CHALMERS

Extracting road properties using a Lidar scanner.

Master's Thesis in Engineering Mathematics and Computational Science.

JONATHAN AHLSTEDT

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2013
Master's Thesis 2014:02

Abstract

In this thesis I present and validate methods to recognize roadway geometry features such as road markings, road curbs, road edges, lanes and the drivable area, from point cloud data obtained by a laser scanner. The methods are developed for a Velodyne HDL 64-E Lidar scanner, and the data used will be off-line so the focus lies on accurate as opposed to fast methods. The scanner is usually used in applications of autonomous driving so most methods are developed to work in real time.
The road marker detection method presented uses an intensity thresholding algorithm. A threshold based on the median intensities of each diode is set to determine segments of highly intense points that are classified as road markings. The curb detection method presented calculates the elevation difference between consecutive points in order to find curbs. A road edge detection algorithm and a method to find the drivable area of the road are also presented. The results show that the road marker detection and the curb detection algorithms perform well. The road edge estimation and the lane estimation work well in highway scenarios, but showed limited scenarios in urban scenarios where results were disturbed by special cases. When the road edge and lane estimations were compared to literature they were up to standard. I have demonstrated the feasibility of the drivable area extraction algorithm; however the algorithm requires considerable amount of tuning to attain optimal performance.
The over all results of the thesis indicate that the algorithms developed are a step in the right direction towards finding robust detection methods for road properties using a Velodyne Lidar scanner.

Acknowledgements

I would like to acknowledge the help of my advisor at Volvo Cars, Yury Tarakanov for his constant input, advice and encouragement. Working for Volvo, and being entrousted in a part of an interesting project, was a great opportunity.

I would also like to thank my advisors from Chalmers University of Technology, Aila Särkkä and Mats Rudemo for their advice and patience. It is easy to get carried away and loose track of the final product as time flies by.

Finally I would like to sincerely thank Johan Villyson, my rival and friend, for nominating and vouching for me in the first place.

Jonathan Ahlstedt, Gothenburg 2013/11/18

Contents

1

Introduction

Roadway departures due to driver inattention or poor control of the vehicle is a reason for a lot of collisions every year. In order to alleviate the roadway departures, the lane departure prevention functions by warning to driver or steering intervention are offered by leading car manufacturers within Active Safety functionality. Perception of the extent of the road and lane markings, something that our eyes are usually very good at, is a huge challenge to the camera sensors that are commonly used by Driver Support functions aimed at helping the driver to keep the vehicle from leaving its lane. In order to evaluate the sensor performance, some other robust way to detect these and estimate their locations on the road needs to be used.

When developing the cars of the future there is a large focus on autonomy. We want cars that are able to sense dangers and avoid accidents, without input from the driver. The scanner used in this project is a Velodyne HDL 64-E Lidar scanner. It is a laser range finder with a 360 degree field of vision able to detect almost anything in its surrounding and it is the primary sensor used in nearly every fully automated car to date. For example, Google uses a Velodyne on the roof of their driver-less mapping cars and almost every contestant in the 2007 Darpa urban challenge[1] for autonomous cars used a Velodyne scanner[1, 2, 3, 4].

Positioning with respect to road and its lanes is paramount to any autonomous car, in order to stay within their lane. Without accurate lane detection algorithms it is impossible, for such cars, to stay within their lane. There are also functionalities in modern cars that rely on knowledge of the extents of the road and the lanes. For example both Adaptive Cruise Control (ACC) and Lane Keeping Aid (LKA) are dependent on knowing where different lanes are located with respect to the car. The built in sensors used in these systems are hard to verify and thus if similar functionalities are to come in

[1]The DARPA Urban challenge was a competition where teams developed autonomous cars able to drive a 96 kilometer urban course in less than six hours. The robots were supposed to obey all traffic regulations in addition to dealing with other robots and vehicles on the course.

the future there needs to be a fast and efficient way of verifying both the correctness of the systems and the output of the sensors. The advanced Velodyne sensor used as a reference sensor might overcome this challenge. The Velodyne scanner is too expensive to be included in commercial cars but it is accurate enough to verify cheaper sensors.
Ordinary car drivers rely almost fully on vision when driving, and without a major worldwide adaptation and improvement of the road infrastructure any semi- or fully-automated vehicle needs sensors to process the same visual information as human drivers[5]. For this reason methods for aiding the cars perception of its surroundings are needed in order to achieve a new generation of cars that are able to cope with dangerous situations on their own.
In this thesis I will present and validate methods for lane marker detection, curb detection, lane estimation and road surface detection using a Velodyne HDL 64-E Lidar sensor. The goal is that the method should generate useful good results in any scenario and that the system should be automated. The method should use raw point cloud data so that it does not depend on any other module developed for the scanner.

There are lots of road properties that could potentially be found but to keep the scope of the project reasonable I will concentrate on the following tasks in an order of importance:

- Road markings
- Road curbs and edges
- Detailed geometry of the road surface.

The data will be offline, so the algorithms do not need to function in real time and hence slower more accurate methods can be used. The project will not extend to anything above the road, beside the road or even on the road and is hence limited only to the road.

The report is organized as follows. Chapter 2 presents the current field of research regarding extracting road properties using a Lidar scanner. This is also where the theory behind the different methods and algorithms is described. In Chapter 3, I go through the process of implementing the methods and the different tools used to generate the results. In Chapter 4 results are presented and compared to ground truth in order to validate the robustness and correctness of the methods used. In Chapter 5 an analysis of the results is conducted and finally in Chapter 6 the drawbacks of the results and potential improvements are outlined.

2

Background

The biggest reason why there have been lots of research on autonomous car development is due to the DARPA challenges. Specifically after the DARPA Urban challenge of 2007 there have been lots of development because every participant in the challenge needed to have working, fast and robust road marker detection methods, curb detection methods and road extraction methods. In rare cases the specifics of the methods were explained in some research papers, but most often the contestants only stated which kind of methods they used and what setup their team had [1, 2, 3, 4, 6, 7]. Usually the methods used by the teams were state of the art and the validity of the methods were demonstrated by the fact that they were able to operate a car, in real-time, without a driver.

Road and road marking related methods using Lidars are fairly new and one of the problems due to the novelty of the field is that there is no commonly used benchmark. Different papers validate their results in different ways, and because of this it is very hard to compare the results of different researchers. Therefore, it is rather hard to determine objectively if results using a specific method are good or actually bad.

It is noteworthy to mention that most articles on the subject of road and lane detection are focused on methods that function in real-time because the researchers have autonomous driving in mind as an application. While all real-time algorithms also work on offline data, there might be better more suited methods for detection without the real-time requirement. For example, usually an extended kalmanfilter is used to predict where lanes are likely to appear in future sensor readings but since the future data is already available in the offline scenario we can actually check the true position of the lanes without prediction.

2.1 Lidar

The term Lidar is a combination of the words light and radar[1] and was first mentioned in 1963 in the article , "The Laser in Astronomy" [8]. Lidars transmit beams of light directed by a rotating polygon mirror. The distances to objects are found by measuring the time dilation between an emitted pulse of light and the registered reflection. The intensity of the reflected light is also measured[9]. Advanced Lidar systems are able to produce three dimensional images of the environments consisting of point clouds representing where the light from different beams reflected on objects. Cameras are more common commercial image sensors partly because Lidars are generally too expensive to be used on the commercial market. The advantage of using a Lidar scanner over a camera based scanner is that cameras do not have a robust direct way of measuring the distance to objects. Also, in conditions with bad lighting cameras might not work while Lidars are unaffected. In the DARPA Urban challenge of 2007 most contestants used a combination of camera sensors and Lidar sensors to benefit from their separate strengths, but the methods used were primarily centered around the usage of Lidar scanners.[1, 2, 3, 4]

One of the more prominent Lidar scanners on the market is the Velodyne HDL-64E Lidar scanner. Instead of one laser emitter and rotating mirrors, 64 separate emitters set to different elevation angles rotate around the scanner's horizontal axis. This setup is able to provide a 360 degree azimuth field of view and a 26.5 degree elevation field of view. The Velodyne scanner generates over one million data points per second and produces point clouds with a frequency of 5-15Hz. Each complete rotation yields an accurate image of the surroundings with a range of 120 meters and a maximum error on the range resolution of about 2 cm [10]. The scanner is mounted on the roof of the car to be able to scan in all directions. In such an installation, it is able to scan both the traffic around the host car and the road surface.

2.2 Road marker detection and lane estimation

Lidar based methods for road and lane marker detection involve the crucial fact that road markings are more reflective than the rest of the road surface. By analyzing the reflectivity measurements of the Lidar scanner road markings can be found. One basic method for lane detection is presented in [11], where they used a Lidar mounted in the car grill to generate point clouds in front of the vehicle. They used a threshold to extract the points with high reflectivity. The lateral positions of the extracted points were plotted in a histogram. Another threshold was used on the histogram to find in which y-coordinate the lanes were present. The method was able to find the host lane at least 90% of the time, but one major drawback is that lanes could only be found if they were not curved.
A slightly more advanced attempt to find lanes was presented in [6], where the main

[1]There is a common misconception that Lidar is an acronym for LIght Detection And Ranging originating from the fact that Radar is an acronym for RAdio Detection And Ranging.

scanner was a Velodyne HDL-64E Lidar scanner. They extracted all points on the road with an intensity higher than the median intensity of points on the road. They then looked for clusters of points to indicate that road markings were present. In combination with an elevation curb detection algorithm they were able to find lanes by using a Radon transform on the candidate points. The paper did not supply test results, but the method was used in practice by team AnnieWAY during the DARPA Urban Challenge 2007.
Another method was developed in [12] where a Denso scanner pointed in front of the car was used. They used a polar grid to group measurements to easily be able to compare points of equal distance from the scanner. An adaptive threshold was then used to distinguish between cells containing road with and without markings. A drawback with the scanner was that only ego lane markings could be detected due to the narrow field of view of the sensor. The researchers seem to be able to find the ego-lane[2] on a highway in most weather conditions but to results obtained did not seem very robust or good compared to the results presented in some other articles.

2.3 Curb detection

In urban driving curbs are the most common indication of the edge of the road. Detecting curbs in a robust manner is therefore an important task in many applications. The most common way of finding curbs is by using the elevation data. In [13] the authors use a Velodyne HDL-64E Lidar to find curbs and estimate their positions. To detect curbs they considered scan lines from one diode at a time. They went through the readings of each scan line starting at a point in front of the car once counter clockwise and once clockwise direction to find the left and right curbs using a set of requirements. The candidates were then verified using elevation data. They then used the RanSaC algorithm to estimate the position of the lanes. From visual results lanes were found in every scenario. The individual locations of curb-points were correctly estimated on easy road patches but not as well in more difficult scenarios with lots of cars, pedestrians crossing the road or in instances were the curb was lower than 10 cm.
Looking at elevation differences is not the only way to detect curbs. In the DARPA Urban Challenge entry report of Stanford[1] the authors claim that a vertical displacement threshold can not be set to efficiently detect curbs without a substantial number of false positives [3]. They recommend comparing the range between two adjacent beams. This works because each laser beam of the Velodyne sweeps out a circle of a fixed radius on a flat surface. On a sloped terrain two subsequent rings are compressed. Consequently as two circles sweep over a vertical object the beams generate nearly the same range.

[2]The lane in which the host vehicle is located.
[3]A false positive is a Point classified as a curb point even though it is not.

2.4 Road extraction

To extract the road from a point cloud is to find the points that belong to the drivable area. If curbs or lane markers on the edge of the road are found the drivable area simply compromises all the points in between. If curbs or lane markers are not present, there needs to be another way to find the edge of the road. One such method was presented in [14]. The authors used a basic 2D lidar sensor to get single scan lines of the road. They then used the fact that most roads are sloped in order to drain the roads. From a single lateral scan line the road can then usually be fit by a second or third order polynomial. They then use the RanSaC algorithm to fit the points to a polynomial in combination with an accumulation scheme to find the most probable points belonging to the road. This method assumes that the road can be estimated with a polynomial but does not require any other indications of road edges. One setback with the method is that the RanSaC algorithm fits the largest segment of points first, so if the sidewalk is wider than the road it will classify the sidewalk as the road. The results of this paper were shown in several pictures but no estimate of how well the road was detected was reported.
In [7] an advanced method for road extraction was proposed to identify road regions and road edges. The method was used by the DARPA Urban Challenge winning entry, Boss. While the vehicle had an on board Velodyne for long range scanning the road detection was performed by a forward-looking LMS-200 SICK Lidar. This is a 2D sensor that, like the sensor used in [14], generates single lateral scan lines of the road. Then road candidates are found as large flat segments of the scan. These segments were then classified to see if the candidate is a road segment. Futhermore curbs were found as points on the side of the road segments. According to the paper the algorithm detects road edge points with a false positive rate of 0.83% and finds road segments with a miss rate of 0.55%.

2.4.1 RanSaC

The RanSaC algortihm was first introduced by Fischler and Bolles in 1981 and is an abbreviation for RANdom Sample And Consensus [15]. The RanSaC algorithm is used to fit a model to data contaminated with many outliers. A prerequisite for using the method is the assumption that the model selected fits well to the uncontaminated data. The method is constructed to find the best possible model parameters and does not give a measure of how well the model fits to the uncontaminated data. If the ordinary data points exceed the number of outliers, the method usually yields good results, but since the algorithm is random there is no guarantee that this is the case. Similar to other random sampling algorithms, the ammount of iterations performed decreases the chance to getting a bad result so if the number of iterations is high enough we never get bad results in practice.
The RanSaC algorithm is a very general method for fitting models to contaminated data, and in fact you can use any model as long as you have a way to estimate the parameters of the model just from using n data points.
The algorithm works as follows:

1. Start with an original data set consisiting of l data points.
2. Select n data points randomly without replacement.
3. Fit the model parameters to the selected data set.
4. Construct the consensus set by adding all points in the original dataset with an euclidean distance smaller than t from the fitted model.
5. Refit the model to all of the data in the consensus set.
6. If this model has the largest consensus set, it is the best model so far.
7. Repeat step 2-6 k times

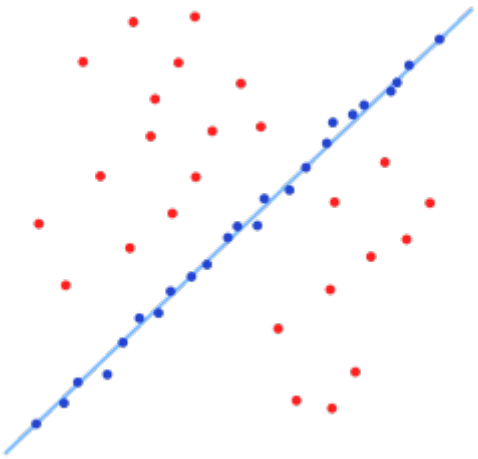


Figure 2.1: A straight line fitted with the RanSaC algorithm.

There are several variants and improvements possible for the basic RanSaC algorithm. One variant is not optimizing the number of data points in the model but rather how well the data fits the model. To do this you have to define how many points there have to be in a model. You then pick the model that minimizes the error of the model fitted by the consensus set given that enough points are in the consensus set. Another common improvement of the algorithm is to iterate step 2-6 until you find a good enough model instead of a set amount of times.
The parameters n and t have to be determined depending on the problem. The same is true for the model used, and for the number of close points required to determine if we have found a good model. The algorithm is stochastic and the number of iterations k determines the probability of generating reasonable results. For this reason, in cases with large amounts of data the RanSaC method is quite time consuming. This is because the method basically tries different model parameters at complete random many times, and then picks the best set of parameters found.

3

Data and Methods

This chapter presents different methods for road property detection using a Velodyne HDL-64E Lidar and the data and resources needed to develop them.

3.1 Data

The methods used in this thesis require data for development and verification. The scanner is expensive and costs approximately 75000 dollars. For this reason, only a few companies and universities are reasy to invest in a Velodyne scanner. Most car companies do not publish the data they collect online and most universities do not have the resources to buy a Velodyne scanner for research purposes. Therefore data sources containing Velodyne data are scarce. Another fact that increases the rarity of usable Velodyne data is that the acquisition needs to include roads, and preferably needs to be synced with a Gyroscope and a GPS. The Gyroscope and GPS synchronization is a way to get a fairly good reading of the motion of the host car. The points in the point clouds collected in the Velodyne data should preferably contain six variables: x, y and z positions, calibrated intensity readings, a diode index and a time stamp.
There were two different sources of data that could have been used to develop the methods presented in this section. The first sourve was the data scanned in the Gothenburg area by Volvo car corporation and the other was the KITTI Benchmark Suite available online for academic use.

3.1.1 The Volvo Car Corporation data

This is a private data source obtained during some test runs performed by Volvo Car Corporation. The dataset so far consists of three runs of around 1-3 minutes stored in large files of around 10-60GB. A Velodyne HDL-64E Scanner in combination with an RT3000 Inertial and GPS Navigation System was installed on a car and the points in

the point clouds contained x,y and z coordinates, intensity, diode id and time.
This dataset was not available by the start of the project.

3.1.2 The KITTI Vision Benchmark Suite

The KITTI dataset is an open available source of state of the art Velodyne Lidar point cloud data for academic use [16, 17, 18]. The dataset contains around 100 different scenarios of 80-300 frames per run, with synced GPS positioning and Gyro readings. Since the Velodyne is set to the standard 10HZ spin rate one scenario translates to 8-30 seconds in real time. Each point cloud consists of around 150000 points of data with x,y,z positions in meters from the scanner and intensity ranging from zero to one. The benchmark suite also includes Matlab and C++ scripts to calculate the positional and rotational difference between frames.

3.1.3 Difference between the two data sets

The main differences and attributes of the data sets are listed in table 3.1.

Table 3.1: A comparison between the different data sources

Features	Volvo	KITTI
Point Variables	x,y,z,I,D,T	x,y,z,I
Calibrated intensity with respect to distance	No	Yes
GPS and Gyro sync	Almost	Yes
Large Variation in Scenarios	No	Yes

The dataset collected by Volvo shows more future potential since the points contain diode index and time stamp variables. The diode index variable is needed in some road property methods. There is a way to estimate the diode index of a point in case the information is not available, but sometimes the estimates are not good enough. The time stamp of individual points can be used to so called ”unwrap” point clouds in order to compensate for faults induced by the assumption that all data in one scan is acquired instantaneously. In other aspects the KITTI data is ready to be used, where as the Volvo data still needs prepossessing. One can manually calibrate the intensity of points with respect to the distance from the scanner since the intensity decreases linearly. Lastly, the acquisition of new data by Volvo has just begun. Soon the Volvo data will possibly include even more diverse scenarios than the KITTI data. Currently lacking from the KITTI data are scenarios in snow, rain, hail and scenarios in lowlight settings where camera sensors may show limited performance,
All that aside the KITTI data while not being perfect, has everything required for the work of this thesis and was available at the start of the project.

A downside to both data sources is the existence of an error in the calibration between the intensity strength of different diodes. Various diodes give differently strong intensity readings. This is not easy to correct since it is systematic but not a scaling error or a transitional error, though possibly a combination of the two. As a consequence, the only way to distinguish between a weak and a strong intensity reading, is to compare points in a sequence of readings from the same diode. Strong intensity readings then correspond to readings stronger than the rest of the sequence. It might be possible to calibrate all 64 diodes by hand in order to get more similar intensity readings from different diodes. Such calibrations have been tried fruitlessly but the topic of how it should be carried out efficiently is beyond the scope of this project.

3.2 Software and Libraries

Two programming languages were used in this project, namely Matlab and C++ in Microsoft Visual Studio 2010. It is easy to start a project working in Matlab due to the high level of the language. With respect to programming, there are many tedious and time consuming tasks in other languages that are easy in Matlab. The work shifted away from Matlab because of the Point Cloud Library available in C++.

The Point Cloud Library, abbreviated PCL, is an open-source library available for academic and commercial use with built in functions to help visualize and work with Point clouds in C++[19]. PCL has lots of use in any application working with point clouds. It has many tutorials for installing and learning the basic functions of the library and has many built in implementations of common algorithms used for different point cloud manipulations. These prebuilt operations can solve many common problems encountered when using point clouds.

Had it been possible it would have been easier and have saved a lot of time if everything was done in Matlab but there are two reasons why this was not possible.

- Matlab does not have a built in tool that is able to visualize big point clouds.
- When working with large datasets the execution of programs in Matlab is too slow.

In the beginning of the project it was reasonable to compute everything in Matlab and then to only use the PCL visualizer in C++ to verify if the methods were working accordingly. As implementations got more time consuming it was more efficient to focus fully on an implementation in C++.

3.3 Road Marker Detection

As shown in [6, 11, 12], lane markings can be found using the intensity data of Lidar scans due to the crucial fact that lane markings have a higher reflectivity than darker asphalt.

The goal of the method is to use the intensity, I, of points on the road to classify if they belong to road markings. We focus on finding lane markings, but the more road markings found the better. An intuitive approach is to classify all points where $I > \tau$ to be road markings, where τ is a threshold depending on the data. This approach is employed in [11] and many other articles. With respect to the data generated by a Velodyne HDL 64-E Lidar simple thresholding does not generate good enough results due to a few reasons.

- The data from just one scan does not generate enough points to fully visualize lane markings on the road. Diodes are separated approximately by 0.4 degrees vertically.
- A calibration error in the scanner causes that the diodes have differently strong intensity readings. This difference in strength is largely but not entirely a scaling error. Therefore one constant threshold for all diodes will not work.
- There are high intensity readings that are not road markings so there will be single points falsely classified as road markings.

I will now propose three methods to solve these problems respectively, namely, adding multiple frames together, adapting the thresholding algorithm and performing a grid noise removal.

3.3.1 Unite multiple frames

One scan from a Velodyne HDL 64-E Lidar generates roughly 150000 points and compared to the data generated by other Lidars, the point cloud is dense. But it is still not dense enough to detect lane markings fully. A way to increase the density of points on stationary objects, like the road is to unite frames using GPS and Gyro information. This procedure is detailed in Algorithm 1.

The concept of considering more than one scan at a time is only possible if you work with offline data and is not common in automotive applications in the literature. There are several benefits to merging multiple frames. The most obvious improvement is that the point density of the road is increased by a factor of 5-20 times depending on how fast the vehicle moves. Hence, any method for road property detection generates more results per unit area. This is why merging frames is beneficial in all parts of this project. Merging frames will also remove most shadowed areas on the road. For example if there is a parked car on the side of the road, the wheels and body of the car will shadow parts of the curb and the road markings behind and under it. If we merge frames this is no longer a big issue, since we get frames from different angles, and the shadow of the car

Algorithm 1 Unite frames

PoseMatrix(i) is a vector which describes the difference of the position and rotation between the first frame and frame i.
CalibrationMatrix describes the difference of the position and rotation between the Lidar and the GPS.
Pointcloud(Frame) is a point cloud of data for a given frame.
UnitedCloud is a point cloud where all the points from all frames are united and positioned correctly relatively to each other.
for all Frames **do**
  Transform Pointcloud(Frame) with CalibrationMatrix and replace the result in to Pointcloud(Frame)
  Transform Pointcloud(Frame) with PoseMatrix(FrameIndex) and replace the result in to Pointcloud(Frame)
  Add Pointcloud(Frame) to UnitedCloud
end for

decreases to only cover areas that are shadowed in every frame. Effectively we are able to find markings and curbs under the car.

3.3.2 The improved thresholding algorithm

There are two ways to handle the calibration error in the diodes, the main cause why the simple thresholding algorithm does not work. One method is to calibrate the separate diodes with clever functions to remove the error. This might be the best way to solve the problem but as previously explained the error is not a simple scaling error, so a good calibration is not easy to achieve. From experiments it seems that the error is non-linear as well and it is not easy to develop an automatic scaling method. The other method is to use some kind of adaptive threshold to determine what is a low intensity point on the road and what is a high intensity point on the road. In this thesis we choose the latter strategy. The improved thresholding method is detailed in Algorithm 2.

Algorithm 2 Road marker detection

α,β are constants depending on the data.
for all Diodes **do**
  M = CalcMedianIntensity(Diode)
  for all Points **p** in Diode **do**
    I = Intensity of **p**
    if $I > \alpha M$ **and** $I > \beta$ **then**
      p is classified as a road marking.
    end if
  end for
end for

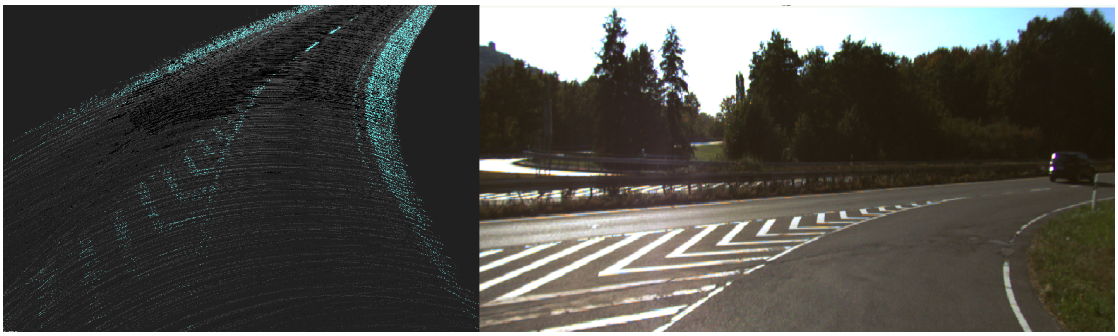


Figure 3.1: The left picture shows a typical result after all steps of the road marker detection method have been performed. The right picture shows a camera image of the surroundings. Note that grass is easily located visually from the intensity data.

The values $\alpha = 1.3$ and $\beta = 0.36$ were used to generate the results of this thesis. The first part of the line
if $I > \alpha M$ **and** $I > \beta$ **then**
handles the diode intensity error described in the first point above. We only consider points with an intensity significantly larger than the median intensity of a diode. Since the goal of this threshold is to express M as the intensity of a normal road point the median is more suitable than the average intensity in a diode. By setting M as the mean intensity instead, high and low values will distort the result. The second part of the if statement is a normal threshold in order to remove noise. The value of β should be low enough not to remove intensity readings of actual road markings but high enough to remove noise. After extensive testing it was found that for the KITTI data the $\beta = 0.36$ value is low enough to remove a lot of noise induced by a few faulty diodes. The α value describes how much larger than the median an intensity reading has to be to be classified as a road marking. The value $\alpha = 1.3$ resulted in a good trade off between true positives and false negatives when working with the KITTI data. In figure 3.1 a typical run of the algorithm is compared to a camera image to give the reader a feeling of what the results look like. More results will be presented in Chapter 4.

There are a countless number of possible adaptive thresholds, and during the development of this particular one, over ten different variants were tried. There are two known problems with the current method.

- Generally, the terrain on the side of the road generates high intensity readings. This causes the median intensity to be too high and sometimes road markings are not found.
- Low intensity diodes are sometimes removed completely by the β threshold.

One way to handle the first issue could be to check the median for the closest 100 points to the point being evaluated. To solve the other issue one could calibrate only the

low intensity diodes, or use a special threshold for low intensity diodes. Some alternative thresholds will be presented in Chapter 4.

3.3.3 Grid noise removal filter

To further remove noise from the high intensity points extracted using the methods above, we present Algorithm 3.

Algorithm 3 Grid noise removal

a,b and c are constants depending on the data.
Create grids with size a in x direction, and b in y direction where the x-y plane is the ground plane with x in the direction of the moving car.
for all Grids **do**
  N = Number of road marking points in Grid
  if N < c **then**
    All points **p** in Grid are no longer classified as road markings
  end if
end for

In this section only a light filter with the values $a = 0.3$, $b = 0.3$, $c = 2$ was used on merged frames to generate results. This filter was used many times with different settings through out the project. The filter can be used on separate frames before we merge, or after the merging of frames for different results. The values of a and b determine the size of the grids. An alternative way is to focus on finding lane markings by choosing $a > b$. Those grids result in points being more likely to be removed if they are not close to other points in the x-direction, an attribute all point on a lane marking share. The value of c should reflect how many points there are in clusters of outliers, so that they can be removed. It is important not to set c too high, as the filter might remove points that are road markings. The values used in this project were found through testing.

By applying Algorithm 1,2 and 3 we have a working method for extracting road markings. Results of the method are shown in Chapter 4.

3.4 Curb detection

The easiest and most common way to find curbs with a Velodyne scanner is to look at differences in elevation data. As shown in [1], one can consider the range data of different points to achieve the same results. The specifics of how to do this and why it might be a better method is discussed in Chapter 4. The elevation based algorithm presented in this paper was largely inspired by [13], in particular how the authors iterated through one diode at a time and used certain conditions to find curbs.

One of the prerequisites for finding curbs using elevation data is that there actually exists a curb and that the curb has a reasonable elevation above ground. In [13] the methods were developed to find curbs 10 cm above ground. The goal of the method presented here is to find curbs as efficiently as possible, no matter how high above ground they are. The first algorithm presented in this section finds potential curb points based of elevation data.

Algorithm 4 Elevation based curb detection algorithm

δ and γ is a constant depending on the data.
e is a constant integer.
We use a convention so that $\mathbf{p}(i,d)$ corresponds to the point number i of diode d, and $\mathbf{z}(i,d)$ is the height in meters of $\mathbf{p}(i,d)$.
for all Diodes d **do**
  n = number of points in d.
  for Index i running from e to n-e **do**
    j is a set of integer values such that $j \in [i-e, i-e+1, ..., i, i+1, ..., i+e]$
    Max = the largest value of $\mathbf{z}(j,d)$.
    Min = the smallest value of $\mathbf{z}(j,d)$.
    $D = |Max - Min|$
    if $D < \delta$ **and** $D > \gamma$ **then**
      $\mathbf{p}(i,d)$ is classified as a potential curb point.
    end if
  end for
end for

The parameter values used to generate the results of this section were, $\delta = 0.18$, $\gamma = 0.1$ and $e = 2$. The δ, γ values are given in meters. Algorithm 4 iterates through one diode at a time. A point is a curb candidate if the largest height difference of the point and it's $2e$ closest neighbors lie in the interval $[\delta, \gamma]$. There is a certain trade off that has to be accounted for when choosing the value of δ. The value of δ should be set high enough to not include noise from small height differences on the road. The value of δ should also be low enough to correctly classify as many curbs as possible. The specific value $\delta = 0.018$ was chosen with this trade off in mind. The use of the noise removal grid presented in algorithm 2 is a way to lower the value of δ so that more curbs can be found.

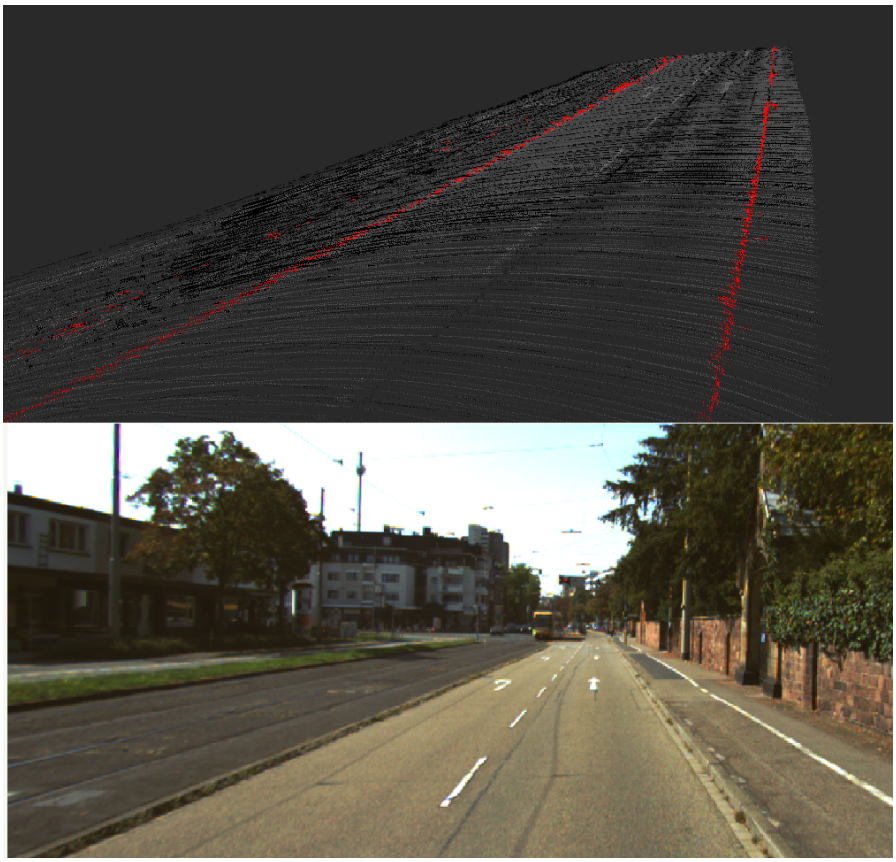


Figure 3.2: The red dots in the above picture are detected curb points found by using Algorithm 4 on a set of merged frames together with two noise filters described in Algorithm 3. The picture below is a camera image of the same area. Images are not perfectly aligned, but curbs are found at the correct locations, with almost no false detections on the road.

The value of γ should be chosen high so that it does not remove any curb candidates. The job of γ is to remove large differences due to errors in the sorting of the data. When using the KITTI benchmark suite data, we do not know what diodes emit what points, and we have to use an estimation explained in Chapter 4 to find which points belong to which diode. Usually, the γ removes false classifications due to points ending up in the wrong diode. The final value chosen, $\gamma = 0.1$, is not a sensitive value and in fact any value in the interval [0.06,0.2] would do. The variable $e = 2$ was found to be an ok value from testing. One aim with the choice of e is to let $1 + 2e$ be equal to approximately the number of points in a diode that belong to a curb. If another value of e is chosen, the values of δ and γ need to be recalibrated.

A drawback is that the method finds holes in the road just as easily as curbs, it only takes the change in height in account. That is why any sudden hight change on the

road will be detected. In particular, a common problem is the tires of parked cars, rails from trams and holes in the ground are common. One way to somewhat deal with this is to add knowledge about if there is a sudden drop or if there is a sudden increase in hight.

After curb points have been detected two noise removal filters are used to improve the results. First, Algorithm 3 with values $a = 3$, $b = 3$, $c = 5$, is used on every separate frame before merging. In this case very large grids are used to remove noise on the plain road. Then, after the merging of frames explained in Algorithm 1, we use another grid noise removal filter with values $a = 0.4, b = 0.4, c = 10$. The second filter requirement is nearly only fulfilled by curbs and other objects on the ground. An example of the results produced is shown in figure 3.2.

Potential curb points can be used to find the edges of the drivable area. If we have data without much noise this can be done by Algorithm 5.

Algorithm 5 Road edge detection

f, g and h are problem dependent variables.
for all Diodes d **do**
  n = points in d.
  Find a point directly in front of the vehicle.
  i_{start} = The index of this point
  for Start from $i = i_{start}$ and iterate backwards to $i = h$. **do**
    c = number of curb points among $\mathbf{p}(i)$,$\mathbf{p}(i-1)$,..., $\mathbf{p}(i-h)$
    if $g \leq c$ **then**
      Set $\mathbf{p}(i-f)$ as the edge of the road.
    end if
  end for
  for Start from $i = i_{start}$ and iterate forward to $n - h$. **do**
    c = number of curb points among $\mathbf{p}(i)$,$\mathbf{p}(i+1)$,..., $\mathbf{p}(i+h)$
    if $g \leq c$ **then**
      Set $\mathbf{p}(i+f)$ as the edge of the road.
    end if
  end for
end for

The goal of Algorithm 5 is to in each diode, find the points at the left and the right edge of the road. In this case the road refers to the drivable area. Edges are detected if there is a curb present or if the road edge altitude differs from that of the road. The variable h and g determine how big the search interval for curb points is and how many curb points we need to find in order to guess that we have found an edge. Since curbs are usually about 6-8 points wide a good interval is about $h = 10$, with a $g = 4$.
A typical curb is usually approximately 8 points wide[1] and around 7 of the 8 points

[1]The number depends on the angle from the host car to the curb and how far the scanner circle is

are detected to be curb points. Since our search interval is 10 units wide there is a 5-2 point offset between the found edge point and the actual edge point. The variable f is designed to compensate for this error and the value $f = 4$ was found to work well through testing.
There are several problems with the current version of the algorithm. To start with, since we do not have the diode id of each point we can only calculate the points using curb points found by Algorithm 4 and the filter in each frame. We can not use the merged frame filter since we need exact diode id to find which points belong to which diode. This could be fixed but I leave it as future work. A way to compensate for this error is to simply use a filter on the generated edge points as well as to remove any points found in the middle of the road. Another error is caused by parked cars. The wheels of cars are very similar to curbs and if multiple cars are parked in a row an edge similar to that of a curb edge will be found. While these are serious problems they don't generally affect the results of the algorithm, other than in rare cases.
A typical result of the road edge detection method presented is shown in figure 3.3. Through inspection of camera images it can be seen that the road edges are found well, with a few outliers in the middle of the road.

from the host car.

Figure 3.3: This figure represents a typical example of how curbs are found using Algorithm 5. Yellow points are the estimated road edges.

3.5 Road extraction

There are two methods described to extract the road geometry described in this Section.

3.5.1 Drivable area extraction

We start by presenting a simple method for finding points that belong to the drivable area between the road edges detected in Section 3.4.

In principle Algorithm 6 is a reverse of Algorithm 4. We find intervals of points with low altitude change. This works for the simple reason that the drivable area is usually relatively flat. The variable τ describes how low the maximum accepted level of altitude change on the road. The variable m determines how large the search interval of the considered points is. In this project I used the values $\tau = 0.01$ and $m = 4$. Here it is important that τ is lower than the minimum altitude change for a curb point. Values for m and τ can easily be found through testing.
An even simpler method could be to state that all points between the road edge points

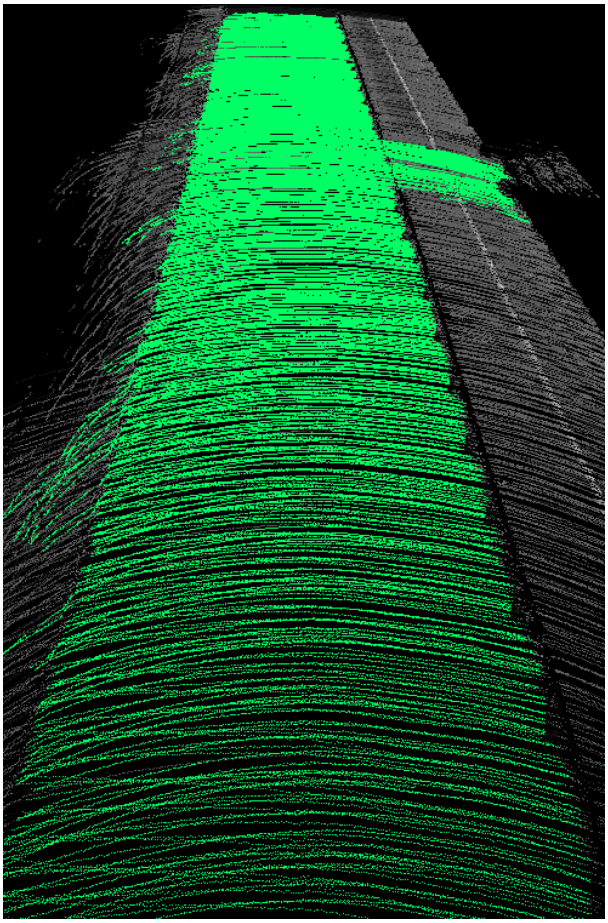


Figure 3.4: This figure shows some results after applying Algorithm 6 and Algorithm 3 on merged frames. The green points in the picture represent the drivable area. There are some errors in a few diodes, but generally the fully green area is the road. These errors can be seen in the figure as lines of greens on the left of the extracted road. The green area to he right of the road depicts a car exit from a property.

Algorithm 6 Points belonging to the drivable area.

Here m and τ are problem dependent variables.
for all Diodes d **do**
  n = points in d.
  Find a point directly in front of the vehicle.
  i_{start} = The index of this point
  for all Indexes i of points between the right and the left road edges of the diode, if present. **do**
    if i is closer to the left edge then **then**
      D = the largest difference in altitude between the points $\mathbf{p}(i)$,$\mathbf{p}(i-1)$,..., $\mathbf{p}(i-m)$
      if $D < \tau$ **then**
        $\mathbf{p}(i)$ is a point that belongs to the drivable area.
      end if
    else
      D = the largest difference in altitude between the points $\mathbf{p}(i)$,$\mathbf{p}(i+1)$,..., $\mathbf{p}(i+m)$
      if $D < \tau$ **then**
        $\mathbf{p}(i)$ is a point that belongs to the drivable area.
      end if
    end if
  end for
end for

belong to the drivable area. The advantage of using Algorithm 6 is that sometimes the road ends, even when there is no curb detected. In these cases the requirement of a small altitude change usually does not allow points not on the road to be falsely classified as road points. Without a noise filter the results of this method are generally not great, but by applying the filter in Algorithm 3 with parameter values $a = 0.5, b = 0.5$ and $c = 60$, most of the noise is removed.
In figure 3.4 we can see a run of the drivable area detection method. This method aims to classify all road points. Since there are many road points we also generate many false classifications even though the rate of false classifications is not high.

3.5.2 Curve fitting of the road edge

The next method implements a version of the RanSaC Algorithm presented in section 2.4.1 to give a curve representation of the edge of the road. The model used is several segments of second order polynomials. The methodology is quite simple.

Algorithm 7 RanSaC based road edge curve estimation

Start with the merged frame data with road edge detections.
We assume the data can be represented by a second order polynomial for every h meters in the x direction.
We divide our data into sections each h meters long in the x (longitudinal) direction.
n,m,d,h and q are problem dependent variables.
for all sections **do**
  for m iterations **do**
    Sample n edge points without replacement.
    Fit a second order polynomial to the sampled data.
    Construct the consensus set, by including all points within distance d of the fitted polynomial.
    C = number of points in the consensus set.
    if C $> q$ **and** C highest so far **then**
      Fit a new polynomial to the consensus set and set it to be the best solution so far.
    end if
  end for
  Use the polynomial generated by the largest consensus set.
end for

We do one run of Algorithm 7 for only left edges and one for only right edges. At first only higher dimensional polynomials were used over larger areas. This was not a bad technique but often the edges of the curve were not as neatly fitted. As stated in the algorithm, a perquisite for the algorithm working is that for every h meters in the x-direction there exists a second order polynomial that describes the road edge well. Even if it is not verifiable, this is required for the RanSaC algorithm to work well.
The variable values used in this project were $n = 4$, $m = 4000$, $d = 0.1$ and $q = 60$, $h = 10$. The variable n sets the number of random edge points drawn in the first step. The number of points should be at least the order of the polynomial fitted +1, but usually more. A high value for n requires more iterations, but generates better results in general. Usually, it is better to increase the number of iterations and keep n as close to the order of the polynomial +1 as possible. The value for d describes how close a point has to be to the estimated curve to be a part of the consensus set. A high value will falsely include noise into the consensus set, and a low value might result in almost no points in the consensus set. h simply represents how large segments we consider when dividing the data. In general you want h to be small enough to be able to model sharp curvatures. Finally q is the number of points required in the consensus set in order to make a prediction at all.
As of now this method is only usable on paths with added frames where the car has not turned 90 degrees. The second order polynomials are based on the fact that the x axis is the direction of the car, and since we are working in the coordinate system of the first

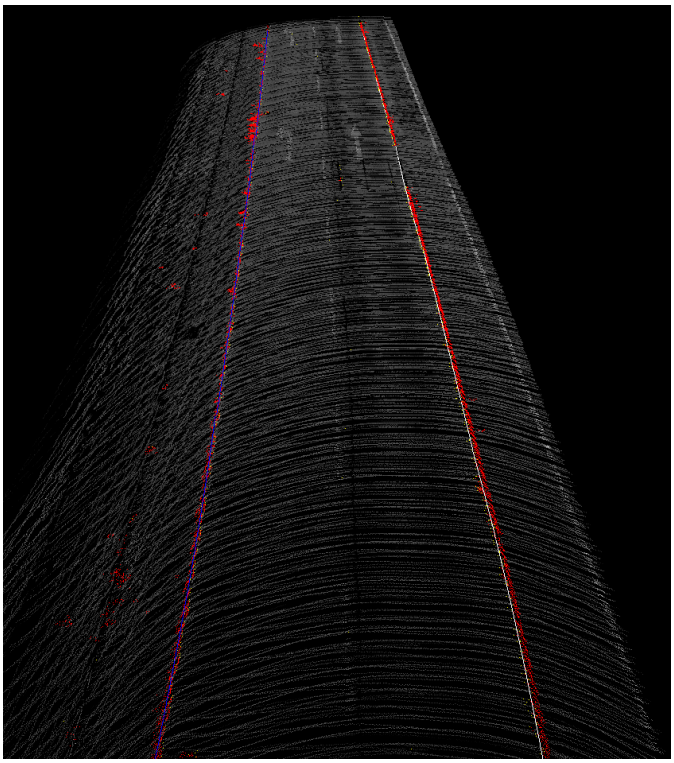


Figure 3.5: In this figure some typical results of Algorithm 7 are shown. The blue line represents the estimated positions of the left edge of the road, and the wite line is an estimation of the right curbs.

frame when merging the consecutive frames, the x axis is the direction of the car in the first frame. In order to deal with this we need to consider what axis is really the direction of the car. The method also requires curbs or other elevation based ends of the road.

In figure 3.5 a run of the Algorithm is shown to clarify what results look like. The blue and the yellow lines show the detected curb positions.

3.6 Lane Estimation

Lane estimation in this paper is performed almost in the same way as in Algorithm 7. The main difference is that we do not try to find a fixed number, but rather the number of lanes are known to vary. To solve this we use the RanSaC algorithm multiple times on the same road segment of data. If a lane is found we remove all points that are close to the estimated lane marking and we start over to find another lane. If no lane is found we stop. The adjusted algorithm is presented in Algorithm 8.

Algorithm 8 Lane marker curve estimation

Start with the merged frame data with road markings detected.
We assume the data can be represented by a second order polynomial for every h meters in the x direction.
We divide our data into sections each h meters long in the x (longitudinal) direction.
n,m,d,h and q are problem dependent variables.
for all sections **do**
  for m iterations **do**
    Sample n road marking points without replacement.
    Fit a second order polynomial to the sampled data.
    Construct the consensus set, by including all points within distance d of the fitted polynomial.
    C = number of points in the consensus set.
    if C $> q$ **and** C highest so far **then**
      Fit a new polynomial to the consensus set and set it to be the best solution so far.
    end if
  end for
  if A good enough consensus set was found **then**
    Save the polynomial and start over with a data set where the points in the best consensus set and close points are removed
  else
    We stop.
  end if
end for

The parameter values used here were $n = 4$, $m = 4000$, $d = 0.1$ and $q = 500$, $h = 30$. There are two deviations difference from Algorithm 7. Higher values of o and h are used. This is because sometimes only one or two dashed lines are present in a 10 meter interval. That is why the intervals are larger. Therefore I also need to increase the number of consensus points needed for a good curve estimation, to compensate for the larger intervals but also to compensate for the fact that there are many lane marking points.
The other difference is that Algorithm was run two times for the left and right curbs where as Algorithm 8 finds the most clear lane estimation first, and then checks if that

lane is good enough to be considered a lane. If so is the case then all points on and close to the lane is removed in such a way so that it does not affect other lanes. Then we start over and perform one iteration of Algorithm 7 with parameter values described above, and check if the solution is good enough. This is repeated until no good solution was found and we are done. In practice this means that if there are no lanes at all in the data, hopefully no good match could be made and we conclude that there are no lanes. To sum Algorithm 8 up, we perform Algorithm 7 several times on road marking data, with adaptations required to ensure that the same lane is not found several times.

4

Results

In order to find how well the methods presented in chapter 4 work we go through a variation of visual results. In the figures 4.1 to 4.7 visual results from six different scenarios are shown. There are no large differences in weather in any of the data sets and they were generated either during 2011/09/26 or 2011/10/03. The first picture is a photograph of a section of the road, taken from one of the four cameras installed on the vehicle used to generate the data and the location is near Karlsruhe, Germany.
The second image shows a point cloud in the PCL visualizer of the same location. In the visualizer the points estimated to be road markings are marked by cyan. Other points receive a color depending linearly on their intensity, where a point with 0 intensity is black and a point with intensity 1 is white. The third picture in every result sequence also shows a point cloud visualized in PCL, with red points marking the detected curb points.
By comparing the original photo to the two different results from the visualizer it is not hard to visually verify that the methods seem to be working as a whole. We still need an objective verification of the results. We start by considering the line marker detection results. In the bottom picture in figure 4.4, the location of the road markings are visible due to their high reflectivity. In general a perfect road marker detection algorithm would find all points where the reflectivity stands out that are also road markings and hence we can, by careful study of a close up of a known lane markings find which points are supposed to be classified as road markings. In figure 4.8 the left picture shows a singled out a white arrow on the ground. We know by inspection of the camera image the outlines of the arrow. By counting every single point within the boundaries of the arrow that has a high intensity reading possibly resulting from the high reflectivity of the white paint and comparing that value to the number of road marking points found by the algorithm we get a rough estimate of the detection probability of a single road marking. By counting, 593 points with an intensity that was visually more intense were found and on the same arrow the algorithm found 542 potential road marking points.

Hence in this case $\frac{542}{593} \approx 91.4\%$, of the road marking points were found.
We may conduct a similar visual experiment on the data of a known curb. In this case we only consider the hight difference of a piece of curb and count all points that have a discernible height difference as a result of the known curb. This is much harder than in the previous example. On a small piece of curb I found 44 curb points by inspection and the algorithm found 41 of them.

Figures 4.9-4.10 shows the results of the estimation of the position of the left and right edges of the road based on the presence of curbs. These pictures are color inverted so that the blue lane estimations are easy to see. The different pictures in the figure generally show an overview of about 50 meters of road. The red points in the pictures represent the probable curb points and the blue lines describe several short second degree polynomials that approximate the geometry of the edge of the road.
The estimates of the location and geometry of the lanes are shown in figures 4.11-4.12. As explained in chapter 3, the estimated road marking points are used to do this estimation. Road markings are colored cyan and the position of the estimated lanes are marked by the red lines. The lines consist of several second order polynomials. The results in figures 4.9-4.12 are easy to verify if they are correct visually, but in order to get a more exact measure of the accuracy we could map the estimated curb and road marking points onto a photo image by using transformation data supplied by the KITTI creators. This has not been done due to the time frame of the work, but alas visual verification is still possible by comparing photo images to Lidar data results.

The results of the detection of the drivable area is shown in figures 4.13-4.14. These results are both good and bad. The relative amount of points classified correctly and incorrectly seems good from visual inspection, but there are large sections of areas not classified well.

Figure 4.1: All three images depict the same location. The first is a photograph of the road, the second shows road markings as cyan points and the third marks potential curb points as red points. Both curbs and road markings were found in this case.

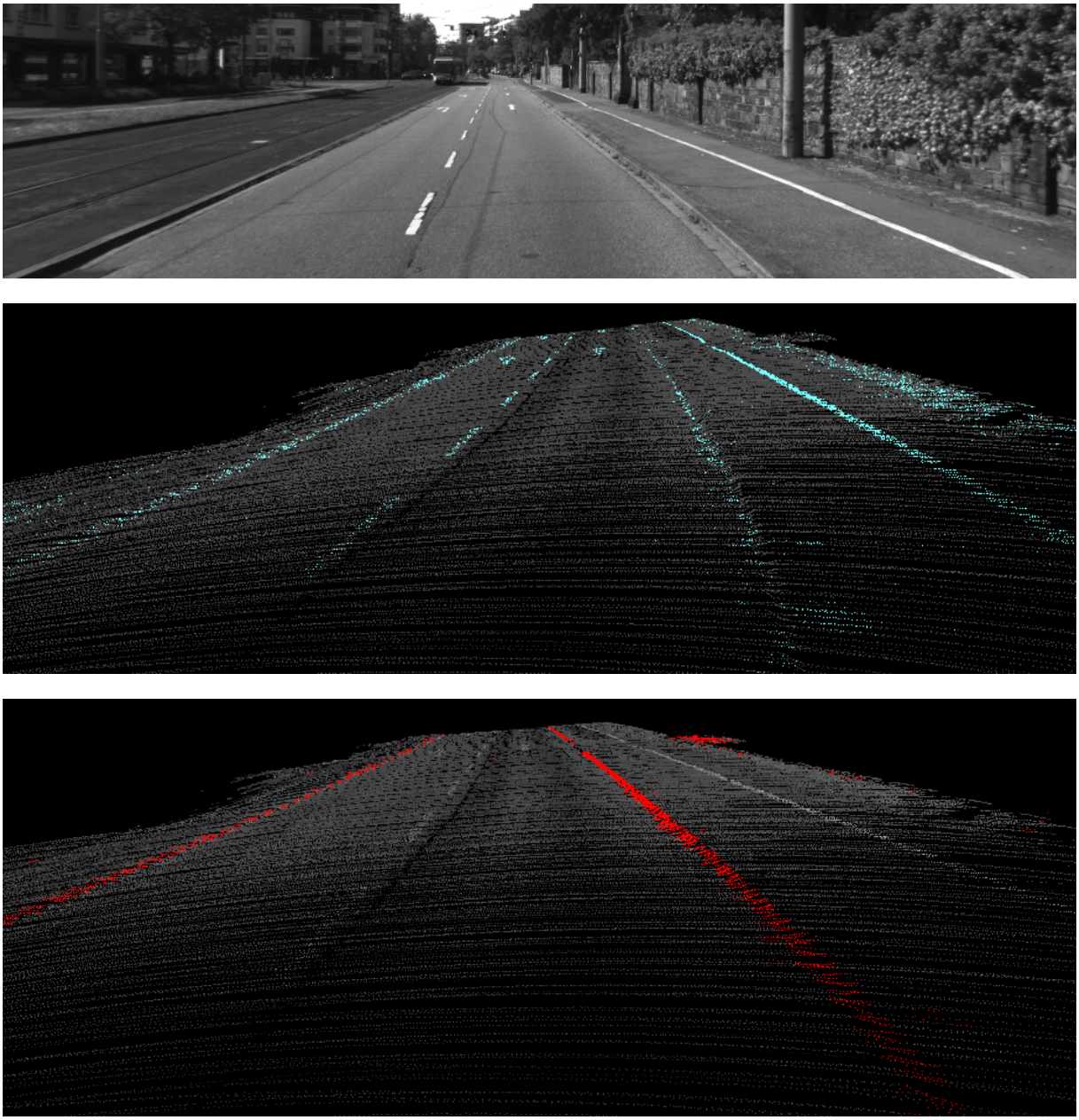


Figure 4.2: All three images depict the same location. The first is a photograph of the road, the second shows road markings as cyan points and the third marks potential curb points as red points. The distinctive curbs on both sides are found. The dashed lines and arrows in the road are found but a lot of points on the curb also have a high intensity.

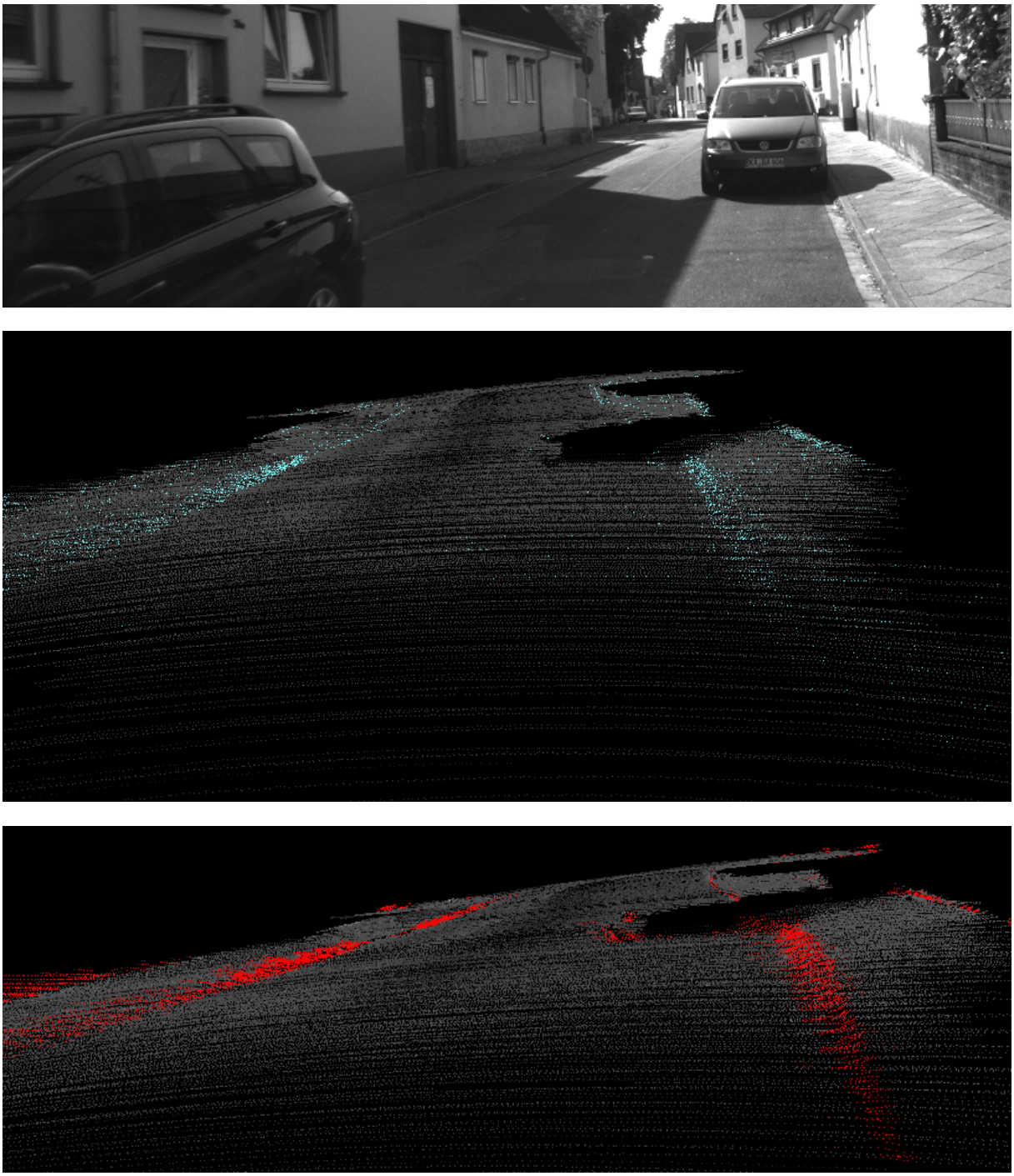


Figure 4.3: All three images depict the same location. The first is a photograph of the road, the second shows road markings as cyan points and the third marks potential curb points as red points. In this case no road markings were present and the only high intensity points are on the side of the darker road. The curbs are found quite well in this scenario.

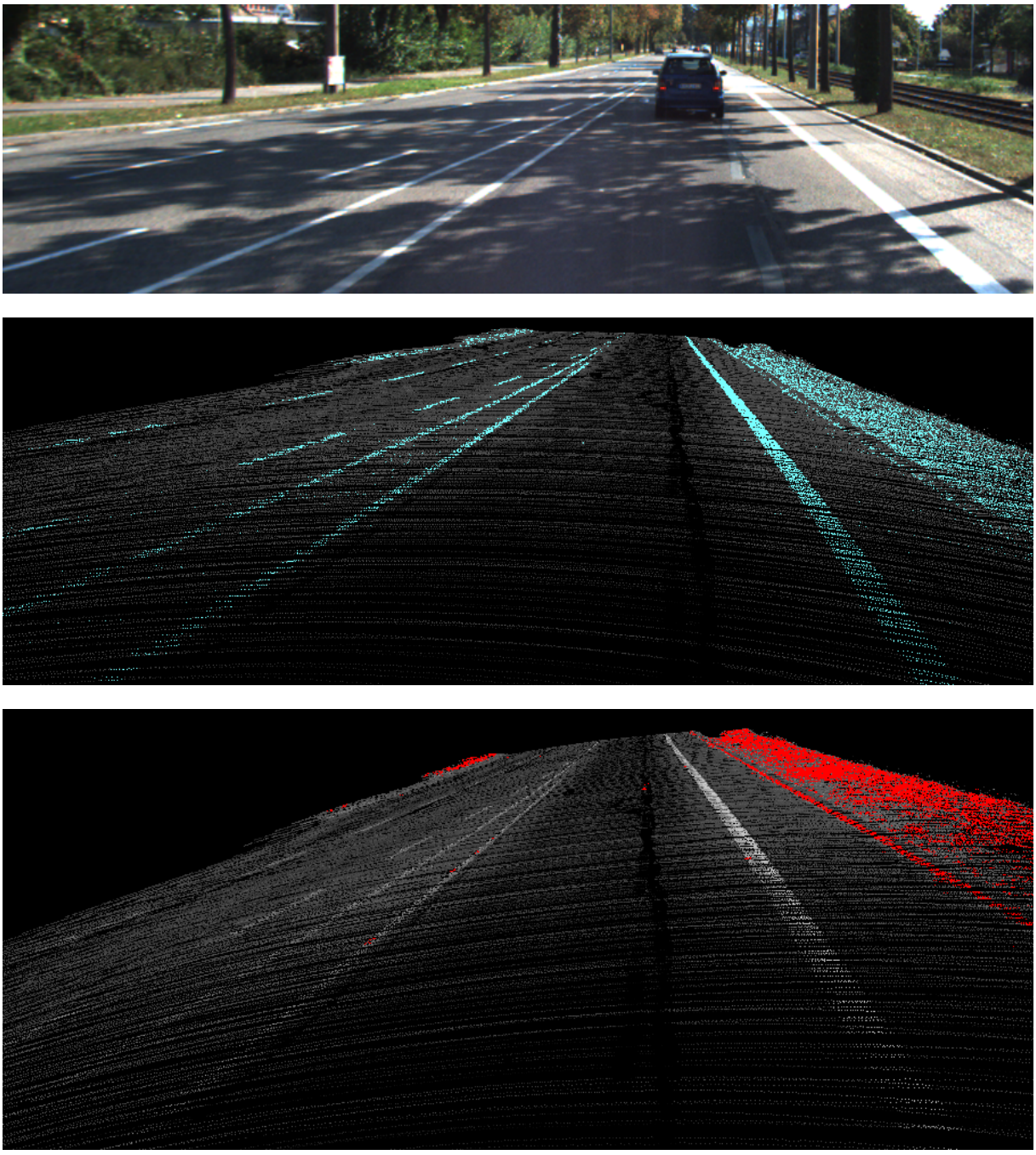


Figure 4.4: All three images depict the same location. The first is a photograph of the road, the second shows road markings as cyan points and the third marks potential curb points as red points. In this scenario there are many very visible lines, present and found. The curb detection picture shows that almost no points on the road are falsely classified as curbs.

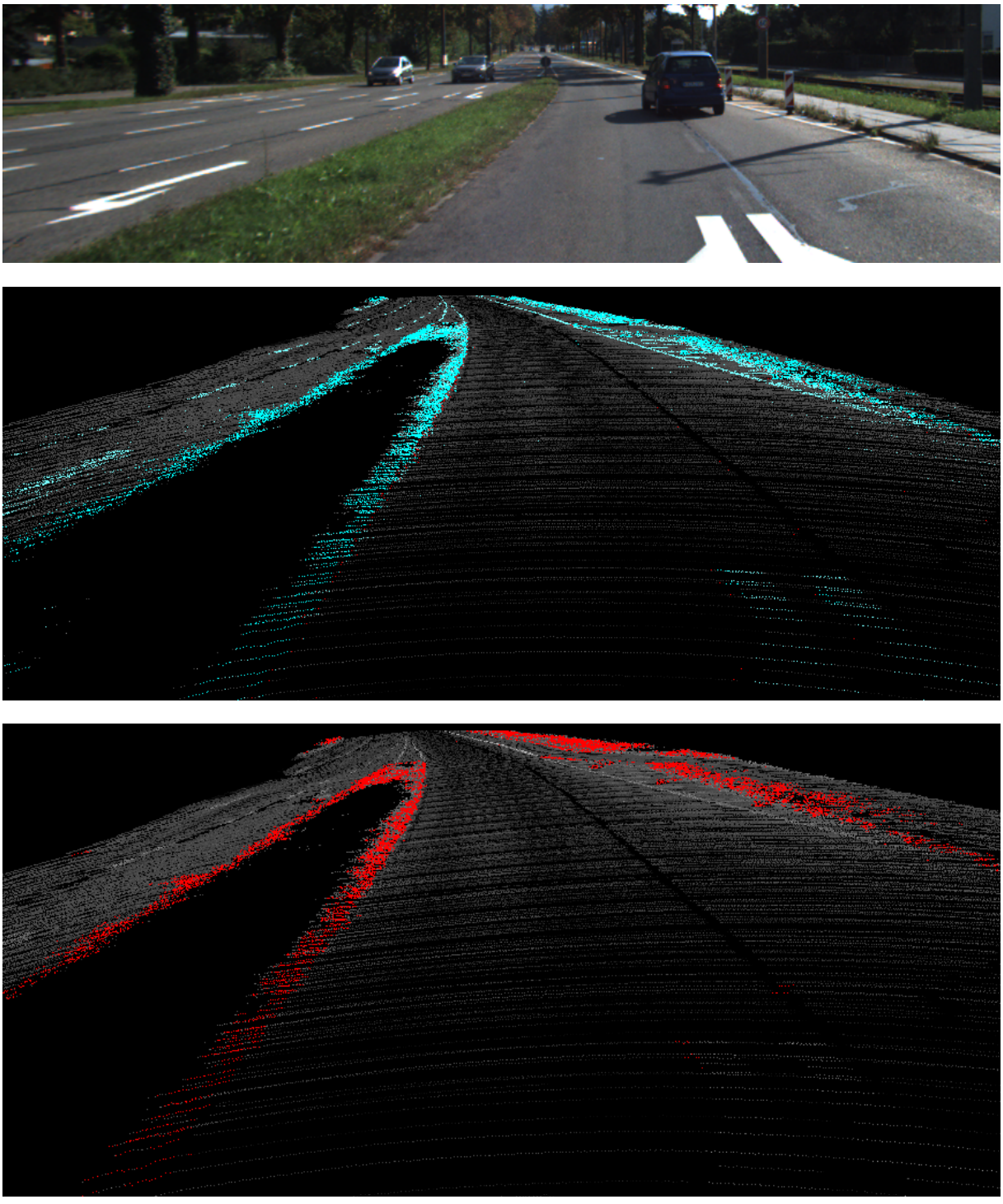


Figure 4.5: All three images depict the same location. The first is a photograph of the road, the second shows road markings as cyan points and the third marks potential curb points as red points. This is the same location as in figure 5.4, only about 30 meters further back. To the left is a patch of grass that is detected both by the line marker detection and the curb detection. Such patches sometimes sever the ability to find road markings.

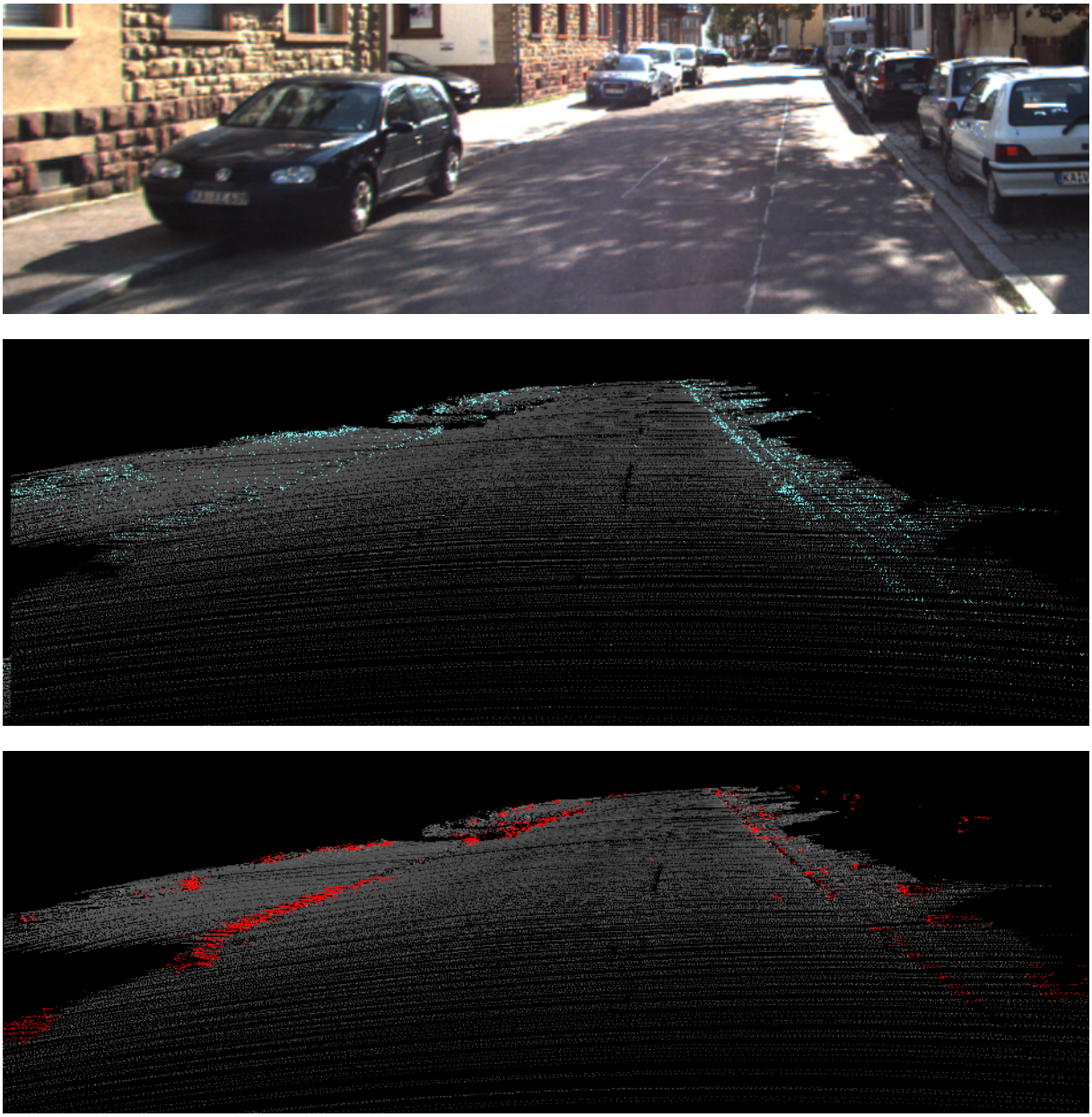


Figure 4.6: All three images depict the same location. The first is a photograph of the road, the second shows road markings as cyan points and the third marks potential curb points as red points. In this case there are no road markings. The curb on the left is found, but the curb on the right is not high enough to be detected.

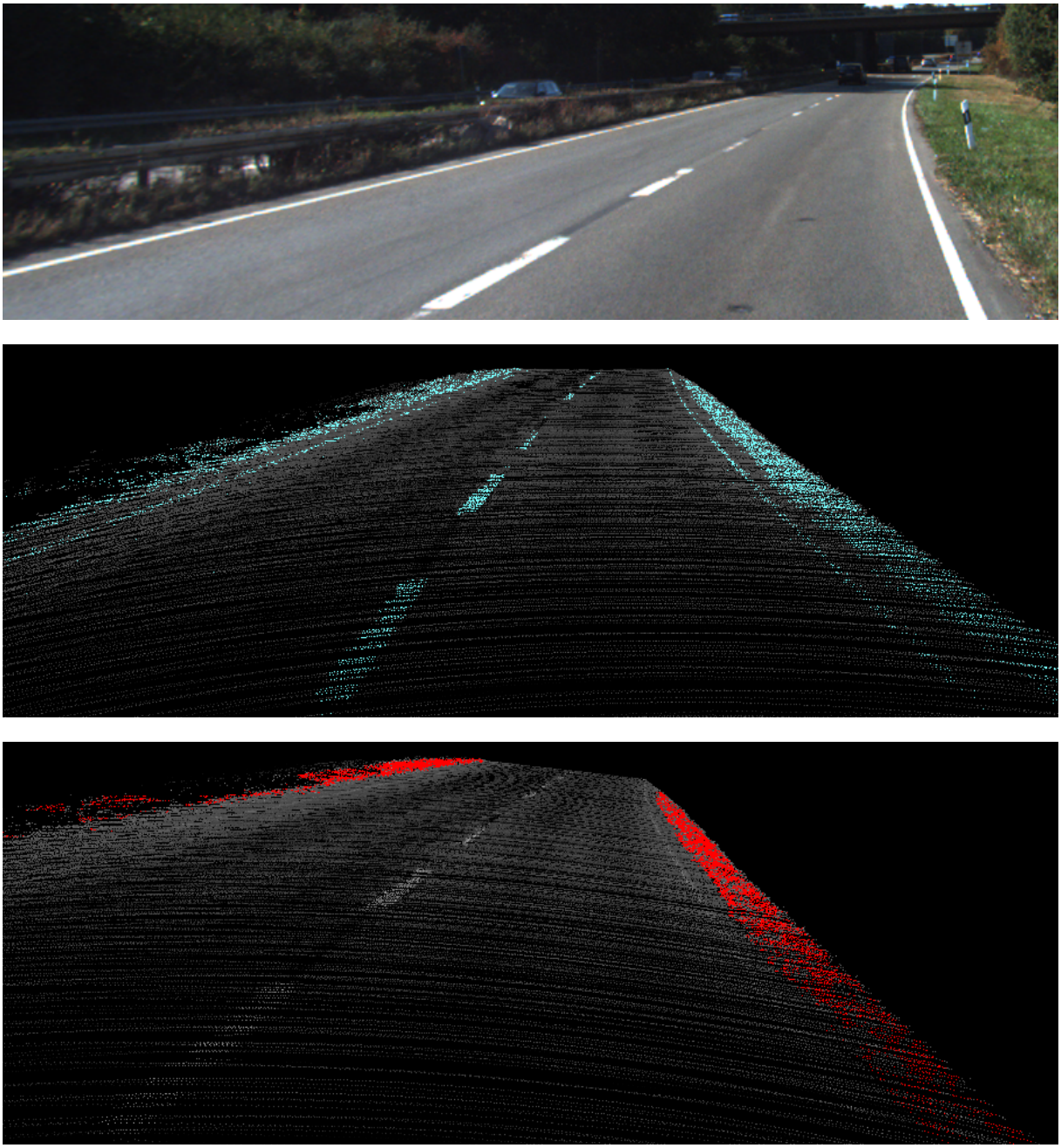
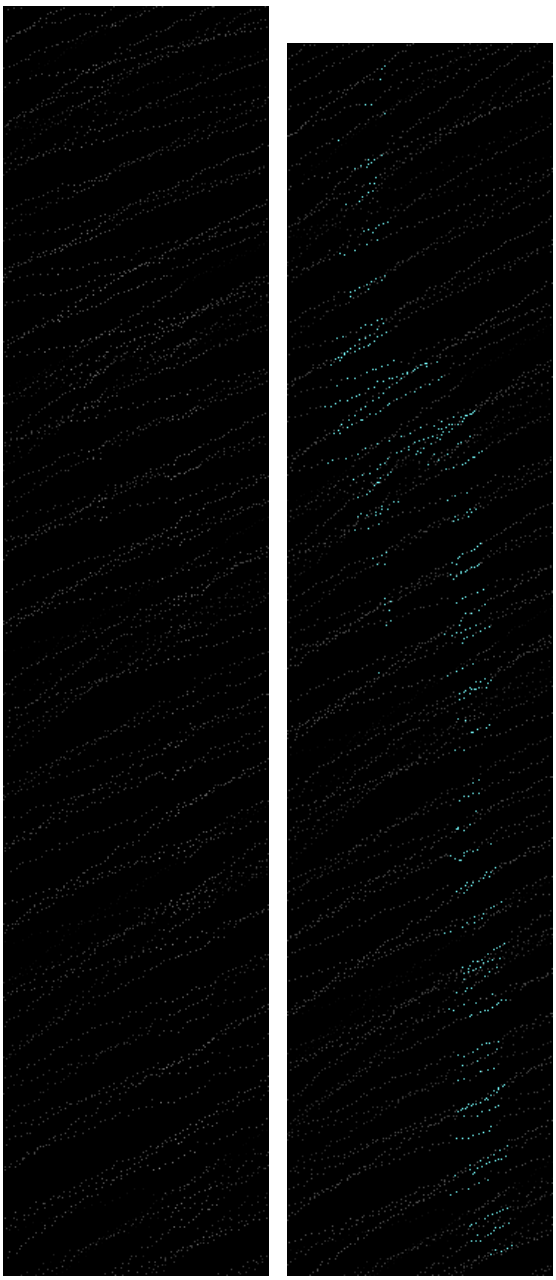


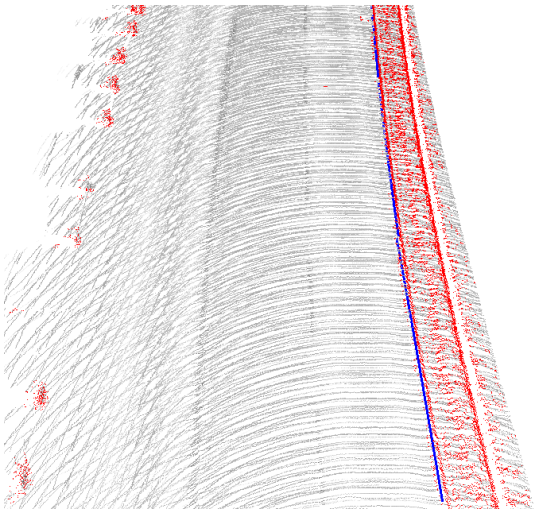
Figure 4.7: All three images depict the same location. The first is a photograph of the road, the second shows road markings as cyan points and the third marks potential curb points as red points.



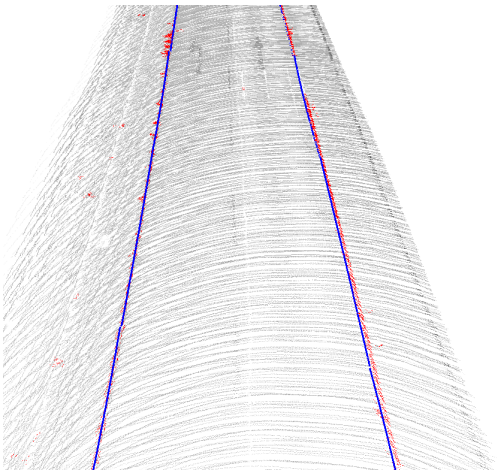
(a) By carefull inspection 593 points on the line had a noticably increased intensity reading.

(b) In the same arrow ficgure there are 542 road marking points detected.

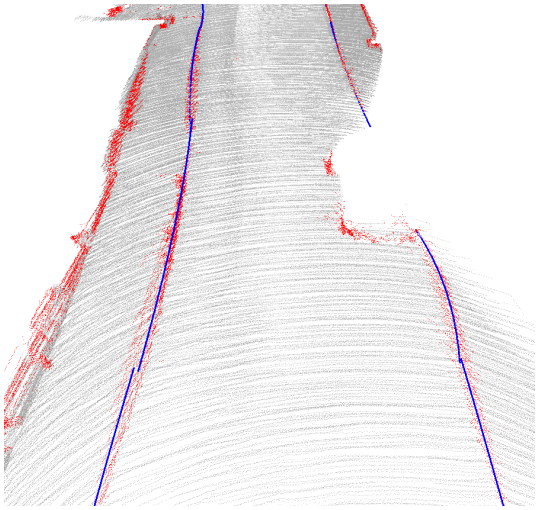
Figure 4.8: A close up of a white arrow on the road. The left picture shows the raw data. The right picture shows the found lane marking points in cyan.



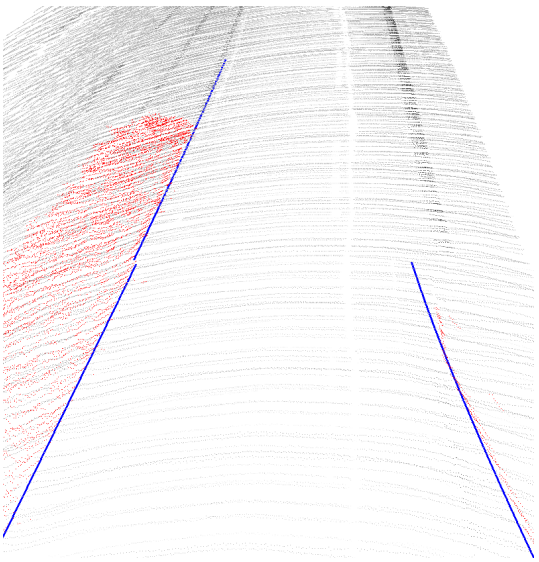
(a) Only the right curb was present and found. It is notable that the line of parked cars to the left do not trick the algorithm into thinking there is a curb there.



(b) Estimation of curbs on the right and left side found.

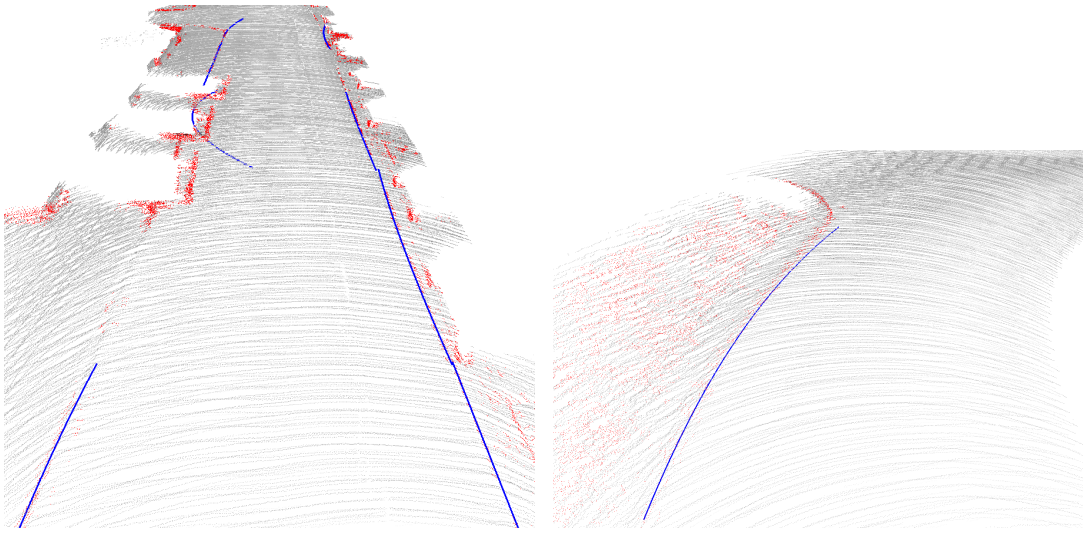


(c) Car parked in the middle of the road makings detection harder, but still possible in this case.



(d) The estimated lane on the left side is actually only a high spot of grass, but is the correct edge of the road, the curb on the right side is only present in the lower part of the picture.

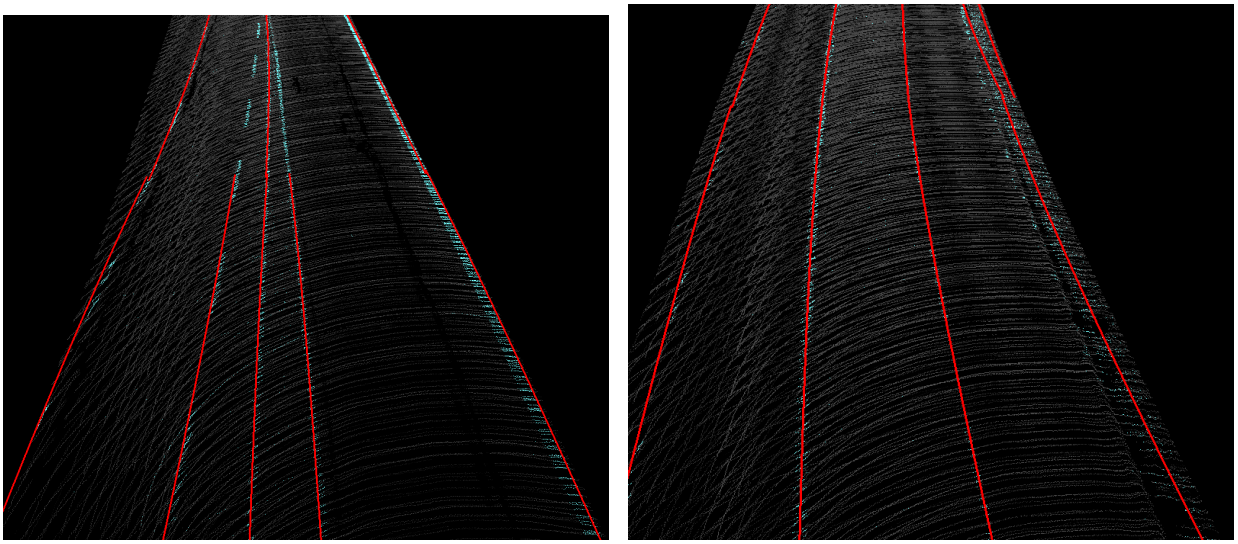
Figure 4.9: Estimation of left and right road edges. The red points are detected curbs and the blue lines are the second order polynomials that describe the estimated edges.



(a) An example where parked cars cause errors on the left side. Otherwise the edges of the road are found where curbs are present.

(b) One of the harder examples. There is no curb on the right side and on the left side there is a sharp curb. We are not able to model every part of the curb fully.

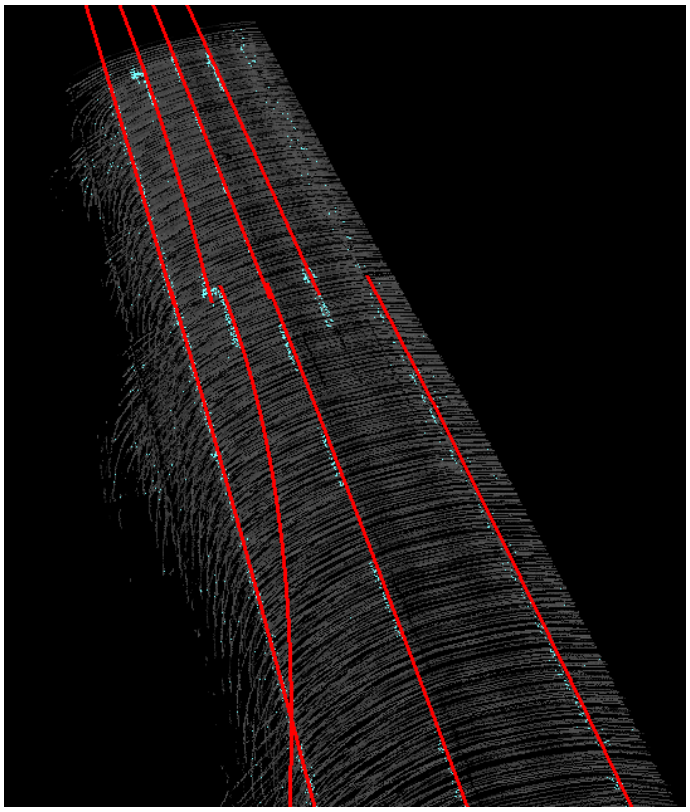
Figure 4.10: Estimation of left and right road edges. The red points are detected curbs and the blue lines are the second order polynomials that describe the estimated edges.



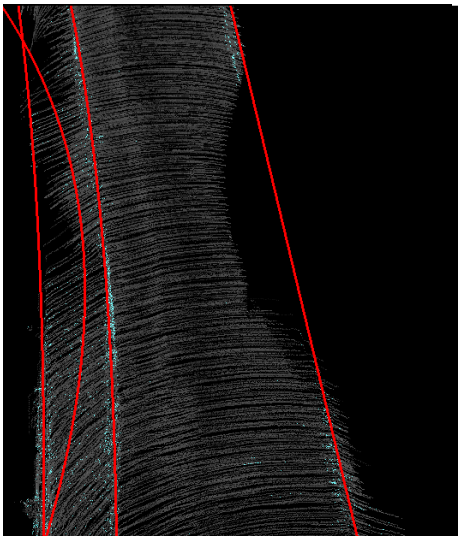
(a) The markings in this scenario only included lane markings, so the estimation is successful.

(b) The three lanes to the left were found, while the curb, stones and grass cause a lane to be found on the right side.

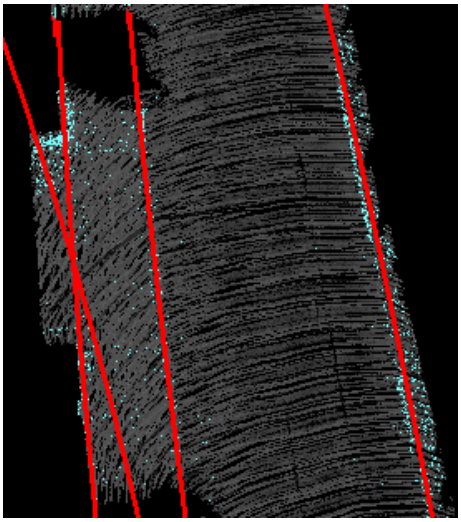
Figure 4.11: Estimation of lanes from road markings.



(a) An example of a scenario were road markings that are not lane markers cause false lane estimations.

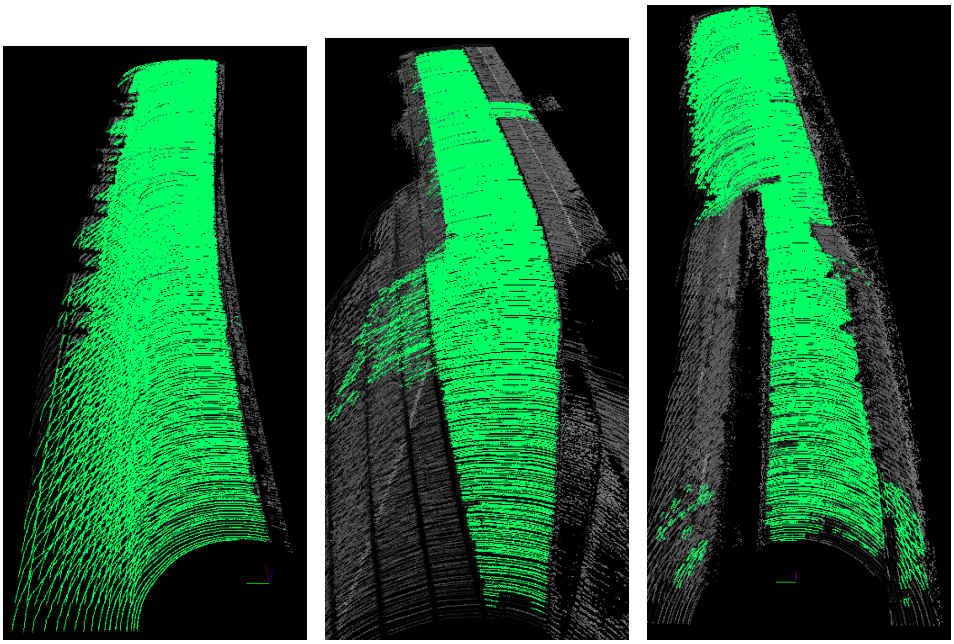


(b) A scenario were lane markings are not present. Errors due to high intensity points off the road cause false lane estimations.



(c) Similar problem as in picture 4.11b. No road markings are present and noise cause false lane estimations.

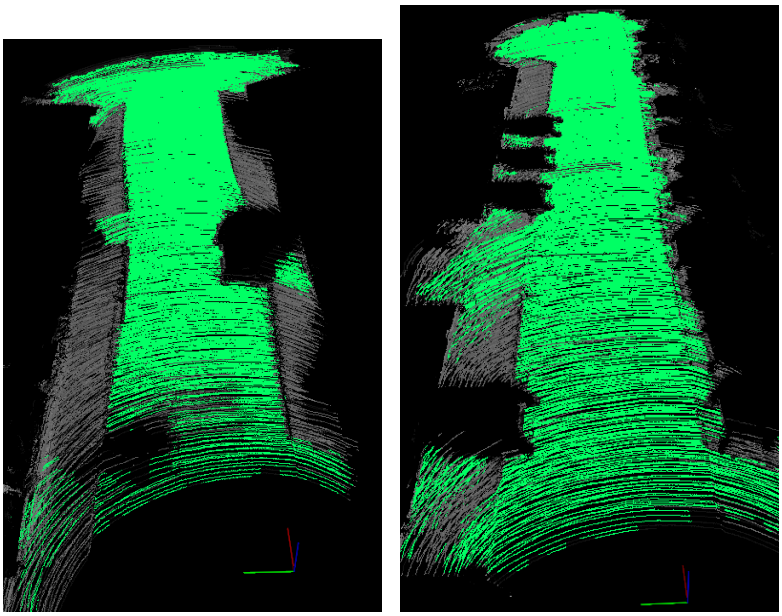
Figure 4.12: Estimation of lanes from road markings.



(a) The drivable area is found well, with an exception to a few blind spots under parked cars.

(b) Drivable area is found, but the railway is misinterpreted to be a curb.

(c) The general outline of the road was found.



(d) The drivable area was found in this case, but also some segments of false positives are present.

(e) This shows that the algorithm requires visible curbs in order to function properly

Figure 4.13: Estimation of the drivable area.

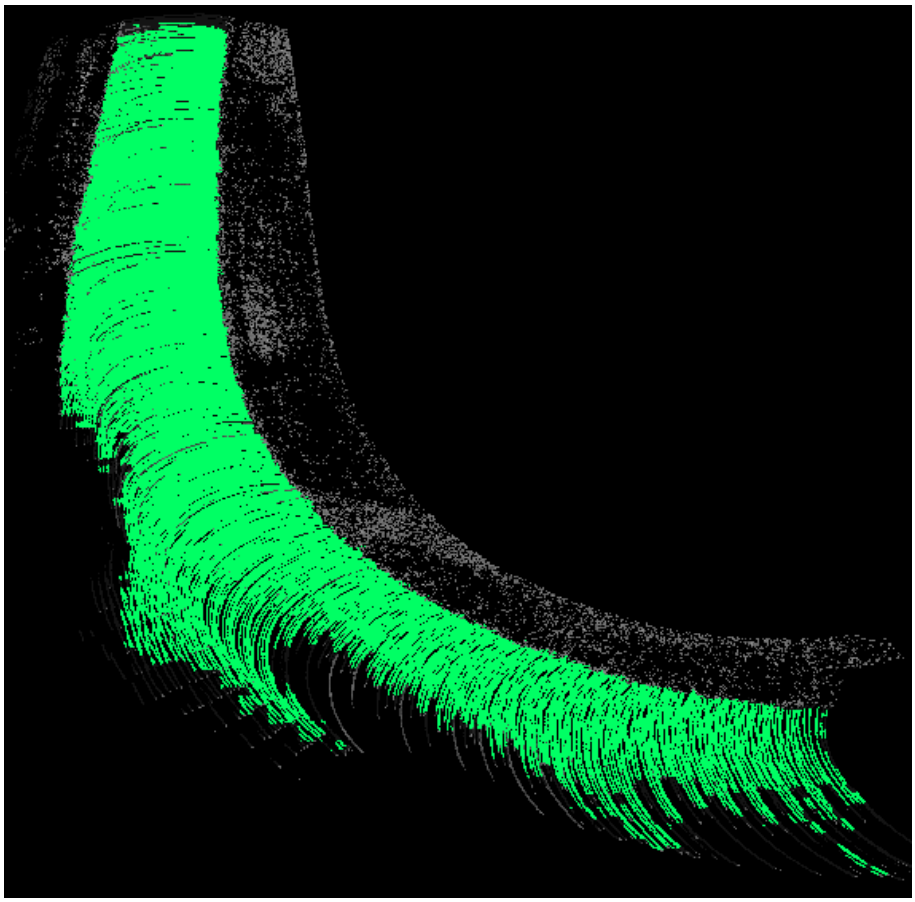


Figure 4.14: Estimation of the drivable area.The left segment of the picture is found well, but some sections of the road are missing in the first few frames, but it might be due to the resolution of points in the beginning being low compared to other parts of the scenario.

5

Discussion

Extracting road properties on offline data using a Velodyne Lidar does not require any complicated programming or mathematics. The methods used are straightforward and intuitive. The challenge is to find a robust method that would work equally well for all different scenarios. The road marker detection process needs to handle cases where no lines are visible and it needs to be able to separate lane markings from other road markings. When detecting road edges you need to handle roads without edges where the end of the road is marked by the transition from asphalt to grass. As many special cases as possible should be handled in order for a road property detection method to be robust. Such are the challenges of road property detection.
The results presented in Chapter 4 show a large variation in quality. To some extent some methods are more carefully thought through. The curb and road marker detection methods used to generate the results shown in figures 4.1-4.7 have received the most attention and development effort, while other methods still require improvements in order to be used on general data.

5.1 Road marker detection

The results in the figures 4.1-4.7 show that it is possible to efficiently find road marking in a large variety of scenarios. Road markings are nearly always found but there are two known errors with the method, that are largely based on Algorithm 2 presented in section 3.3.2. The method uses a combination of a threshold on intensities that is set and one that depends on the median of a diode.

- Problems arise in some cases where the calculated median value does not represent an average low intensity point on the road.
- The constant intensity threshold used is still set high enough not include some road markings.

The first issue arises in sections where the side of the road represents a bigger part of the data than the actual road. The goal of the median, is to find the value of a low intensity reading on the road. The mean value takes all values in to consideration, which is why it is not as appropriate for this task. When more than half the points lie outside the road, there is a large chance that the median intensity will be too high to accurately represent a point on the road. Grass and gravel on the side of the road generate high intensity readings that are picked up by the road maker detection algorithm. Hence pre-processing the data to remove as much of non road points as possible is important. This thesis does not propose a good automatic way to do this, rather when combining multiple frames we consider points where $y \in [-7,7]$, where y is the axis along the ground plane perpendicular to the travel direction of the car. This is an easy way out and the values -7 and 7 are sometimes changed manually depending on problem. It would be preferable to have an automatic way of finding the parameters, or a whole new way to segment the data but segmentation of the road is not a focus of this project, rather results assume that the data has been pre-processed to include mostly road. In the right side of the second picture in figure 4.7 typical noise from off-road grass is shown.

The second issue with Algorithm 2 concerns the fact that low intensity diodes will sometimes have lane marker point with intensity values systematically under the value β presented in Section 3.3.2.This issue is the cause for nearly every missed classification. This is a selected choice since the threshold can not be lowered any more without introducing large amounts of noise. If a grid noise removal algorithm is used an alternative almost equally good threshold would be,

$$\tau(\tilde{x}_{Int}) = \begin{cases} 0.36 & \text{if } \tilde{x}_{Int} \in [0.1,1] \\ 0.2 & \text{if } \tilde{x}_{Int} \in [0,0.1) \end{cases}$$

where $\tilde{x}_{Int}$ is the median intensity of a diode.
It is important to note that lines that are far away in the result pictures seem to not be accurate. This is purely a fault in the visualizer. There are too many points in the picture, so the visualizer drops some of the points that are far away. The points are all still there is you zoom in on specific sections in the visualizer. The still non movable picture of road markings does not do the results justice. As shown the road arrow example in the previous section about 90% of the points are probably found an classified.

If we compare the results to that of [11], [11], [6] and [12] the offline availability really seems to generate much more dense good results. Usually the detection rate of road markings is not analyzed but rather how well lanes are estimated from the detections so no direct comparison in numbers can be made, but by visual comparison it is clear that merged frames of a Velodyne Lidar give excellent results with the method used in this paper.

5.2 Curb detection

The results of the curb detection are probably the most satisfactory. As shown in figures 4.1-4.7 it is a reliable algorithm that works nearly on every scenario. One of the few problems is that it might be too sensitive. It find even low curbs and if a 10 cm curb is present it works nearly perfectly.Another known problem is that the algorithm does not distinguish a rail road from a curb. Also the algorithm does not distinguish a rise from a drop in altitude. Preferably it should only consider cases where there is a sudden rise in altitude when comparing the the height of the road. This problem does generate interesting results though. For example a hole in the road will now be marked red, and other sudden road changes can be detected even though it was not intended during the design of the method.

As with the road marking detection, the detection rate of curb points is generally not discussed in literature. By visual comparison the results of this paper seem to reflect the results of [13]. The only big difference is that more curb points are found in this paper due to the increased density of the data, but the detection rate seems to be the same.

5.3 Road edge detection

The road edge detection is heavily dependent on the results of the Curb detection. Since curbs were found when present, the road edge estimation only works when curbs are present. When no curbs are present there is no reason why the road edge detection should even find the edge of the road. There are a few problems encountered that can be seen in some of the result pictures in the figures 4.9-4.10.

- When parked cars are in the way sometimes there is a blind spot where the curb is not found. This increases the difficulty of finding a good road edge.
- If there are many parked cars in a row, the wheels will form a path following the road.
- The current method does not handle sharp turns.

The first issue can be seen in figure 4.10a where the estimated polynomial finds the best solution to include some curb points and some points on the wheels of a parked car. Similar problems arise when many parked cars are alongside the road. The most obvious solution to the problem would be to use some pre-segmentation to remove all objects from the data. By the use of, for example, an euclidean voxel grid, parked cars could be removed.

The third issue can be resolved by improving the method to use differently long polynomials depending on the curvature of the curb. To get a smooth edge of the road even for strange locations sharp edges should be modeled by polynomials of maybe 1-2 meters in length, while straight roads can be modeled by polynomials of the third order with

length of around 60 meters or as in the current implementation, with polynomials of the second order that are all 10 meters.

Once more the results seem to suffer about the same problems and advantages as in [13]. In their paper the road edge estimation worked when there were 10 cm curbs present on both sides and found by the scanner. The detection found faulty points if there were objects in the middle of the road, but generally showed satisfactory performance. The same is true for the results in this paper and that is why I consider the results about equal.

5.4 Lane estimation

In easy scenarios with only visible lanes the estimation of lanes worked well. Such examples can be seen in figure 4.11. The detection method is however challenged in specific complicated cases, such as:

- Sections of grass and gravel are found as road markings.
- There are many road markings that are not lane markings.
- We do not know how many lanes there are supposed to be in a scenario.

Most of the issues are covered in one of the results pictures in figure 4.12. The first problem with the high intensity points that are not on the road is that since the RanSaC algorithm tries to fit in as many points as possible, these sections are always prioritized. One improvement that would fix most of the problems could be to first run the road edge detection, and if edges are found we only look for lanes within these boundaries. This would solve many problems, for example the mishaps shown in figure 4.12 b and c. It would still require that curbs are present, and in every case where the road edge detection is not working the errors would be directly transferred to the lane estimation. The second and the third issues are a bit more tricky. If we, in some way, found the width of the road, we could calculate how many lanes there are supposed to be and use the width between them to only consider the points that could be apart of the lane. In figure 4.12 a we see how arrows are sometimes mistaken for lane markings, but in this case the edges of the road were wound. From the road edges we could calculate the width of the road and find that there are only 2 lanes. With this information only points near the middle of the road could be lane markings and the arrows would not cause false lane estimations.
Another way to discredit non lane markings could be to use the first lane estimation in order to find new ones. That is after a lane is found we look one lane width to the left and the right to see if there are other parallel lanes. The obvious setback with this method is if the road markings that are in fact not the road lanes cause the first lane estimation.

The results produced by this paper can be compared to that of [12] and [11]. In their applications they find lanes on easy to spot highways with well marked lines going straight. Usually only the host lane was found efficiently in these papers. In our case the enhanced amount of points allow the detection work well on the host lane and other lanes. In most applications on a high way, sometimes the lanes of the cars going in the other direction on the other side of the road are found. The method needs to be improved in order to work efficiently in urban scenarios but the fact remains that the increased amount of data really helps in this application.

5.5 Road extraction

The estimation of the drivable area was not the focus of this project, so the aim is to demonstrate the feasibility of the method rather than obtaining high accuracy results. In order to do a much better estimation of what points belong to the drivable area of the road a combination of a good road edge detection and a lane detection should be used. The outermost lanes and curbs should then show which points are drivable.
The current method finds all points were there is not much altitude change. If there is continuous big altitude change we stop. If well defined curbs are present this works alright, but the biggest problem shown in all figure 4.13 b-e, is that if curbs are not 10 cm high not all diodes register any big altitude change.
In all scenarios but the first, shown in 4.13 a, there are too many points that are falsely classified as road points. That is why the method needs to be substantially improved if it is to be used. This can be compared to the results presented in [7] where the authors presented a false positive rate of 0.83% and finds road segments with a miss rate of 0.55%. The method presented here requires more work in order to match the accuracy of the winers of the Darpa Urban Challenge.

6

Conclusion

This report studies and presents methods for extracting road properties using a Velodyne HDL-64E Lidar. Instead of working with single frames of Velodyne data we use the fact that we have off-line data and unite about 100 frames at a time to improve the density of the data. The road marker detection method presented uses an improved thresholding algorithm, using the median densities of each diode to determine segments of highly intense points that are classified as road markings. The curb detection method presented considers the height difference between consecutive points in order to find curbs. A road edge detection is used to find points where a potential left and right curb start. With the road edge points we classify points in between two curbs as road points if there is not a large height change among consecutive points. By using the RanSaC algorithm on the road edge data we were able to estimated a line representation of the location of the left and right curbs, a similar algorithm was used and presented to find lane markers.
Results showed that the road marker detection and the curb detection methods functioned well and generated good results, apart from a few special cases. The road edge detection works in cases where a curb is present and when there are not too many parked cars in the way. The lane estimation works only when the road has been segmented well and where lane markings are the only road markings. The detection of the drivable area is implemented but has a limited reliability. Further studies and possibly combination with other methods are required to improve it.
To conclude, promising results have been obtained, with a lot of room for improvement. There are many improvements that could be made. The data used to generate the results of this project were gathered at the KITTI Benchmark suite. For example, some improved methods would require an even better source of data with a time stamp and a diode number for each point.

6.1 Future work

During the work many opportunities an extension of the work were discovered. Below is a list of topics that could be taken in the future to improve the algorithms developed here.

- An estimation of how deteriorated road markings are.
- The computation of a parameter, or a set of parameters that describe road properties such as roughness of the road.
- A pre segmentation of the data to find road and sidewalk areas.
- Improve the existing results using the diode id of each point.
- A thorough investigation of how to calibrate the intensity errors in the diodes of a Velodyne scanner.
- Statistical analysis of road bumps.

Since we work with off-line data of combined frames, we get an incredibly dense point cloud and in fact from visual inspection of such point clouds there is a noticeable difference between the amount and spread of road markings detected on a new lane marker and an old deteriorated one. This gives rise to the question, could one use united frames of Velodyne data to estimate how well defined road markings are in reality. A classifier could be used to estimate what kind of a road marking a certain segment is and with that information a value for deterioration could for example be the ratio of points found to be intense to the amount of points present in a perfect road marking.
Another question of interest to car developers would be to find a way for the car to know what kind of ground the road is to be able to adjust other systems accordingly. That is why a classification of road roughness could be of value. One such value could be to look at the difference between the average height difference between two points on a flat road with respect to a rough road.
Another improvement that would enhance the methods presented in this paper could be to develop an effective pre-segmentation to separate the road and it's edges from surrounding terrain. A segmentation of the ground plane has been presented by my colleague Johan Villyson, and it finds the ground plane automatically and efficiently, what is needed is to also segment out parts of the ground plane that do not belong to the road.
In Section 3.4 some problems with the road edge detection were discussed. One easy, but grand improvement would be to use point clouds with more information in each point. If the points included diode id there are many small improvements to the existing methods.
As discussed in Section 3.1 there exists an error in the intensity readings of the Velodyne data that causes intensities to be differently strong in different diodes. If this were not the case, the road marker detection method cloud be reduced to a simple threshold on

intensities. One way to perform a calibration of the diodes is to find exactly what the errors look like through analytical testing of intensity values from different diodes. Once a model for the error is found one could devise a scheme to calibrate the diodes by hand or design a more clever automatic calibration.
Finally another interesting topic for future work could be to use the data to find and classify road bumps and other road irregularities. Advanced cars could use this information to adjust the suspension system to braise for impact.

The Velodyne scanner is not suitable as a standard sensor in production cars due to its high price and large size, but the fact remains that the methods are general for any Lidar equipment. The resulting methods in this paper could be used to validate existing cheaper scanners in a car, to see if they detect road markings and curbs correctly. In the future maybe powerful Lidar scanners are commercialize-able and then adaptations of the methods used in this paper could be used to detect road prosperities in future smart and autonomous cars.

Bibliography

[1] Montemerlo M, Becker J, Bhat S, Dahlkamp H, Dolgov D, Ettinger S, et al. Junior: The stanford entry in the urban challenge. Journal of Field Robotics. 2008;25(9):569–597.

[2] Kammel S, Ziegler J, Pitzer B, Werling M, Gindele T, Jagzent D, et al. Team AnnieWAY's autonomous system for the 2007 DARPA Urban Challenge. Journal of Field Robotics. 2008;25(9):615–639.

[3] Urmson C, Bagnell JA, Baker CR, Hebert M, Kelly A, Rajkumar R, et al. Tartan racing: A multi-modal approach to the darpa urban challenge. 2007;.

[4] Bohren J, Foote T, Keller J, Kushleyev A, Lee D, Stewart A, et al. Little Ben: The Ben Franklin racing team's entry in the 2007 Darpa urban challenge. Journal of Field Robotics. 2008;25(9):598–614.

[5] Hillel AB, Lerner R, Levi D, Raz G. Recent progress in road and lane detection: a survey. Machine Vision and Applications. 2012;p. 1–19.

[6] Kammel S, Pitzer B. Lidar-based lane marker detection and mapping. In: Intelligent Vehicles Symposium, 2008 IEEE. IEEE; 2008. p. 1137–1142.

[7] Zhang W. LIDAR-based road and road-edge detection. In: Intelligent Vehicles Symposium (IV), 2010 IEEE. IEEE; 2010. p. 845–848.

[8] Ring J. The Laser in Astronomy. New Scientist Jun 20. 1963;p. 672–673.

[9] Cameron ES, Szumski RP, West JK. Lidar scanning system. Google Patents; 1991. US Patent 5,006,721.

[10] Atanacio-Jiménez G, González-Barbosa JJ, Hurtado-Ramos JB, Ornelas-Rodríguez FJ, Jiménez-Hernández H, García-Ramirez T, et al. Lidar velodyne hdl-64e calibration using pattern planes. International Journal of Advanced Robotic Systems. 2011;8(5):70–82.

[11] Dietmayer K, Kaempchen N, Fuerstenberg K, Kibbel J, Justus W, Schulz R. Roadway detection and lane detection using multilayer laserscanner. In: Advanced Microsystems for Automotive Applications 2005. Springer; 2005. p. 197–213.

[12] Lindner P, Richter E, Wanielik G, Takagi K, Isogai A. Multi-channel lidar processing for lane detection and estimation. In: Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on. IEEE; 2009. p. 1–6.

[13] Yao W, Deng Z, Zhou L. Road curb detection using 3D lidar and integral laser points for intelligent vehicles. In: Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS), 2012 Joint 6th International Conference on. IEEE; 2012. p. 100–105.

[14] Smadja L, Ninot J, Gavrilovic T. Road extraction and environment interpretation from Lidar sensors. IAPRS. 2010;38:281–286.

[15] Fischler MA, Bolles RC. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM. 1981;24(6):381–395.

[16] Geiger A, Lenz P, Urtasun R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: Conference on Computer Vision and Pattern Recognition (CVPR); 2012. .

[17] Geiger A, Lenz P, Stiller C, Urtasun R. Vision meets Robotics: The KITTI Dataset. International Journal of Robotics Research (IJRR). 2013;.

[18] Fritsch J, Kuehnl T, Geiger A. A New Performance Measure and Evaluation Benchmark for Road Detection Algorithms. In: International Conference on Intelligent Transportation Systems (ITSC); 2013. .

[19] Rusu RB, Cousins S. 3D is here: Point Cloud Library (PCL). In: IEEE International Conference on Robotics and Automation (ICRA). Shanghai, China; 2011. .