomous driving with avoidance of static obstacles. *IEEE Transactions on Intelligent Transportation Systems*, 13, 2012. 1

[5] Prof. R. Collins. Planar homographies. In *Planar Homographies*, 2007. 1, 9

[6] Focal Flight. Image resolution. http://focalflight.com/capabilities/image-resolution. 3

[7] Garmin. About gps. http://www8.garmin.com/aboutGPS/. 4

[8] King Hann, Kah Phooi, and Siew Wen. Lane detection and kalman based linear parabolic lane tracking. *IEEE*, 2009. 6

[9] A.B. Hillel. *Recent progress in road and lane detection: a survey*, volume 25. Springer, 2012. 2

[10] Prof. W. Hoff. Computer vision. In *Computer Vision*, 2012. 9

[11] Albert S. Huang, David Moore, Matthew Antone, Edwin Olson, and Seth Teller. Finding multiple lanes in urban road networks with vision and lidar. *Springer Science and Business Media*, 2009. 3

[12] Sren Kammel and Benjamin Pitzer. Lidar-based lane marker detection and mapping. *IEEE Intelligent Vehicles Symposium*, 2008. 3

[13] A. M. Kumar and P. Simoni. *Review of Lane Detection and Tracking Algorithms in Advanced Driver Assistance System*, volume 7. IJCSIT, 2015. 1, 6

[14] Logitech. C920. http://www.arp.nl/webmedias/datasheet/575183859ce969320b1a2beb.pdf. 7

[15] MathWorks. Edge detection methods for finding object boundaries in images. http://se.mathworks.com/discovery/edge-detection.html. 2

[16] MathWorks. Hough transform. http://se.mathworks.com/help/images/hough-transform.html. 2

[17] NovAtel. An introduction to gnss, 2015. http://www.novatel.com/an-introduction-to-gnss. 4

[18] S.A.M. Peters. Road positioning and lane identification for autonomous vehicles. Eindhoven University of Technology, 2016. 4

[19] E. Unger. Kamerabasierte spurerkennung, June 2012. 6

[20] S. Vecek and R. Dillmann. Mono-camera based road marking and lane detection. https://his.anthropomatik.kit.edu/, 2007. 1, 6, 7

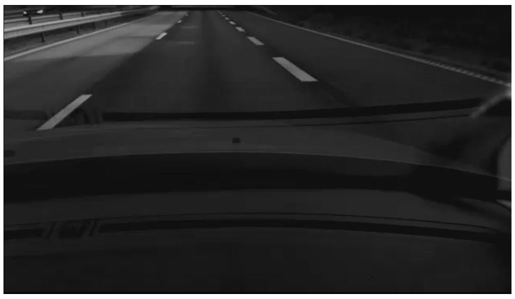
[21] V. Voisin, M. Avila, B. Emile, S. Begot, and J.Bardet. *Road markings detection and tracking using hough transform and kalman filter.* Springer, 2005. 6

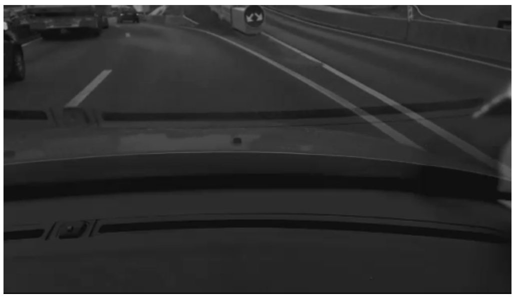
[22] Hunjae Yoo, Ukil Yang, and Kwanghoon Sohn. Gradient-enhancing conversion for illumination-robust lane detection. *IEEE Transactions on intelligent transportations systems*, 14, 2013. 2, 7

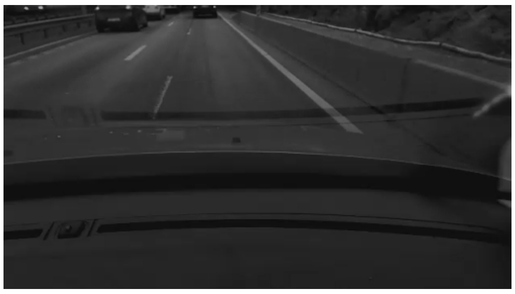
[23] J. Ziegler, P. Bender, and M. Schreiber. Making bertha drive an autonomous journey on a historic route. *IEEE intelligent transportation systems magazine*, 6, 2014. 1

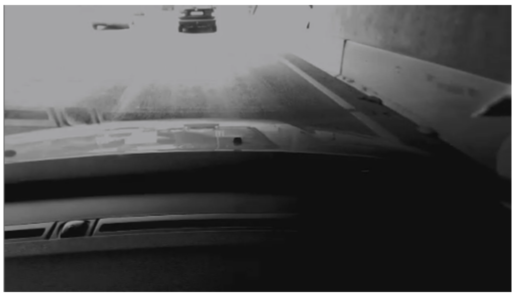
A. Appendix

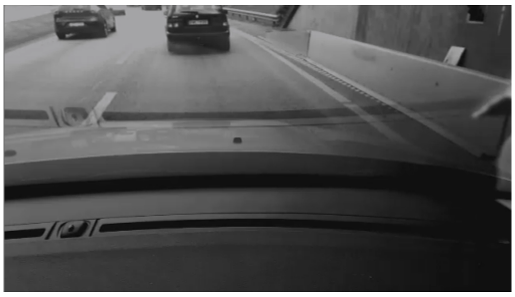
A.1 Driving scene during failure scenarios

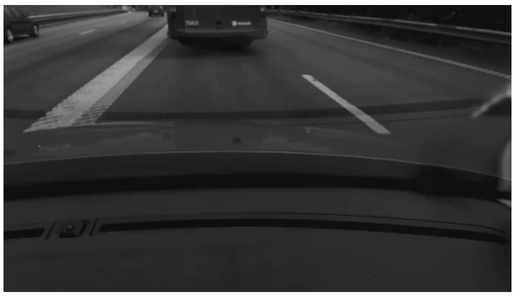

(a) Corresponding to 117s of Figure 4.2a

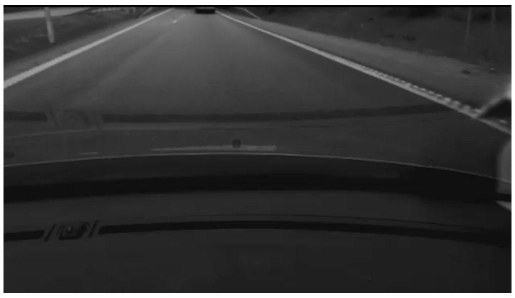

(b) Corresponding to 310s of Figure 4.2a

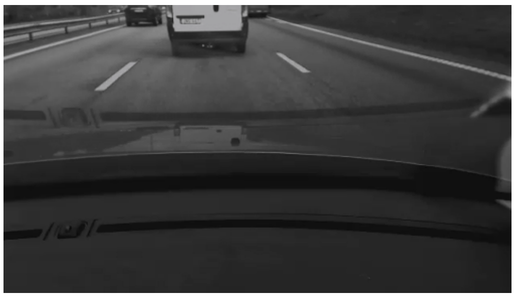

(c) Corresponding to 323s of Figure 4.2a


(d) Corresponding to 333s of Figure 4.2a


(e) Corresponding to 337s of Figure 4.2a

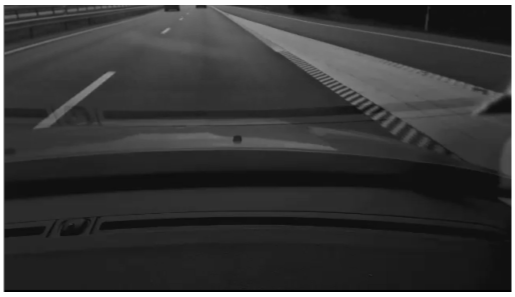

(f) Corresponding to 34s of Figure 4.2b


(g) Corresponding to 58s of Figure 4.2b


(h) Corresponding to 123s of Figure 4.2b

(i) Corresponding to 273s of Figure 4.2b

(j) Corresponding to 357s of Figure 4.2b


(k) Corresponding to 144s of Figure 4.2c

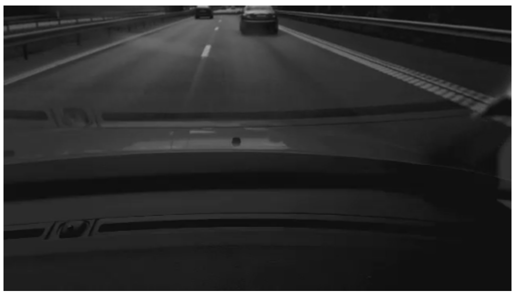
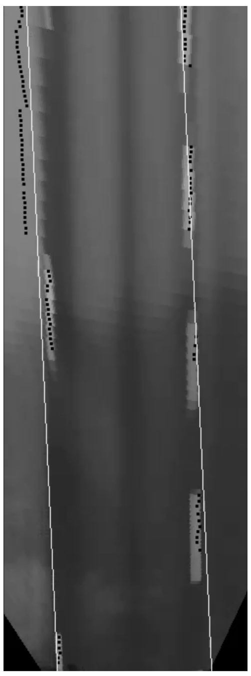
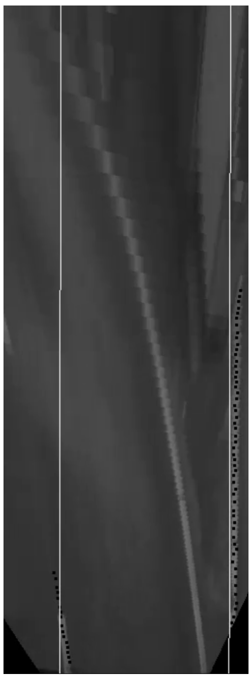
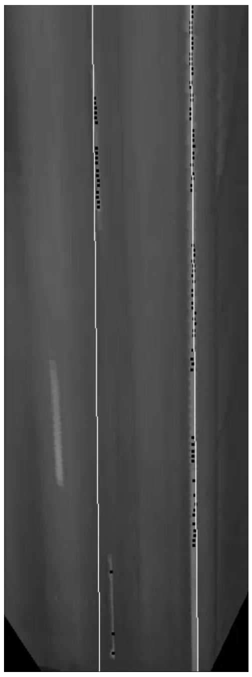

(l) Corresponding to 162s of Figure 4.2c

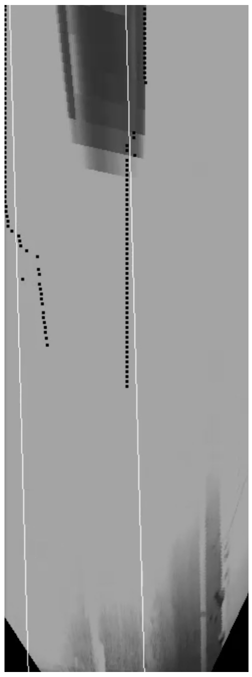
Figure A.1: Driving scene during failure scenarios

A.2 Result of algorithm calculations during failure scenarios


(a)


(b)


(c)


(d)

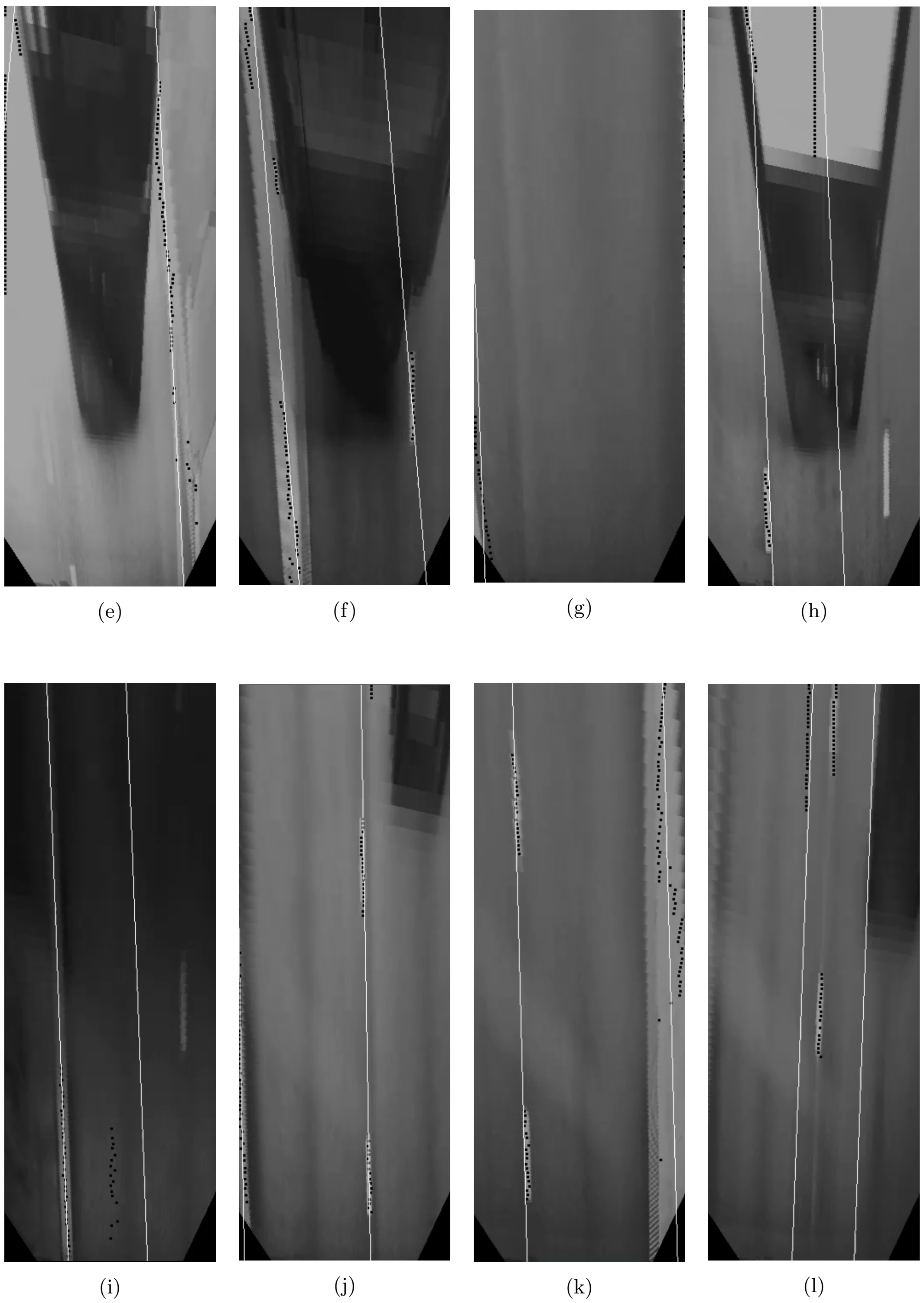


Figure A.2: Result of algorithm calculations during failure scenarios