B.Sc.Eng. Thesis
Bachelor of Science in Engineering

Automation and Control
Department of Electrical Engineering

Roadrunners Autonomous

Navigation using linear regression

Asger Johan Kühl (s163933)
Pelle Schwartz (s147170)

Kongens Lyngby 2019

DTU

DTU Electrical Engineering
Department of Automation and Control
Technical University of Denmark

Elektrovej
Building 326
2800 Kongens Lyngby, Denmark
Phone +45 4525 3576
susand@electro.dtu.dk
www.aut.elektro.dtu.dk

Abstract

This thesis focuses on a simplistic approach to solving an autonomous path planning task for an autonomous concept car made for a fuel efficiency competition at the Shell Eco-Marathon. The car participated in the Autonomous Challenge at the Shell Eco-Marathon. The project uses already processed LiDAR data for navigation purposes. An existing algorithm for navigation is effective under ideal conditions, but is unable to navigate using only one side of the road. This thesis focuses on improving the autonomous navigation by implementing a solution that depends on only one side of the road to work. This solution is implemented through two primary tasks: determining if both sides of the road can be used to navigate along and navigate using only one side of the road in the cases where only one side is available. It is shown that the algorithm was able to successfully navigate in a simulation environment and on the application of previously recorded scan data, showing that it is possible to navigate using only one side of a road as reference.

Preface

This thesis was prepared at the department of Automation and Control at the Technical University of Denmark in fulfillment of the requirements for acquiring a bachelor's degree in electrical engineering.

Advisors:
Jens Christian Andersen
Ole Ravn

Asger Kühl
Pelle Schwartz

Asger Johan Kühl (s163933)
Pelle Schwartz (s147170)

Kongens Lyngby, July 9, 2019

Contents

Abbreviations

AUC	Autonomous Competition, held at SEM
GPS	Global Positioning system
LiDAR	Light Detection and Ranging, see glossary
OS	Operating System
ROS	Robot Operating System [5]
SEM	Shell Eco-Marathon
SLAM	Simultaneous Localization And Mapping

Glossary

- **Drive point** - a target XY-coordinate with an angle representing the direction which the car should be oriented when reaching target point.
- **Ecocar** - The concept car developed by DTU Roadrunners, primarily designed for a fuel economy challenge. Secondarily also designed for autonomous driving.
- **Global coordinates** - world coordinates, with origin at the starting point of the odometry.
- **LiDAR** - abbreviation for Light Detection and Ranging. This thesis refers to the Velodyne VLP-16 *"Puck"* mounted on the car. It uses LiDAR technology to yield a 360° scan around itself and in 16 angles vertically.
- **Obstacle point** - A point generated by the situational awareness of the car, which resembles the location of something with the height of the barrier. A group of these together resembles the barrier's location.
- **Odometry** - Odometry is the use of data from motion sensors to estimate change in position over time.
- **Publish** - ROS term for when one node (program) sends information to other nodes.

CHAPTER 1

Introduction

1.1 Background

The purpose of this bachelor thesis is to improve the autonomous navigation of a car created by the DTU Roadrunners team to compete in the Autonomous Challenge (AUC) at the Shell Eco-Marathon (SEM).

1.1.1 Shell Eco-Marathon

The 2018 Shell Eco-Marathon is an annual competition where students from universities around Europe compete to make the most fuel-efficient vehicle. As a new addition in 2018 Shell introduced another challenge at the Eco-Marathon, the autonomous competition, where the car used for the Urban Concept class must be used to complete different autonomous tasks. Urban Concept is a class with certain requirements for the vehicle to make it similar to a car used for public roads. Requirements include four wheels, two doors, minimum and maximum dimensions etc.

Figure 1.1: The Ecocar driving on the competition track at the AUC 2019.

1.1.2 DTU Roadrunners

The DTU Roadrunners is a student based organization who build, maintain and improve a car for competing in the Shell Eco-marathon. The car is referred to as the Ecocar. Figure 1.1 is a picture of the car. The authors of this bachelor are both members of this

team and the bachelor is written based on the DTU Ecocar, a picture of the team that participated is in Figure 1.3.

1.1.3 The Autonomous Challenge

The autonomous challenge consists of 4 different tasks. The Autonomous Challenge (AUC) 2019 was held at the Shell Eco-Marathon Challenger Event in Holland from the 22nd to the 25th of May. All challenges must be driven autonomously without any inputs from the driver. If any barriers or obstacles are touched, the attempt is failed. 25 points are earned for each completed challenge (1-3).

1. Autonomous driving. Driving on a stretch of road enclosed by barriers with two turns.

2. Gate challenge. Is held on a straight piece of road with 3 pairs of pillars in a slalom-fashion. All 3 gates must be passed through for completion.

3. Parking challenge. The car must identify a parking spot on the side of the road, steer towards it and come to a halt inside a marked square.

4. Business presentation: Up to 25 points can be awarded during a business presentation of the autonomous system.

1.1.4 The Autonomous System in the Ecocar

The base of the autonomous system is an Intel NUC, a 64-bit PC running Ubuntu 16.04. The autonomous sensor system in the car consists of a Velodyne VLP-16 LiDAR(Light Detection and Ranging) as the primary sensor input and a camera mounted at the top of the car. The speed of the car is determined using an optical encoder based near the wheels and a gyroscope, which are used together with the speed of the car to do odometry. A complete description of the autonomous system can be found in Si Høj's master's thesis [6].

1.1.5 Gazebo simulation software

A simulation of the car and the car's autonomous systems was developed using the Gazebo simulation software by a Roadrunners team member, Thomas Passer Jensen, and described in his report [3]. The simulation supports randomly generated tracks, simple air and rolling resistances. An image of the car on the simulated track can be seen on Figure 1.2. The simulation uses the same code framework as the car. This means that it simulates real data for the sensors, which can then be used for navigation by the same code made for the actual autonomous system. It is not currently able to simulate spectators or other objects outside the track. It is therefore great for prototyping new

solutions but most implementations need real world testing before a concept can be proven.

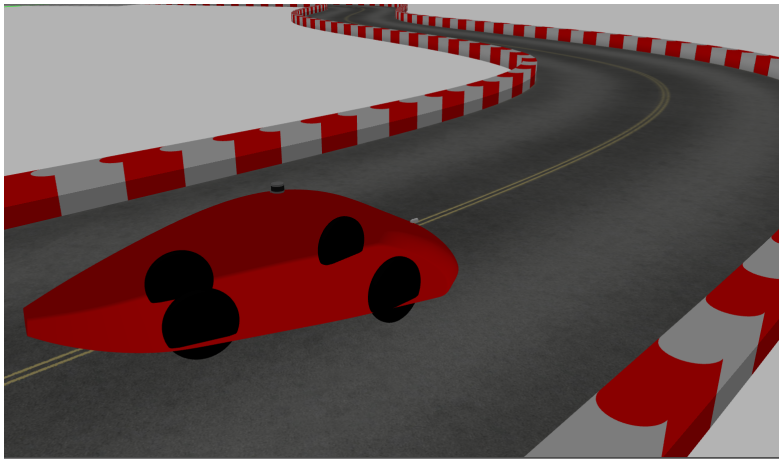


Figure 1.2: Screenshot of the Gazebo simulation software, showing the simulated Ecocar on a randomly generated track.

1.2 Goals for this project

The primary goal for this project is to make the cars navigation perform seamlessly at the AUC 2019. This is aimed to be achieved through the following objects:

- Detect whether a barrier to the left and/or right of the car is usable for path planning to determine whether to use the single barrier navigation or Voronoi-based navigation.
- Navigate using only one barrier.

1.2.1 Constraints

The following are design considerations due to this project being a part of the autonomous system already established through the work of other projects:

- The autonomous system is running on a PC with the Ubuntu OS, using the software framework ”Robot Operating System” (ROS). ROS requires that programs are coded using either C++ or Python.
- To reduce the scope of the project, existing solutions from last year’s competition for converting the raw sensor input from the LiDAR to obstacle points in a two-dimensional space are used.

- The hardware used for the project is limited to devices already mounted on the car: The LiDAR, camera, engine communication and steering and braking actuators.
- The competition is set on a race track enclosed by barriers with a height of approximately 40cm. The width of the track is at least 6 meters.
- Due to the competition being held before the conclusion of this project, most development was done before the competition and the tuning of the solutions was done afterwards with data collected from the competition.

Figure 1.3: Members of the DTU Roadrunners with the Ecocar, taken at the Shell Eco-Marathon Autonomous challenge 2019

1.3 Reading guide

Chapter description

- **Chapter 2 Problem analysis**
  - Analyzing and defining the problem for the thesis.
- **Chapter 3 Obstacle Point Classification**
  - How the barriers on either side are detected and classified as belonging to either left or right barrier.
- **Chapter 4 Single Barrier Navigation**
  - The implementation and test of navigation using one barrier.
- **Chapter 5 Results**
  - Results from testing the new navigation algorithm with the implementations from chapter 3 and 4.
- **Chapter 6 Future Work**
  - Plans and ideas for how to further evolve the designed algorithm as well as the autonomous Ecocar in general.
- **Chapter 7 Conclusion**
  - Summary of all conclusions made during the thesis.
- **Bibliography**
  - Bibliography with abbreviations for references used throughout the thesis.
- **Appendix A Figures used to calibrate the single barrier navigation**
  - Regression lines and drive points using single barrier navigation in various situations with different values.
- **Appendix B Data from AUC London 2018**
  - Figures showing the path planning of the implemented algorithm.
- **Appendix C Map of competition track**
  - Figures showing the competitions presented at the Shell Ecomarathon autonomous competition 2019.

CHAPTER 2

Problem analysis

2.1 State of the art

Autonomous navigation is a subject with many known solutions and techniques. Each solution depends on what kind of information it relies on, as well as how this information is interpreted and utilized. They have different levels of complexity and performance. Some of the more common solutions are highlighted in the next few sections. Each of them have strengths and weaknesses and are often used in combination to harness their various strengths. Table 2.1 sums up the pros and cons by the three types of navigation's strengths and weaknesses.

2.1.1 GPS based navigation

Using Global Positioning System (GPS) localization it is possible to determine a vehicle's location on a map. To navigate a land vehicle using only GPS requires the path to be mapped beforehand. The major advantages of this is its low device cost and ease of use. Bing-Fei Wu et al. has shown that GPS navigation is possible and reliable for an accurate following of a mapped path, but needs other kinds of real time scans to be able to react to changes on the road[9].

At the AUC 2019 competition, for an example the team from the Amsterdam University of Applied Sciences navigated using only a GPS receiver. Since the track had no moving obstacles the team could map the path using a human driver and follow this with great precision. The drawbacks of this technique is that it is not able to adapt to changes and that is is not able to drive on a track without prior knowledge of any obstacles.

2.1.2 Camera based navigation

An apparent strategy for autonomous navigation is the use of cameras as they yield the same information as a human driver uses. Camera based mapping was used by Vidal-calleja et al. in a project in 2008 for continuously improved mapping[8]. This was proven to be effective but computationally expensive, thus needing high computational power or low driving speeds.

2.1.3 LiDAR based navigation

The term LiDAR covers the use of laser scans of the surroundings. This can be used to find the distance between the LiDAR and everything around it. These scans can be made fast and are relatively quick to process compared to image processing. A high-speed 360° LiDAR, such as the Velodyne VLP-16 used by the autonomous system on the Ecocar scanner, is costly but it excels at obstacle detection.

At the SEM AUC 2019, all but one team used LiDAR based navigation, making this the most popular navigation tool at this competition.

	Pros	Cons
GPS	- Effective localization of car - Low price - Easy implementation - Does not work in tunnels etc.	- Requires a mapped path - Does not consider local obstacles - Requires accurate calibration
Camera	- Yields visual input - Can see colors of signs etc. - Low cost	- Computationally heavy - Sensitive to lighting conditions - Complex implementation
LiDAR	- Accurate local scans - Fast and mostly reliable - Easier to implement than camera - Computationally effective	- High device cost - Cannot see colors - Prone to noise e.g. rain, falling leaves etc.

Table 2.1: Pros and cons of using the three most common sensory inputs for autonomous driving.

2.2 Previous solution

When the car performed in 2018, the navigation solely relied on the information from the LiDAR [7]. The path planning system used an algorithm, based on the Voronoi algorithm described in Aktor's and Christensen's master's thesis [1]. What it does is to find the middle between all points, pick out the middle-points which has the farthest distance from one another then follow these points using a high-order degree fit to generate drive points along. While this solution yields accurate points in the middle of the road when barriers on each side are available, it does not work if only one barrier is visible. It will then aim for the middle of the barrier resulting in navigation straight towards that barrier. This happened for the Ecocar at the AUC in London 2018. The navigation was close to make the car hit the barrier but did get back to the middle of the track before hitting it.

The steering algorithm used in the car is named Line Drive, and was implemented through Aktor's and Christensen's master's thesis[1]. The Line Drive is based on the named technique mentioned in the article by Andersen et al. and tries to steer towards a two-dimensional point with an angle[2]. Therefore, to utilize the already implemented

steering algorithm, a drive point containing XY-coordinates and a direction must be generated for the steering algorithm.

2.3 Analysis of Goals

Based on experiences from SEM 2018, the main goal of this project is to ensure the car does not use the Voronoi-based path planning algorithm in cases where the information yielded by the obstacle detection is not fit for that algorithm and in these cases makes another path planning. Aktor and Christensen suggested either fusing the LiDAR data with data from a camera or creating a solution to navigate using only a single barrier[1].

Fusing data could be done by overlaying the camera data with the LiDAR data to obtain depth information for the camera picture, which would be very useful if the system used a computer vision based approach, to estimate the distance to any objects recognized using computer vision. Alternatively, if the system used object detection based on the LiDAR data, the camera could be used to increase the confidence of the detected objects. Since the car drives on a closed track with no humans or signage none of these solutions have been implemented.

To improve the current system by implementing a single barrier navigation, the system would need a decision-making module. This should determine whether the system should use the single barrier navigation or the Voronoi-based navigation.

This leads to the goals of this thesis which are further analyzed in the next two sections.

2.3.1 Obstacle Point Classification

The obstacle detection divides obstacles from flat ground in an area around the car. It uses data from the LiDAR scans and has proven to be mostly reliable, albeit occasionally not detecting the barrier on one side of the road. This can happen due to noise interfering with the LiDAR scans, in the form of people standing at the side of the road or small objects such as falling leaves or rain. Problems of this nature has been described by A. Kapp and was also experienced in the AUC 2018, where the car had difficulties at the start of the autonomous driving challenge[4].

The problem is that the Voronoi-based navigation algorithm does not work properly in situations where only one barrier is visible. This problem is sought to be solved by the obstacle point classification. It should detect whether barriers detected on either side of the car is usable and report whether the Voronoi-based algorithm can be used or not. If not, the best barrier should be used for single barrier navigation. Secondly it must also tell if the opposite barrier should be followed in cases where the Voronoi-based algorithm navigates too close to one barrier. The problem is described in depth in the implementation section of the obstacle point classification, Section 3.1.

2.3.2 Single barrier navigation

In the case where only a single barrier is visible or detected by the vision system of the car, a new algorithm for placing drive points is needed. The problem is to generate drive points using only one barrier consisting of XY-points. Two simple solutions are as follows: To keep a fixed distance to the barrier or to just turn away from the barrier, if the car gets too close. Both of these solutions are useful as a backup path planning, for controlling the car until the main system is able to detect both barriers and plan a more optimized path. These solutions are further elaborated in Section 3.1.

Due to the sensor setup, the car cannot drive too close to one barrier, as barriers located very close to the car are not detected by the LiDAR. The assumptions needed for the fixed distance solution to be viable is that the visible barrier always keeps the same distance to the middle of the road and that this distance is known prior to driving. Additional measures would be needed if the width of the track is unknown beforehand.

Another known problem is that the algorithm used to interpret the LiDAR data may group the barriers together with other obstacles located near them, such as human spectators or other objects nearby the track. To ensure the path is created based solely on the barrier and not other objects in the environment, filtering of the data received is needed.

Summing the problem up, the inner edge of a barrier must be extracted to get an exact development of the essential part of the barrier to be followed and the development of points of these points must be used to create a drive point's location and direction ahead of the car. The implementation of the obstacle point classification, single barrier navigation and a mission control is depicted on the flow chart in Figure 2.1.

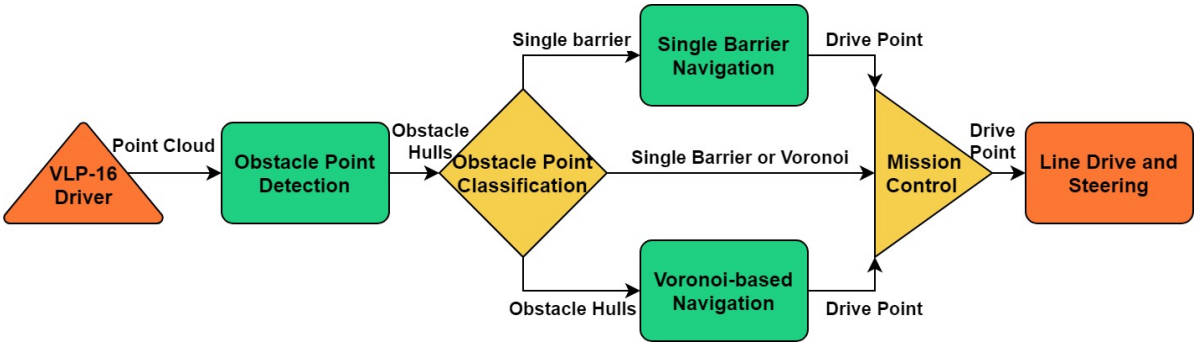


Figure 2.1: Flow chart of the process from a LiDAR point cloud to a drive point generated by either the Voronoi-based navigation or the single barrier. The obstacle point classification yields a single barrier for the single barrier navigation and all obstacle hulls are passed on to the Voronoi-based navigation. It also tells the mission control, which drive point to pass to the Line Drive steering algorithm.

CHAPTER 3

Obstacle Point Classification

The goal of this chapter is to describe how to determine whether a barrier on either the left or right side of the car is suitable for path planning. Note that the common coordinate system for robotics is used to describe local objects relative to the car. The origin is the center of the car and the x-axis increases in the direction the car faces and y-axis increases to the left of the car.

The existing obstacle detection algorithm in the autonomous system detect objects on the track and groups them with other nearby objects. Obstacles hulls and obstacle points are depicted in Figure 3.1. The obstacle point classification makes use of the grouping and tries to determine which obstacle group is the left and right barrier and whether these barriers are usable for navigation or not. Furthermore it determines if the car is too close to one of the barriers, in which case the Voronoi-based navigation has shown to fail and in this case it chooses an alternative navigational tactic.

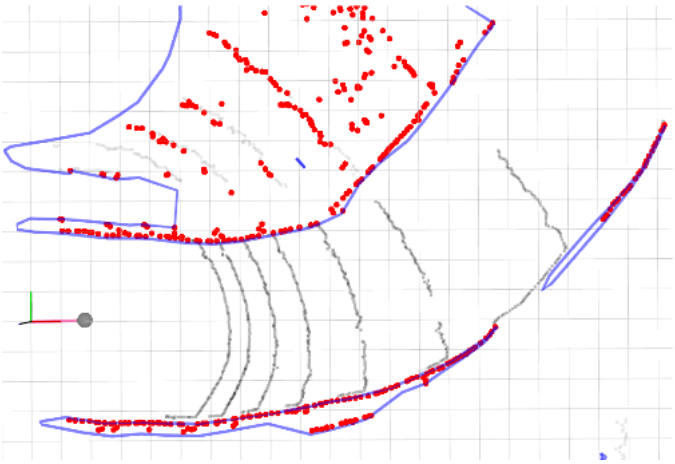


Figure 3.1: Obstacle points generated by obstacle point detection developed in Oliver's thesis on a 1x1m grid[7]. The points are grouped in obstacle hulls with corresponding IDs as displayed by the blue lines. In this example the points on the top left side would all have the same ID and then the line on the bottom would have another. A third group is grouped on the right

3.1 Implementation

3.1.1 Scan areas

Using the provided top-down two-dimensional coordinates delivered by the obstacle detection node, the system should consider if there are large enough barriers on each side to use in path finding. The apparent solution is to just find the closest point with a positive y-value and the closest with a negative y-value, however the car's orientation might be so it faces a barrier so the same barrier is on both sides of the x-axis.

This gives rise to the idea of having an area on either side of the car, which is scanned for obstacles. Figure 3.2 shows the design of the scan areas on both side of the car. At first the solution was to count all obstacle points inside the scan area and use this number as an identifier whether a barrier is usable or not. This caused problems when points beyond the barrier was counted as well, thus yielding a misclassification problem. This solution was used at the AUC 2019 with some success, but there were many cases where the classification was wrong. Afterwards it was apparent that another classification method was needed: to find the closest barrier ID inside the scan area and then look at the size of that barrier, to evaluate whether that is usable or not.

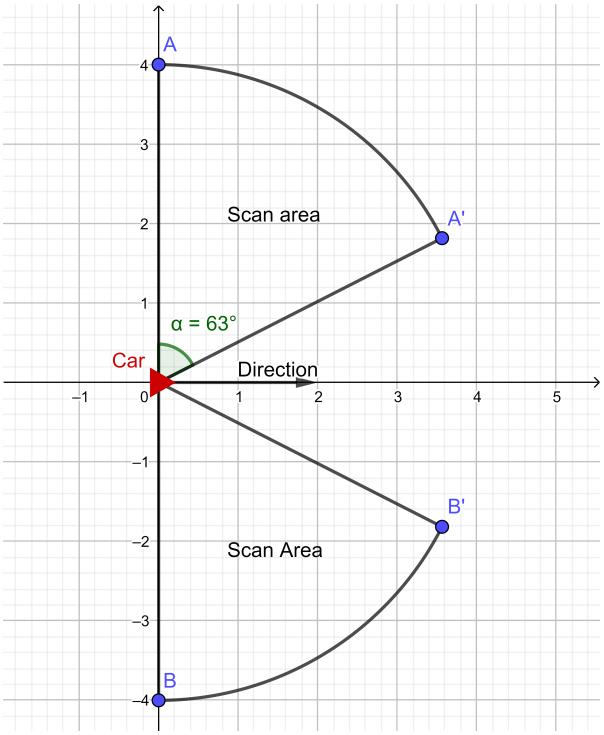


Figure 3.2: Graph displaying how the areas on either side of the car are scanned for obstacles. The x-axis increases in the facing direction of the car.

The algorithm finds the ID of the barriers closest to the car and within the defined areas on both side of the car Each area is is bounded by three lines. The local x-value

must be positive, that it is in front of the car. The local coordinate must not be farther than a certain distance from the car, to avoid classifying ”barriers” detected outside the track. This distance is calculated as euclidean distance. The final boundary is made with linear function with a slope to avoid detecting barriers right in front of the car.

Four values are returned by the barrier detection: the ID of the closest barrier point found on either side, as well as the distance from the car to those two points. These values can now be evaluated to determine and publish 4 different situations: either the Voronoi-based algorithm should be used, the car should just drive straight ahead or it should follow the left or right barrier. The decision tree for the algorithm is displayed in Figure 3.3.

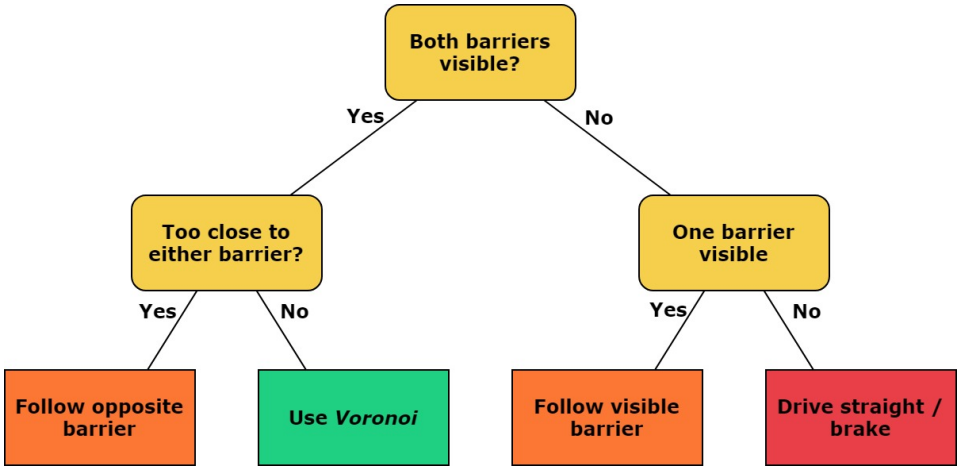


Figure 3.3: Decision tree for the actions of the obstacle point classification. Yellow boxes are decisions. Orange boxes represent the use of single barrier navigation and the green box is the Voronoi-based solution. The red box is in the case where no barriers viable for navigation are found at all.

3.1.2 Minimum distance to barrier

The obstacle point detection needs to be able to tell if it one barrier is too close to one barrier, which is one of the decisions mentioned in the decision tree in Figure 3.3. The ratio of the distances between the barriers on each side, φ, is calculated as shown in equation 3.1.

$$\varphi = \begin{cases} \dfrac{d_l}{d_r}, & d_r \neq 0 \\ \infty, & d_r = 0 \end{cases} \tag{3.1}$$

The value of φ can be used to determine on what side of the road the car is. It will converge to zero as the distance to the left barrier d_l gets smaller than d_r and converge to infinity in the opposite case. Whenever φ is lower than a fixed number φ_{min}, the car is too close to the left barrier and should follow the right barrier instead of using the Voronoi-based technique. And in cases where the car is too close to the right barrier,

$\varphi > \varphi_{max}$, the left barrier should be followed. Using this the correct decision will also be made when no barrier is found in the left or right area because the distance used will be set to zero.

The ratio is used rather than the difference in distance, $\varphi = d_l - d_r$ to make the obstacle point classification be adaptive to a road where the width would change. The values of φ_{max} and φ_{min} would not need to be tuned.

3.2 Testing the Obstacle Point Classification

Tuning of this function was done by changing four internal configuration parameters and comparing the results. This is documented in 3 different situations: two where it should choose single barrier navigation and one where it should choose the Voronoi-based approach.

The following parameters were adjusted through testing, to find suitable values for the desired classification:

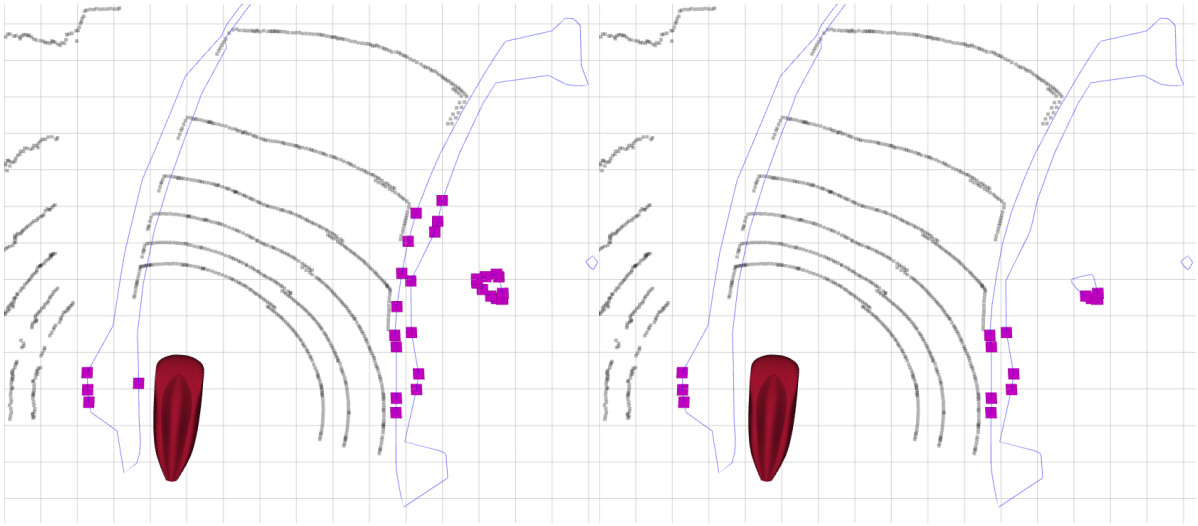
- `scan_angle`. Determined by the angle from the perpendicular line to the sloped line, shown as the line from the car to A' in Figure 3.2.
- `obst_max_dist`. The maximum distance an obstacle point can have from the car to be considered a barrier during the obstacle point classification.
- φ_{min} and φ_{max}. Limits for the relation between the distance to the closest barrier on the left and right side of the car.

The values for these were tested through several situations and three characteristic situations are highlighted in the next three sections. The values of φ_{min} and φ_{max} are reciprocal, $\varphi_{min} = \frac{1}{\varphi_{max}}$ and set to $\frac{1}{3}$ and 3. This interval means the car is more than three times closer to one barrier than the other when the value of φ is outside this interval. When the value of φ is outside the boundaries of φ_{min} and φ_{max} it means the car is too close to one barrier and should follow the opposite barrier instead of using the Voronoi-based algorithm even though both barriers may be classified as available.

3.2.1 Situation 1

The first situation is where the car is getting close to the left barrier. This is a situation where the system should classify the left barrier as unusable and classify the right barrier as usable for single barrier navigation. The reason for only using the right side is that the elevated LiDAR may not be able to correctly detect the barrier on the left side. Figure 3.4 shows how a misclassification can happen if `scan_angle` is too sharp. With `scan_angle` of 27°, the part of the left barrier actually closest to the car is ignored.

Since the calculated value of φ, shown in Table 3.1, is smaller than $\frac{1}{3}$, the algorithm chooses to follow the right barrier in this situation, which is as desired.



(a) `scan_angle = 45` (b) `scan_angle = 27`

Figure 3.4: Situation 1. Two visualizations on a 1x1m grid. of the same situation where the car is getting close to the left barrier. The car is facing upwards and slightly to the right. The gray dots are points scanned by the LiDAR. The blue outlines areas where obstacle points has been detected. `scan_angle` is different on the two pictures and the purple dots represent obstacle points found inside either scan area. Table 3.1 shows the distances to the closest point on either side.

d_l	d_r	φ
1.36	6.00	0.228

Table 3.1: Values of the distances in situation 1, as seen on Figure 3.4, to the closest point on the barrier on either side when using a scan angle of 63°. φ is the ratio between the distances d_l and d_r. The value is of φ is lower than

3.2.2 Situation 2

Situation 2, depicted in Figure 3.5, is set as an ideal situation, where the car is in the middle of a straight road heading straight ahead on the road. This situation has both barriers available and should classify both barriers as usable.

Results of situation 2 showed that `obst_max_dist` needed to be at least 4 m to make sure that it detects the barriers on both sides when the car is in the middle. The distances to each barrier results in a φ value within the limits of $\frac{1}{3}$ and 3 and the Voronoi-based algorithm's drive point is used. Changing `scan_angle` was not found to have an impact on the the classificaiton in this case.

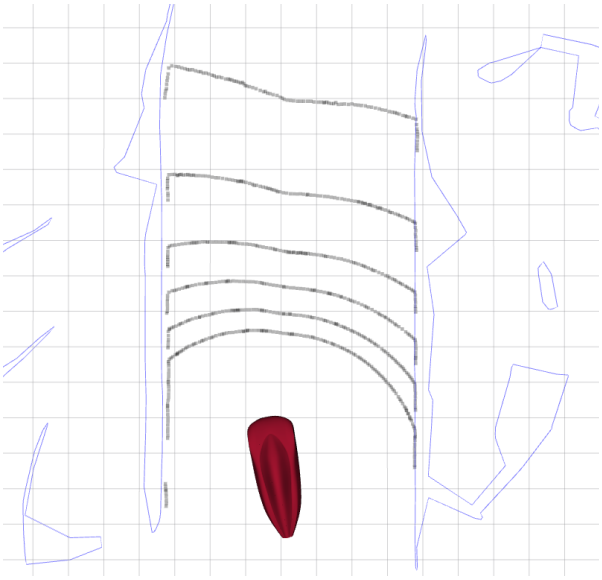


Figure 3.5: Situation 2, the ideal situation on a 1x1m grid. In this situation the car is in the middle of the road and facing upwards and slightly to the left. The gray dots are points scanned by the LiDAR. The blue outlines areas where obstacle points has been detected. Table 3.1 shows the distances to the closest point on either side.

d_l	d_r	φ
3,30	3,97	0,831

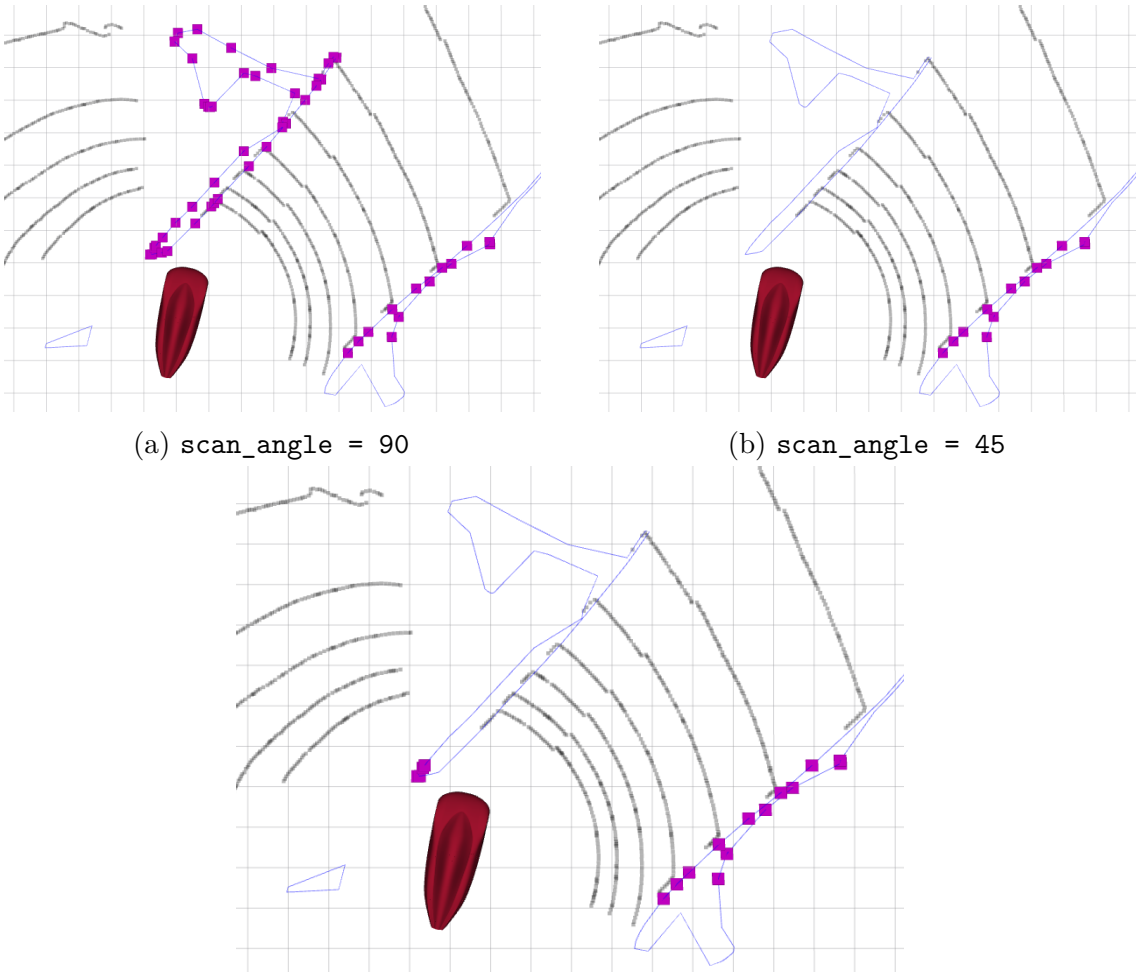
Table 3.2: Values of the distances in situation 2 to the closest point on the barrier on either side when using a scan angle of 63°. φ is the ratio between the distances d_l and d_r.

3.2.3 Situation 3

Situation 3, as seen in Figure 3.6a, shows the necessity of having a `scan_angle` smaller than 90°, since the car in this situation is headed towards the left barrier and the scan angle of 90 results in points belonging to the left barrier being in the scan area of the right side. This shows that having a too high a scan angle will lead to misclassification, as the same barrier can be detected as the closest one on the right and the left side.

3.3 Results

The best value of `obst_max_dist` was found to be 10 m. In all tested cases the barrier needed to be detected is within this distance. The 3 tested situations does not point towards an upper limit, but if barriers are further away than 10 meters it is likely not a barrier needed for navigation and rather an off-track object such as human spectators or



(a) `scan_angle = 90` (b) `scan_angle = 45`

(c) `scan_angle = 63`

Figure 3.6: Situation 3. A situation where the car is close to the left barrier and heading towards it. It is drawn on a 1x1m grid. The gray dots are points scannedby the LiDAR. The blue outlines areas where obstacle points has been detected. Table3.1shows the distances to the closest point on either side.. Note how the closest part of the left barrier is not detected by the LiDAR, due to its proximity to the car.

`scan_angle`	d_l	d_r	φ
90	2,39	3,60	0,665
63	2,39	5,20	0,460
45	0	5,20	0

Table 3.3: Situation 3. Distances to the left and right barrier as well as the ratio between them with 3 different scan angles. φ is the ratio between the distances d_l and d_r. The distance to the right barrier, d_r, changes between `scan_angle` 90 and 63 as the closest barrier point in the scan area belongs to the left barrier.

a building. The chosen `scan_angle` is 63° as this has shown to be effective at determining the side the barriers are located on. If the `scan_angle` is too small, a barrier might be overlooked and if it is too large, there is a high probability that points from the one side's barrier is counted in the other side's scan area.

Figure 3.7 shows that the classifier works as intended. As soon as the car gets too close to either side, it concludes that the opposite barrier should be followed instead. There is one point at the second blue part of the line, where it is close to the barrier but yet it concludes that both barriers are visible and usable. That very situation is the one investigated in situation 3, Section 3.2.3. This could be changed by making the to something like $\varphi_{min} = \frac{2}{5}$ and $\varphi_{max} = \frac{5}{2}$. This is not chosen to avoid switching between the two navigational algorithms too often if the road is narrow.

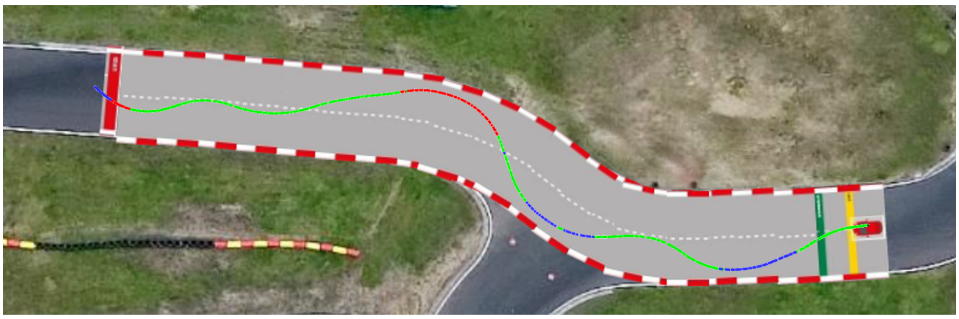


Figure 3.7: Plot of a drive-through of the autonomous challenge where the car gets close to the barrier multiple times. The color of the line indicates what the result of the obstacle classifier. Green means both barriers are detected, blue means follow right barrier and red means follow left barrier. The run started on the right side and finishes on the left side.

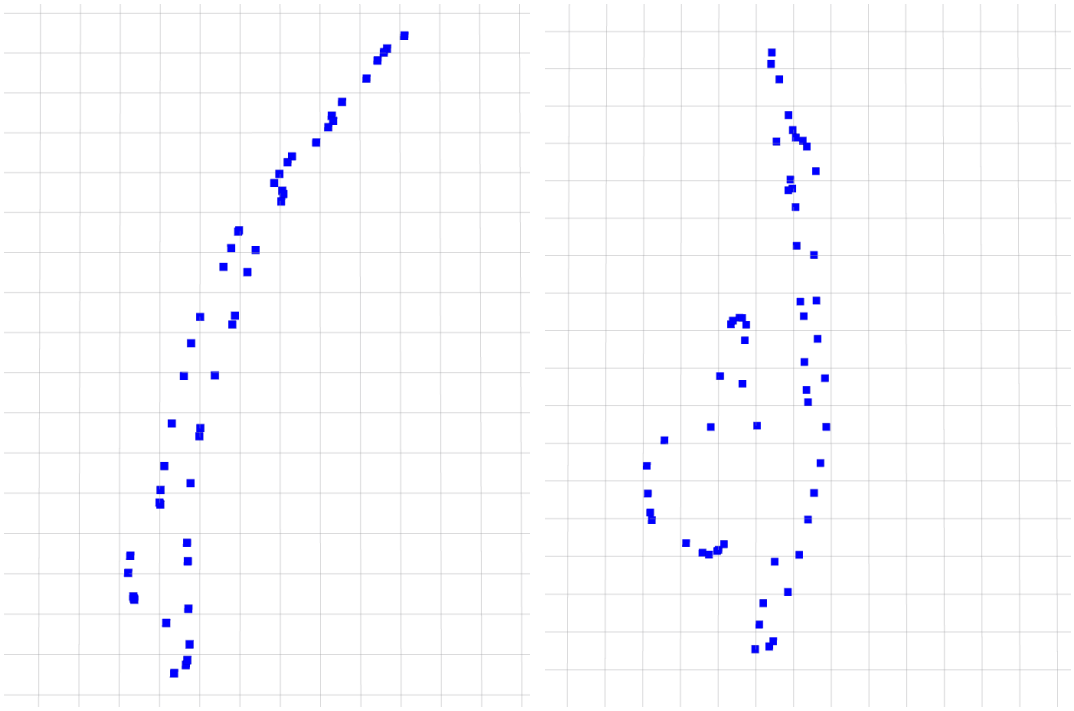
3.4 Chapter Summary

The obstacle point detection was tested on data from the test track and was successfully able to classify the usable barriers at the times it was too close to a barrier. It was also able to identify when a barrier was usable for path generation and when the car was in the middle of the road it correctly identified both barriers as usable. It did so by finding the closest obstacle points in an area on each side of the car and count the amount of points in this barrier, deeming it usable or not depending on this. The distance to the closest points on each side were successfully used to determine whether it was too close to one or the other barrier.

CHAPTER 4

Single Barrier Navigation

The previous Voronoi-based solution uses a complex algorithm. The solution sought in the single barrier navigation stands in contrast as it tries to follow the barrier in the simplest way possible. This yields a fast processing time, but may prove insufficient in certain cases.



(a) A barrier turning right. (b) A left barrier turning left with noise.

Figure 4.1: Visualization of two barriers detected by the obstacle point detection shown on a 1x1m grid.

4.1 Design

In the problem analysis part, Section 2.3.2, the main issues with using a single barrier for navigation were outlined and the assumption that the barriers bordering the road always keeps the same known distance to the middle of the road was stated. Figure 4.1

shows two single barrier detected by the obstacle point detection. The problem is to use these points .

Note, a detected barrier consists of several discrete points, as can be seen on Figure 4.1a.

For keeping a fixed distance to the barrier, several algorithms were considered:

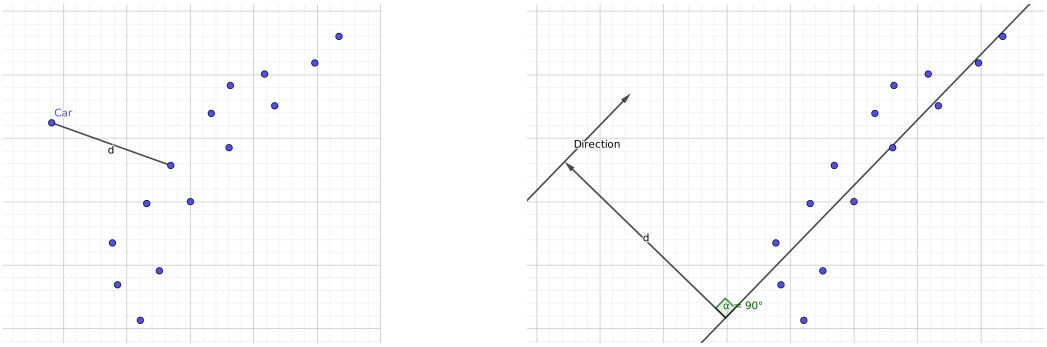
- **Distance to closest point**: Look at the nearest barrier point directly to the left or right, if it is too close, turn away. If it is too far away, turn towards it.
- **Distance to regression**: Generalize the barrier using polynomial fit of all the barrier points, move this the fixed distance towards the middle of the road and follow it.
- **Polyline fit**: Recreate the barrier as line segments and move these the fixed distance towards the middle of the road, then follow this new line.

It is also noted that the current steering system in the car does not control the steering wheel directly, but instead the navigation algorithms assign a drive point, which the car drives towards in a manner similar to the one described in Section 2.4 in the article by Christian Andersen et al. [2]. The drive point consists of a desired location and orientation of the car and the steering software tries to position the car at the point with the desired orientation.

4.1.1 Distance to closest point

An illustration of this algorithm is shown on Figure 4.2a. This algorithm is easily modeled as a simple feedback loop, where the input is the distance to the closest barrier point looking either to the right or the left and the output is the steering angle. This distance to the nearest barrier point is illustrated as the line label ”d” on Figure 4.2a. When driving on a road with continuous barriers this strategy is optimal, since the controller can be very finely calibrated. The first obstacle is the way the car navigates, using drive points instead of directly controlling the steering wheel. This does not disqualify this strategy, but it adds a layer of complexity, since the drive point will have to be set so the car behaves according to intent or the steering algorithm should be changed.

There are some weaknesses in using the distance to the closest point to keep a certain distance to the barrier. The biggest weakness is that any gaps in the barrier is considered a corner and this strategy could try to drive the car around this perceived corner. This solution does not make use of the many points of barriers generated by the LiDAR scanner and could be feasible with only two one-dimensional LiDAR scanners.



(a) Distance to closest point solution. (b) Distance to regression solution.

Figure 4.2: Illustrations of the proposed methods for navigation using a single barrier. On Figure 4.2a the shortest distance to the barrier is measured and then used to decide to either turn towards or away from the barrier. On Figure 4.2b a fit is made and then displaced. The displaced line can then be used for navigation.

4.1.2 Distance to regression

Looking at the biggest weakness of the distance to closest point algorithm, a solution could be to do a polynomial fit of the visible barrier, to smooth out noise and to bridge any gaps in the barrier. By displacing the fit a fixed distance towards the middle of the road, the fit can be used to drive along, since the barriers are assumed to follow the road, resulting in a driving path following the road. This is illustrated with a 1st order regression on Figure 4.2b.

The major parameter to decide is what order of polynomial fit to use. A 0th order polynomial fit is disregarded, since the steering algorithm considers the final orientation of the car and would therefore not be useful while turning. Any order higher than the second would only be useful in case the algorithm has to generate a path that considers several turns of the road, before a new path is calculated. This means that only if the navigational algorithm was run at a low frequency it could be necessary to calculate a higher than 2nd order fit and because the algorithm runs at a frequency of 20Hz, it is only needed to consider one turn at a time.

The choice is then between a first and a second order fit. A concern when choosing a higher order fit is overfitting when few points are used and a concern for a first order fit is that it might simplify sharp corners to a degree that the car will hit the corner. Due to the softness of the corners in the competition track a first order linear fit was chosen. A map of the challenges can be found in Appendix C.

The algorithm will also have to output a single drive point for the steering algorithm to drive towards. This was done by choosing a point on the fitted line in front of the car with the same orientation as the line and then displacing it perpendicular to the fit. Since the line represents the barrier the drive point would be parallel to the barrier.

The major strength of this smoothing is the robustness in regards to missing obstacle points, either because they are physically not there, or because the LiDAR software did

not detect them. A secondary minor strength is the forward compatibility in case the navigation software is upgraded to use a path. Instead of the current system of a single point, although an upgrade to a second order linear fit might be considered in such a case.

The greatest weakness of this algorithm is in case very small amounts of data is used. In the case of the LiDAR software only detecting a small part of the barrier, very small amounts of points could be passed on and if they are very clustered, this algorithms estimate of the barriers direction could be up to 90° off. However, the obstacle point classification algorithm makes sure the barrier used has a big enough size for an effective fit, thus eliminating the greatest weakness.

4.1.3 Polyline fit

This option is to connect the detected barrier points using line segments, move these segments the fixed distance towards the middle of the road, and then try to drive along these line segments. This is illustrated on Figure 4.3, where only the leftmost line segments are moved.

The major strength is avoiding overfitting, since each fit is exact to the data. This is a great strength if the detected barrier performs several turns, since they are all represented in the output. However the strength of this algorithm lies in creating longer paths with several turns and is diminished since the current steering algorithm does not consider a path but only a single point.

The major weakness is shown in regards to how the points should be connected. Looking at Figure 4.1a, connecting each point the nearest unconnected point might create a zig-zag pattern which is regarded as a bad path to drive along. The travelling salesman problem has solutions that yields a polyline in which all points are connected to no more than two other points and no lines intersect, resulting in a polygon resembling the outline of the barrier. The outline of the barrier could then be moved the fixed distance towards the middle of the road and be used to determine the path to drive.

How the algorithm should choose a drive point to pass on to the current steering algorithm is the second part of the considerations. A simple solution would be to do an average of all the line segments positions and orientations and use this as the drive point, however the result of this is suspiciously close to ”distance to regression” solution, reducing this solution to a more complex way of achieving the same. Alternatively, as the proposed solution to the connection problem returns an area this could be used by a well described path finding algorithm, such as the potential field path finding algorithm. Choosing this solution will introduce some other weaknesses in cases where only a small area is detected, or where a barrier is detected with a jagged shape, as can be seen on Figure 4.1b, which might result in incorrect driving directions.

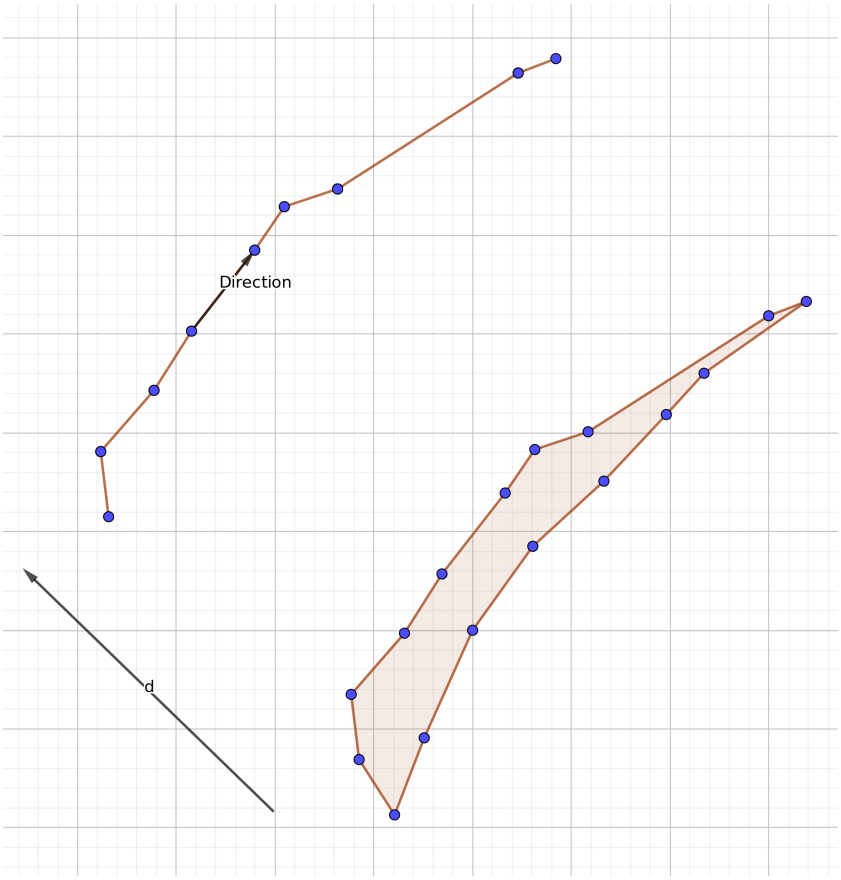


Figure 4.3: Illustration of the proposed polyline solution. On the right is a barrier bounded by a polyline. Part of the line is extracted and then displaced to be used for navigaiton.

4.1.4 Chosen Method

The chosen method was to keep a distance to a regression, since it is a strong upgrade from the simple distance to point, removing most weaknesses. The reason for not choosing the polyline solution is that in order to counteract the weaknesses of it, it will be very similar to the distance to regression solution, but done in a more complex way.

4.2 Implementation

In the implementations, considerations have been done to determine where the drive point should be placed using the fit, what estimation method should be used for the fit and how the noise filtering should be done.

The input for the function is a group of points, which the obstacle point classification determines to be big enough to do a satisfactory regression. An example of points generated by the obstacle point classification can be seen on Figure 4.1.

As shown, the barrier detection is sensitive to noise in the form of either people or off-track objects. The solution is to filter out the edge of the barrier closest to the car, as was illustrated on figure 4.3. Since tests have not shown any misclassifications inside the track and assuming the car is driving on the track the obstacle points closest to the car is where the actual barrier is located.

Following is a written example where the barrier to the right is determined by the obstacle classification used to describe the algorithm. Recall the common coordinate system for local coordinates in robotics is used here.

The algorithm finds the leftmost point in slices parallel with the y-axis. This is done by going through the array of barrier points detecting which point has the maximum y-value in each slice. The slice size should be big enough to avoid all of the obstacle point to be stored but small enough to keep an acceptable resolution. The size of the slices are determined in Section 4.3.1. Any points too far away from or too close to the car are also filtered at this step, since points too far away are deemed uninteresting with respect to what the local path planning this algorithm tries to achieve. Points too close are removed to make sure the fit represents a barrier the car has a chance to react to. The distances for these values are found in Section 4.3.2.

The filtered points are then used for a linear regression using the least squares method. This is chosen for the simplicity and because any outliers already have been removed. This gives a line that resembles the development and direction of the barrier. A parametric equation of this line will yield a vector that points in the direction of the barrier, which is used to determine the drive point.

The drive point is determined by finding a point along the linear regression line and then displace it perpendicularly to the line. Moving the point in this direction is done because it is assumed to be a good approximation of moving the point towards the middle of the road. The exact distance the chosen point is from the car have been chosen based on tests as described below.

A summary of all the parameters:

- **Slice size**: The size of the slices done across the x-asis, used to only pick the points observed from the inner part of the barrier.

- **Maximal distance to points**: The upper limit of how far away points can be from the car to be used in the regression.

- **Minimal distance to points**: The lower limit of how close points can be to the car to be used in the regression.

- **Displacement along the linear fit**: Where along the linear fit the drive point is localized, approximately also the distance in front of the car the drive point is placed.

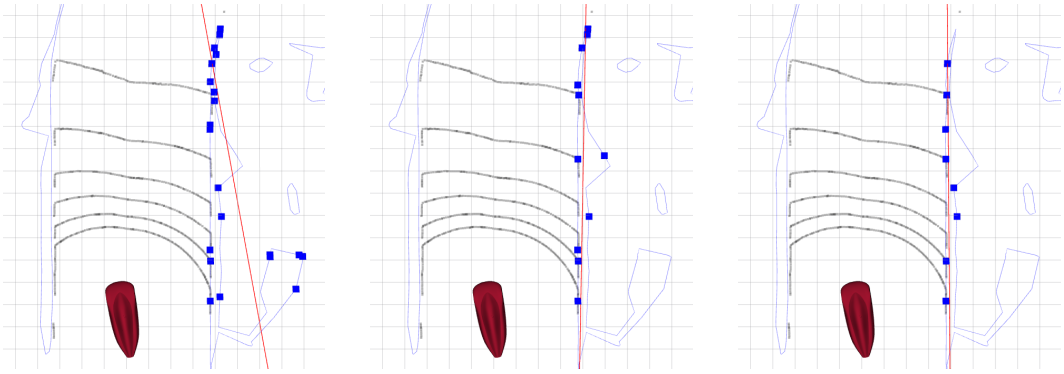
- **Displacement perpendicular to the linear fit**: The distance the drive point is displaced from the linear fit, perpendicular to the fit.

4.3 Testing

Test scenarios are used to find a satisfactory drive point in as many situations as possible by calibrating the decision parameters.

4.3.1 Slice size determination

The first variable to determine is used for filtering out the inner barrier points. This was done looking at a situation where additional points were grouped with the barrier, which could yield a less than ideal regression if all points were used. To find the ideal value, the right side of situation 2 from Section 3.2.2 is examined. This is ideal for a linear fit, as the barrier is arranged in a straight line and the obstacle detection has detected something outside the barrier to follow.



(a) `slice_size = 0.1` (b) `slice_size = 0.5` (c) `slice_size = 1.5`

Figure 4.4: Situation 2 used in Figure 3.5. Set with 3 different values of slice size. Single barrier navigation is set to follow the right barrier. The used points are marked in blue and the red line is the regression.

`slice_size`	R^2
0.1	0.0490
0.5	0.959
1.5	0.990

Table 4.1: R^2 value for the 3 different slice sizes used in Figure 4.4. The closer the value of R^2 is to 1, the closer the points are to the fit.

To make sure only the left side of the right barrier is extracted, as shown in Figure 4.4, it is clear the slice size should be more than 0.5 m. In this way only the leftmost points are used. This can also furthermore be shown by calculating the R^2 value as shown in Table 4.1. The value does get lower as there are fewer points and the details of the side are lost. However a slice size of 1.5 m yields a very high R^2 value in the

case of a straight line and yields a satisfactory outline of the development of the side of the barrier. The following Figures in this chapter all use a slice size of 1.5 m which all represents the side of the barrier satisfactory.

4.3.2 Distance to used points

As described earlier, detected points too close to the car or too far away are filtered. Points behind the car are also filtered since it is too late to react to them. The actual filtering distances were determined by looking at data collected from the car in a position where errors were expected, namely when a turn of the road was visible and including too many points could result in clipping the corner.

From data collected from the AUC 2019, a LiDAR scan received when near the sharpest corner was extracted and a linear regression from this scan was repeatedly generated with different filter parameters each time. The tested distance limits are between 0 m and 20 m.

Comparing Figure 4.5 and 4.6b based on position 1, it is seen that setting a minimum distance for the points does not improve the regression, but merely removes points that could be used to verify the resulting direction. As a result, minimum distance is is set to 0 m.

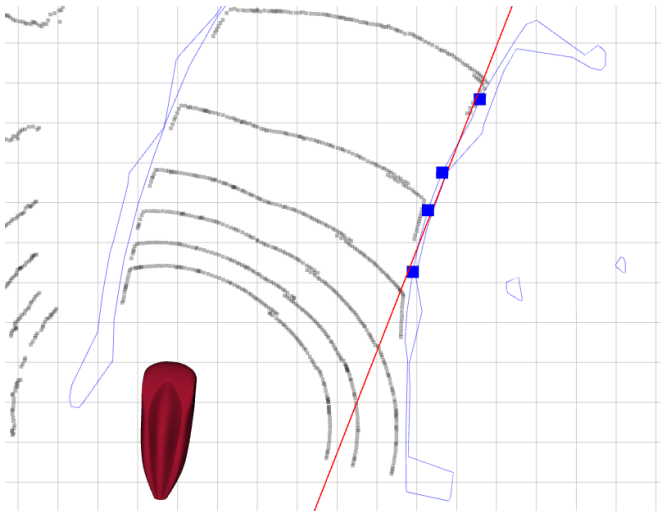
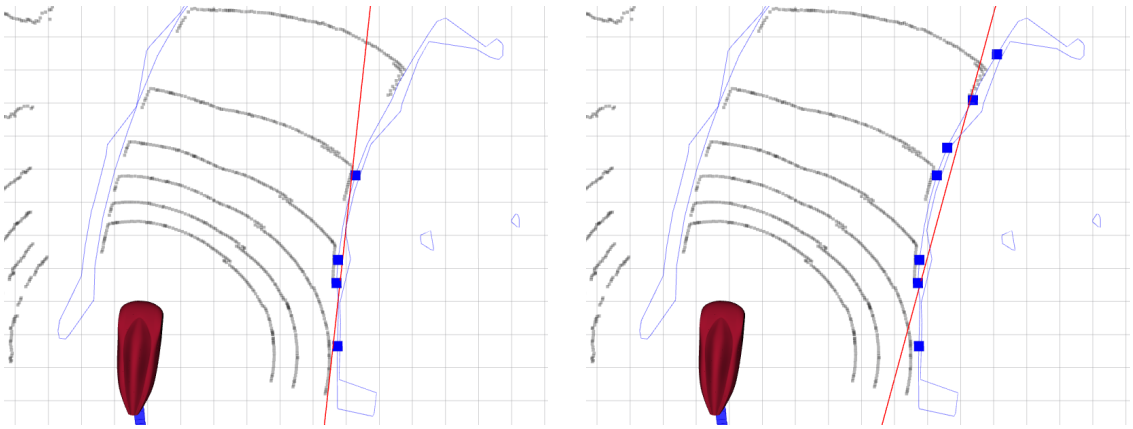


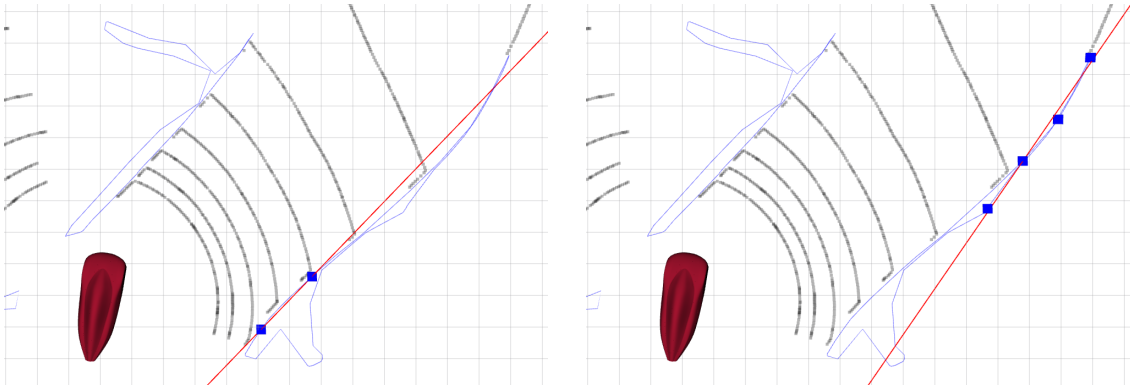
Figure 4.5: A representation of which points of the detected barrier are used to do a first order regression that later is to be used to generate a drive point for the car. The points are filtered to be between 7 m and 20 m from the car. The gray lines are the raw LiDAR scan, the blue lines are the estimated edge of the barrier, the blue points are used to do a regression and the resulting regression is shown in red.



(a) Points are between 0 m and 10 m from the car.

(b) Points are between 0 m and 20 m from the car.

Figure 4.6: A representation of which points of the detected barrier are used to do a first order regression that later is to be used to generate a drive point for the car. The points are filtered to be a certain distance from the car. The gray lines are the raw LiDAR scan, the blue lines are the estimated edge of the barrier, the blue points are used to do a regression and the resulting regression is shown in red.



(a) Points limited to be between 0 m and 10 m from the car.

(b) Points limited to be between 10 m and 20 m from the car.

Figure 4.7: A representation of which points of the detected barrier are used to do a first order regression that later is to be used to generate a drive point for the car. The gray lines are the raw LiDAR scan, the blue lines are the estimated edge of the barrier, the blue points are used to do a regression and the resulting regression is shown in red.

When deciding on a maximum distance, looking at position 1, setting a maximum distance of 10m, as seen on Figure 4.6a, the resulting regression is nearly parallel with the current orientation of the car, while a maximum distance of 20 m as seen on Figure 4.6b, shows an angle that would be more favorable for the car to have when entering this corner.

A second position where the car is turned more towards the barrier was also investigated. Changing the limits in this situation showed no major differences in the calculated regression, as can be seen on Figure 4.7. All regressions from the tests can be seen in Appendix A.

Based on this, the maximum distance is limited to 20 m as this generates satisfactory points in all tested situations.

4.3.3 Drive point displacement

The drive point is generated from the regression and displaced perpendicular to the regression line towards the middle of the road. The displacement was chosen to be half the width of the road, as driving in the middle of the road was considered optimal. The displacement of the drive point along the resulting regression determines how far in front of the car the drive point is localized. Placing the drive point too close to the car could result in the car driving past this point, before a new drive point could be calculated and therefore the maximum speed of the car and the computing time into must be taken into consideration.

The maximum speed the Ecocar can reach is $60\,\mathrm{km\,h^{-1}} = 16.6\,\mathrm{m\,s^{-1}}$. The run speed of the algorithm is much faster than this (less than 1ms) and the scan frequency of the LiDAR is 10 Hz. Being limited by the LiDAR scanning frequency, the car can drive up to 1.6 m before any new data is received and a new drive point can be calculated. Placing the drive point at least 2 m from the car will ensure the line drive steering always has a location in front of the car to drive towards.

Figure 4.8 shows the resulting drive point when either 2 m or 4 m in front of the car. Both seems to be valid options and based on this, the drive point was placed 4 m in front of the car to ensure the car would not drive past is.

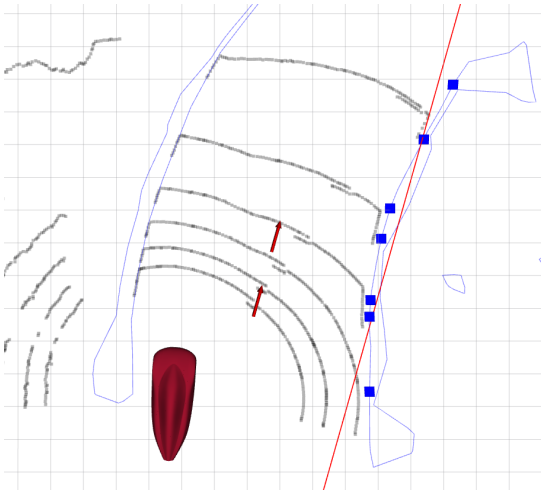


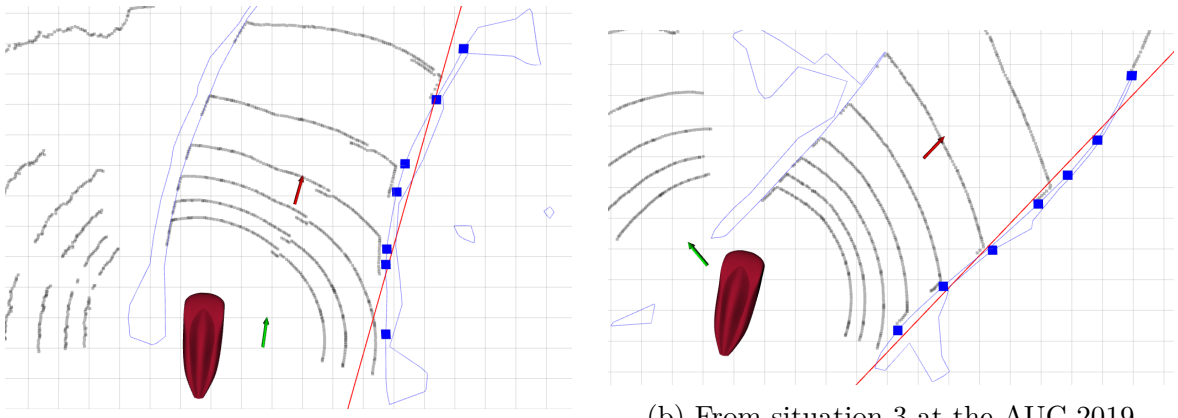
Figure 4.8: Two generated drive points placed $3.5m$ from the regression and either $2m$ or $4m$ from the car. The grey lines is the LiDAR scan, the blue lines are the detected barriers, the red line is the regression used to generate the two red drive points, represented as arrows.

4.4 Results

Figure 4.9a and 4.9b shows the drive points generated using the Voronoi-based algorithm as well as by the single barrier navigation for comparison. The main difference between the drive points on Figure 4.9a is how far away from the car the drive point is located. The best drive point to use would be the one generated by the single barrier algorithm, since the car would have time to turn, but the one generated by the Voronoi-based algorithm could also be used without causing any crashes.

On Figure 4.9b is an example of why the single barrier navigation is since the car is too close to the barrier for the LiDAR to detect it, it looks like there is a gap in the barrier. The Voronoi-based algorithm decided that this gap was big enough to fit the car through, which would result in the car steering towards the barrier making the drive point generated by the single barrier navigation the best drive point to use in this situation.

These examples shows that it is only in specific situations where the single barrier navigation is more useful than the Voronoi-based approach, while it in most cases is approximately the same.



(a) From situation 1 at the AUC 2019

(b) From situation 3 at the AUC 2019

Figure 4.9: Comparison of where the single barrier and Voronoi-based path finding algorithms would place the drive point, single barrier in red, Voronoi in green. The car represented as the red shape, the detected barriers are outlined with blue and the LiDAR scan is shown in grey.

4.5 Chapter Summary

Single barrier navigation was done by filtering the points observed to belong to the same barrier, taking only the innermost of them. Using the filtered points, this algorithm performs a first order linear regression using the least squares method and creates a drive point based on this 4 m in front of the car. It was shown to be an adequate navigational solution.

CHAPTER 5

Results

5.1 Simulation Results

After determining the parameters, the algorithm was tested in the Gazebo simulation designed by the DTU Roadrunners member Thomas Passer Jensen [3]. In the simulation, the single barrier algorithm was instructed to always generate drive points and the simulated car drove using these.

The simulated track can be seen on Figure 5.1 with the driven path displayed on top. The top corner of the simulated track is an example of a turn too sharp for the algorithm to plan a viable path which resulted in a crash. However corners as sharp as this is not encountered during any of the AUC challenges, but it shows that a different algorithm is needed in case a challenge with sharp turns is presented in the future. The simulation was stopped at this time, since this result shows the ability and the limits of the single barrier algorithm.

Simulation is mainly useful for determining if the concept is good and this has shown the algorithm could be used for navigation on its own. The simulations were effective to show how the single barrier navigation behaved in different situations, but the obstacle point classification was always under ideal situations. In a simulation no noise is encountered, so this is not the best way to show the usefulness of the filtering done, for this old data collected from AUC 2018 and 2019 was used.

5.2 Data Processing Result

The final algorithm was also applied on the sensor data acquired from the AUC 2018, where the car was directed towards the wall by the Voronoi-based navigation algorithm. Figure 5.2 shows the resulting drive points generated by both algorithms. The plot shows that the single barrier navigation would have helped in this situation. The obstacle point classification would firstly have detected that the left barrier was too close and then generated drive points on the right. After following these for a while, the middle of the road would be reached and the drive points generated by the Voronoi-based algorithm would be used again. More Figures of generated drive points in various situations can be found in Appendix B.

The fixed distance used as displacement is efficient as long as it is less than half the road width, but it places off-center drive points when the road width changes, as it did at AUC 2018. This could be solved by calculating the distance to the closest obstacle point

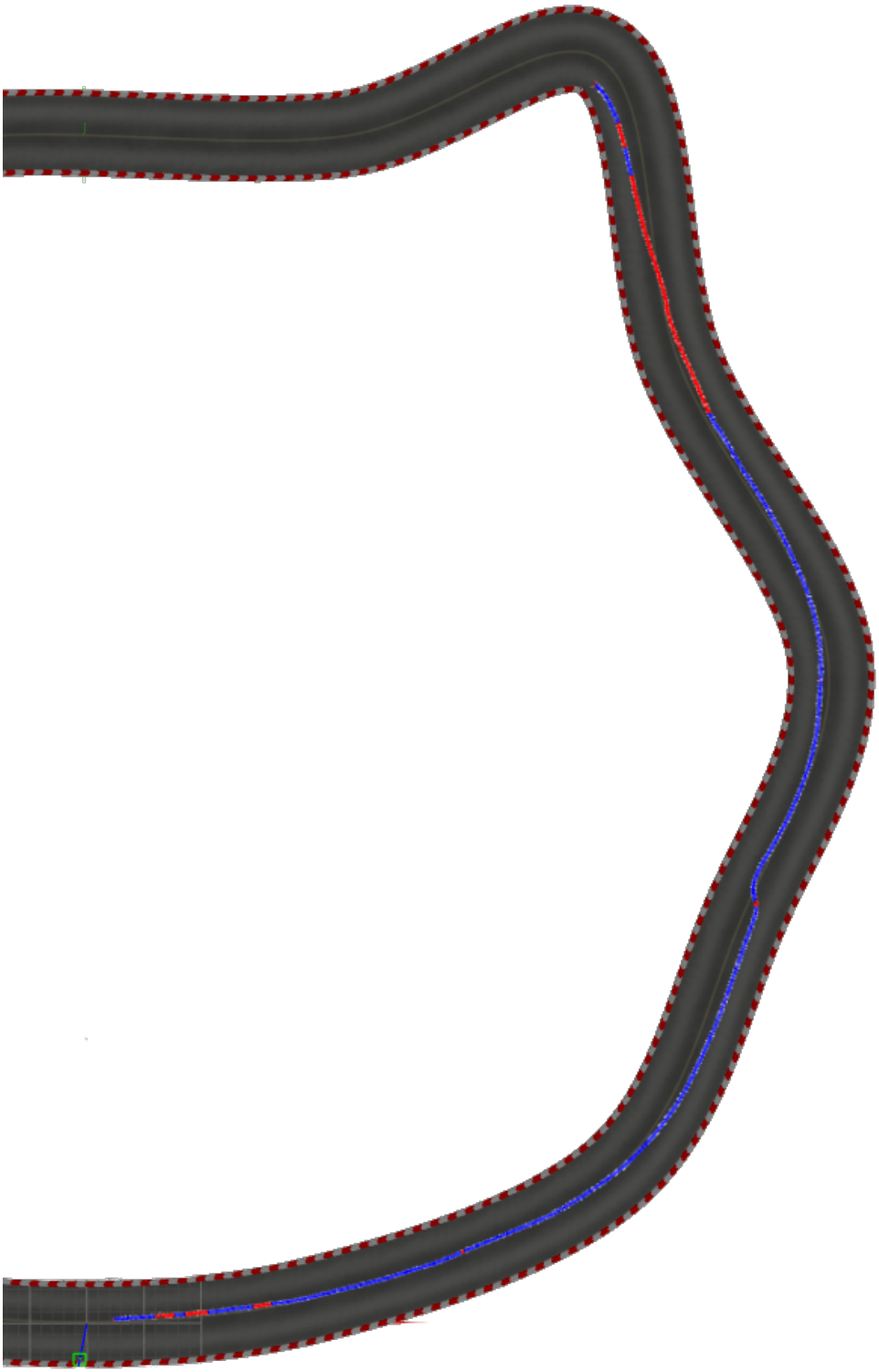


Figure 5.1: A top-down screenshot from the Gazebo software. It is displaying the track the final simulation was tested on and the path estimated by the odometry system is displayed on top.

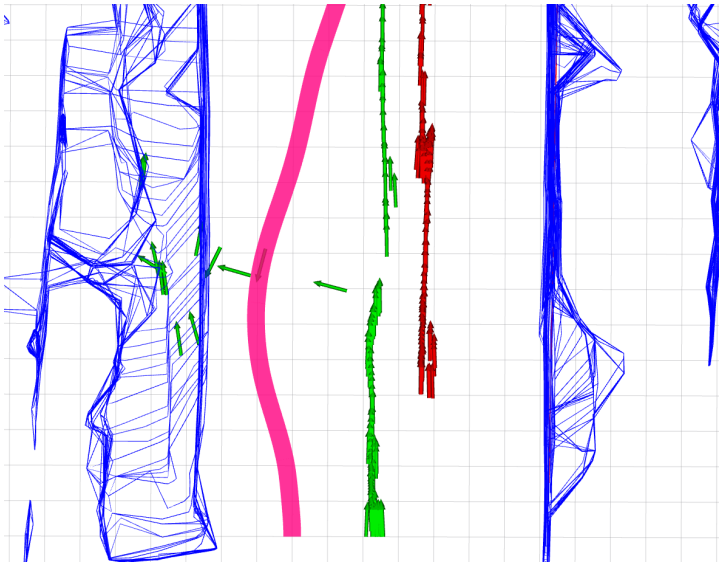


Figure 5.2: Plot of the first 15 meters of the track at AUC 2018 shown on a 1x1m grid. The blue lines represent detected obstacle hulls. The pink line is the car's trajectory. Arrows in green are drive points generated by the Voronoi-based solution and the single barrier navigation's drive points are shown as red arrows.

and then use this to determine the optimal distance. These values will not represent the exact width, but a running average of the last 5 measurements could be used to get a satisfying determination of the width of the road ahead of the car.

The point's placement in Figure 5.3 are all satisfactory drive points, however they do tend to sometimes flicker. Calculating a weighted running average on the last 5 or 10 drive points local coordinates and using this as the given drive point could yield a more stable path, eliminate outliers, and in return make the steering more smooth.

Applying the algorithm to data collected at the AUC 2019 shows a collection of satisfactory drive points, as shown on Figure 5.4. The gap in the drive points is a result of generating them from the left barrier and then switching to generating them using the right barrier. This suggest that the placement of the drive point should be based on distance from the car instead of on the local x-value.

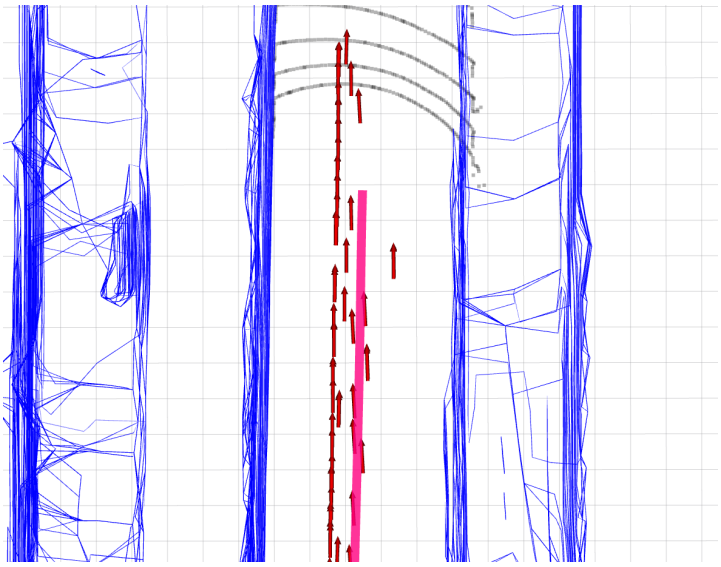


Figure 5.3: The car's trajectory in pink, outline of obstacles in blue and drive points generated by the single barrier navigation in red on a 1x1m grid. Made with data from the AUC 2018. Most of the drive points are generated from the right barrier.

Figure 5.4: Red drive points generated by the single barrier navigation on data from the AUC 2019. Purple is the car's trajectory and blue is the detected obstacles. The jump in drive points in the middle is caused by the change of single barrier navigation from following the left barrier to following the right barrier.

CHAPTER 6

Future Work

The following chapter presents suggestions for improving the navigation of the autonomous Ecocar.

6.1 Single barrier navigation as primary navigation

The single barrier navigation could be used as a primary navigation, completely replacing the previous navigational algorithm, as was shown in the results, Chapter 5. This requires the single barrier navigation to generate a more smooth path, and it might be feasible if the following 3 features is added to the navigational algorithm.

Higher order regression

The linear regression used when following a single barrier works well with the turns encountered in simulation and testing, however on sharper turns, a higher order regression might be more suitable. This could be done in cases where the R^2 value for the regression goes below a certain threshold.

Filtering and outlier detection for drive points

To make the single barrier navigation potentially usable as primary navigation algorithm a smoother line of drive points is needed. Filtering the drive points using a moving average filter and with an outlier detection will improve the performance. This will yield a smoother line of generated drive points and should eliminate sudden changes if a single scan generates an outlier as well as resulting in a smoother steering.

Road width detection

Right now, the width detection is set to a fixed distance. This is effective as long as the width of the road is relatively constant. However to avoid manually changing this parameter and to be able to drive on a road with a changing width, upgrading the

algorithm to displace according to the road's width would make the cars autonomous system more versatile.

6.2 Other improvements

Simultaneously Localization And Mapping

The use of simultaneously localization and mapping (SLAM) would increase the robustness of the system and could be used with more advanced navigational algorithms. Currently the system delivers a drive point based on a single scan. A SLAM consists of recognizing the detected obstacles and matching them with prior detected obstacles to ensure the detected obstacle is not noise, while remembering the location of all obstacles even in areas not currently being scanned. Using this style of mapping and navigation, the system's situational awareness will improve and if a challenge of driving multiple laps presents itself, the performance could increase for subsequent laps. The earlier implemented solution based on the Voronoi-based algorithm would also greatly benefit to have more accurate barriers yielded by an effective SLAM.

Camera-LiDAR sensor fusion

Right now the navigation depends solely on the laser scans yielded by the Velodyne VLP-16 scanner. This makes the system dependent on this working effectively all the time. This thesis focuses on handling times when the LiDAR obstacle detection makes errors. To further improve the robustness of the scans of the surroundings, the LiDAR scans and camera data could be combined. The LiDAR and camera is located close to each other so each pixel from the camera could be assigned a distance to make accurate parking detection or other navigation regarding stripes on the road, potholes etc.

CHAPTER 7

Conclusion

The Roadrunners' Ecocar was prepared for the Shell Eco-marathon autonomous competition 2019 by introducing a system for switching between two steering algorithms, based on how visible the barriers along the sides of the roads are. Originally, the system was implemented in 2018 and used barriers on both side of the road to navigate. However, this thesis presents a new approach for autonomous driving, wherein only a single side of the road was used to navigate.

The system for deciding which steering algorithm to use, the obstacle point classification, was able to determine the closest barrier on each side of the car. If the barrier contained more than 10 obstacle points it was deemed fit for navigation. The obstacle point classification satisfactorily determined which solution to use depending on the distance to and availability of barriers on either side. The algorithm performed reliably in simulations and when used on data collected during the Shell Eco-marathon autonomous competition 2018 and 2019.

The navigational algorithm designed, was able to reliably filter outliers detected by the sensors and was able to correctly steer the car in simulations when driving on tracks like those used at the Shell Eco-marathon autonomous competition. It was shown that this navigational algorithm is unable to follow the road in very tight turns of approximately 90° or more. The navigational algorithm designed kept the car the assigned distance from the side of the road and made sure the car had a target to drive towards by ensuring the calculation time was smaller than the time it took to drive to the current target.

Summing it up, the system was able to: Determine if a barrier on either side of the car was usable for path planning and plan and drive along a path created using only barriers on one side of the car.

Bibliography

[1] Lars David Aktor and Jesper Martin Christensen. "Sensor Based Navigation for Autonomous Car". An optional note. Master's thesis. 2800 Kgs. Lyngby: DTU - Technical University of Denmark, July 2018.

[2] Jens Christian Andersen, Ole Ravn, and Nils. A. Andersen. "Autonomous Rule-Based Robot Navigation in Orchards". In: *IFAC Proceedings Volumes* 43.16 (2010). 7th IFAC Symposium on Intelligent Autonomous Vehicles, pages 43–48. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20100906-3-IT-2019.00010. URL: http://www.sciencedirect.com/science/article/pii/S1474667016350303.

[3] Thomas Passer Jensen. *Pre-competition preparations for an autonomous Shell Eco-marathon car*. Technical report. Kongens Lyngby, 2018.

[4] A. Kapp. "Robust object segmentation and parametrization of 3D lidar data". In: *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.* June 2005, pages 694–699. DOI: 10.1109/IVS.2005.1505184.

[5] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA Workshop on Open Source Software*. 2009.

[6] Henning Si Høj. "Platform Integration for Autonomous Self-Driving Vehicles". An optional note. Master's thesis. 2800 Kgs. Lyngby: DTU - Technical University of Denmark, June 2017.

[7] Oliver Lyngaard Topp. "Situational Awareness for an Autonomous Vehicle". An optional note. Master's thesis. 2800 Kgs. Lyngby: DTU - Technical University of Denmark, July 2018.

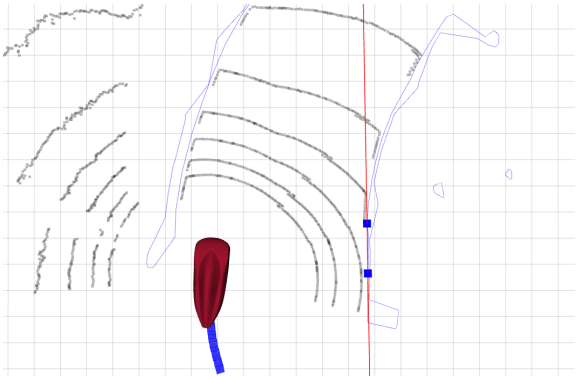
[8] Teresa Vidal-calleja, Alberto Sanfeliu, and Juan Andrade-cetto. "Autonomous Single Camera Exploration". eng. In: (2008). DOI: 10.1.1.103.6086.

[9] Bing-Fei Wu et al. "GPS navigation based autonomous driving system design for intelligent vehicles". eng. In: *Ieee International Conference on Systems, Man and Cybernetics, 2007* (2008), pages 3294–9, 3294–3299.

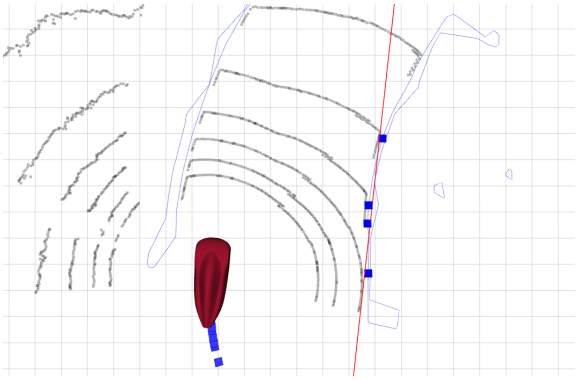
APPENDIX A

Figures used to calibrate the single barrier navigation

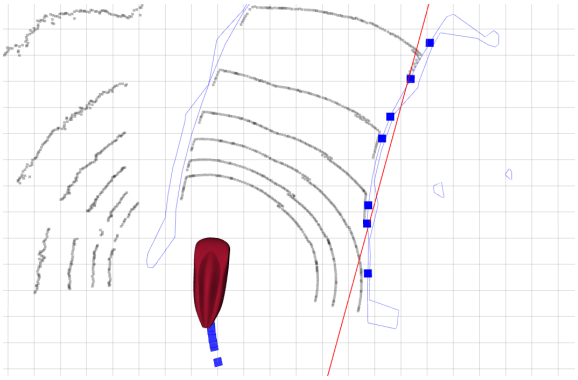
Following is a collection of visualizations showing the laser scans done by the LiDAR in grey, the outline of the detected barriers in blue, the points used to do the regression used in the single obstacle navigation in blue, and the resulting regression as a red line. The Figures show the result of varying the minimal and maximal distance the points need to have from the car and the impact on the regression line. This test was done with the car standing in two different positions.



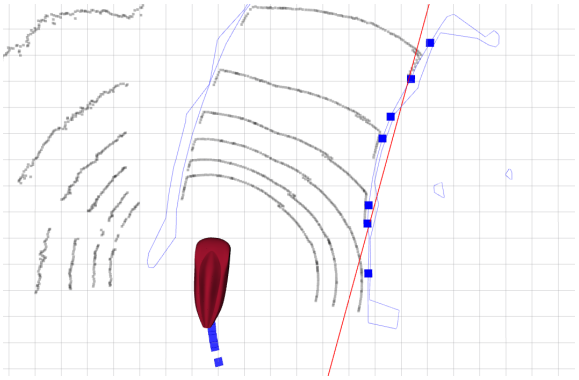
(a) Position 1, minimum distance 0m, maximum distance 7m



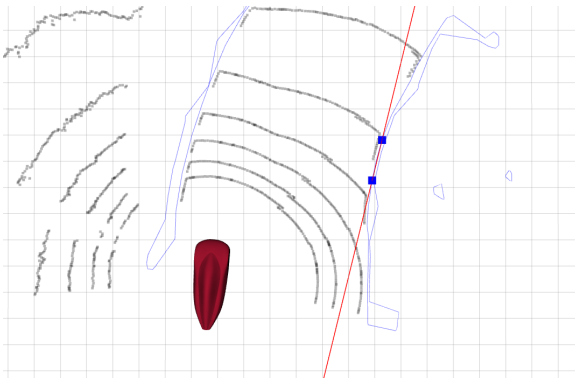
(b) Position 1, minimum distance 0m, maximum distance 10m



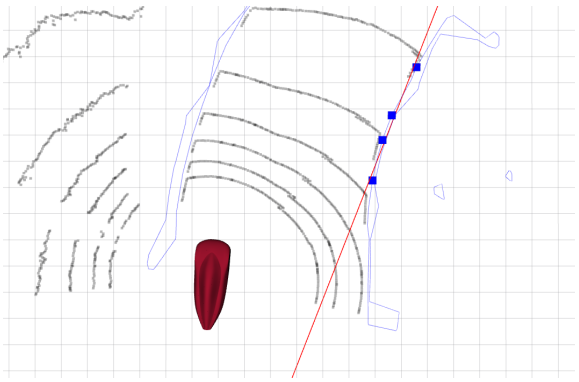
(c) Position 1, minimum distance 0m, maximum distance 15m



(d) Position 1, minimum distance 0m, maximum distance 20m

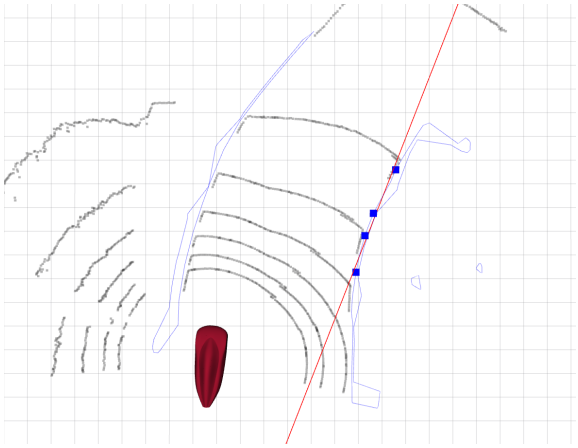


(e) Position 1, minimum distance 7m, maximum distance 10m



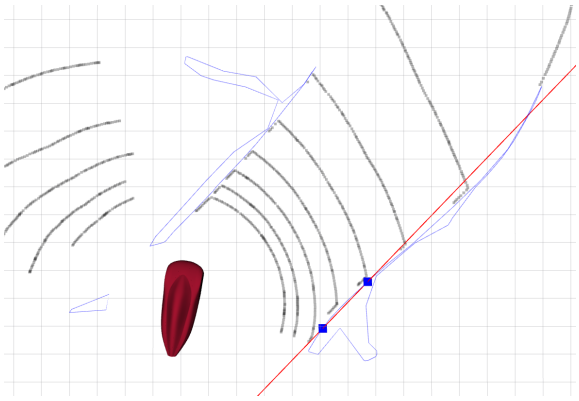
(f) Position 1, minimum distance 7m, maximum distance 15m

Figure A.1: Figures from the car in position 1

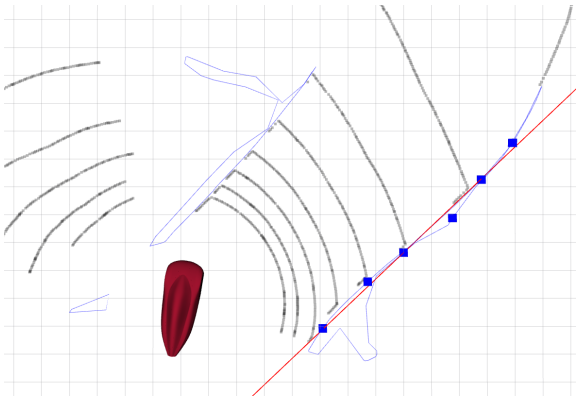


(a) Position 1, minimum distance 7m, maximum distance 20m

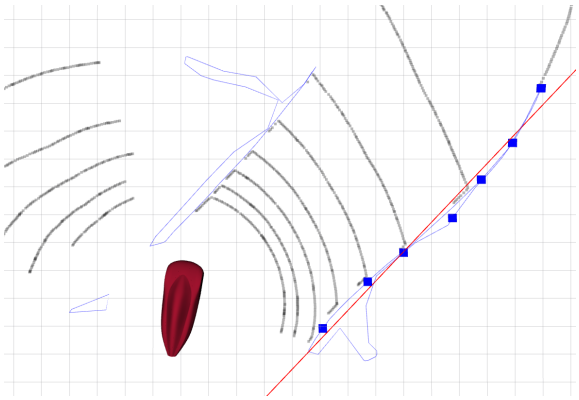
Figure A.2: Figures from the car in position 1



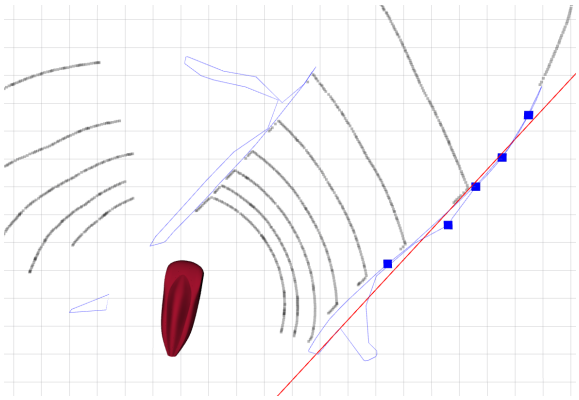
(a) Position 3, minimum distance 0m, maximum distance 10m



(b) Position 3, minimum distance 0m, maximum distance 15m

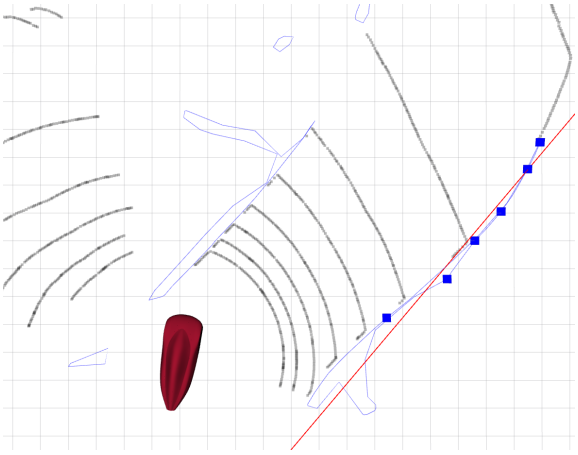


(c) Position 3, minimum distance 0m, maximum distance 20m

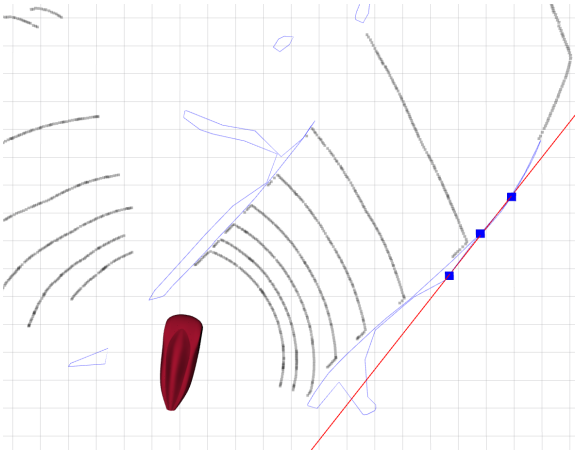


(d) Position 3, minimum distance 7m, maximum distance 15m

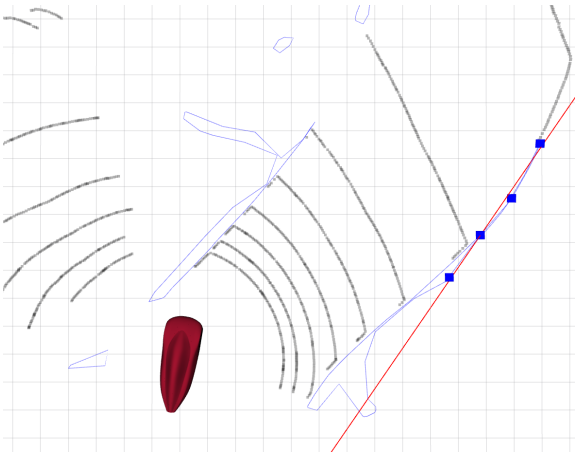
Figure A.3: Figures from the car in position 3



(a) Position 3, minimum distance 7m, maximum distance 20m



(b) Position 3, minimum distance 10m, maximum distance 15m



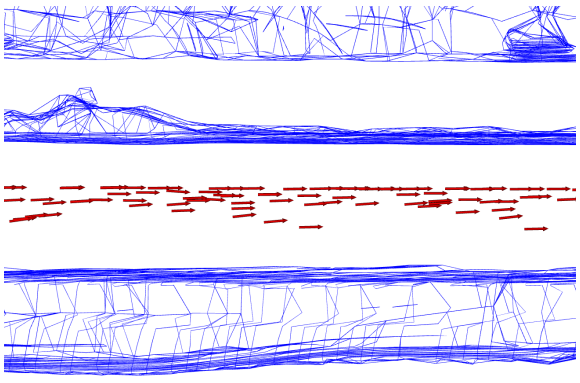
(c) Position 3, minimum distance 10m, maximum distance 20m

Figure A.4: Figures from the car in position 3

APPENDIX B

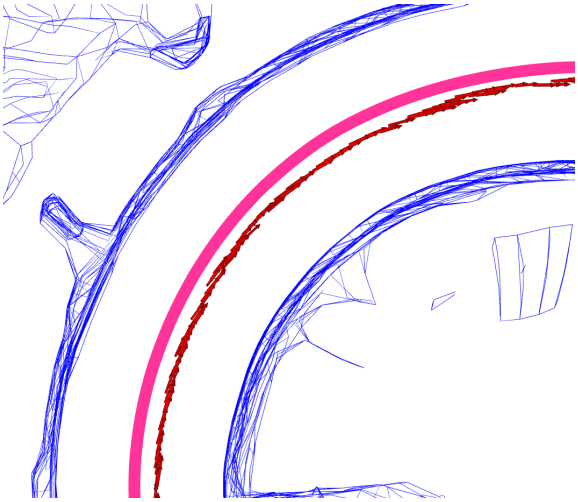
Data from AUC London 2018

Visualizations of data from AUC 2018 in London, showing barriers detected by the barrier detection in blue, drive points generated by the single barrier navigation in red, and the path driven by the car in pink. These drive points were generated after the competition and had no impact on the direction the car drove at the competition.

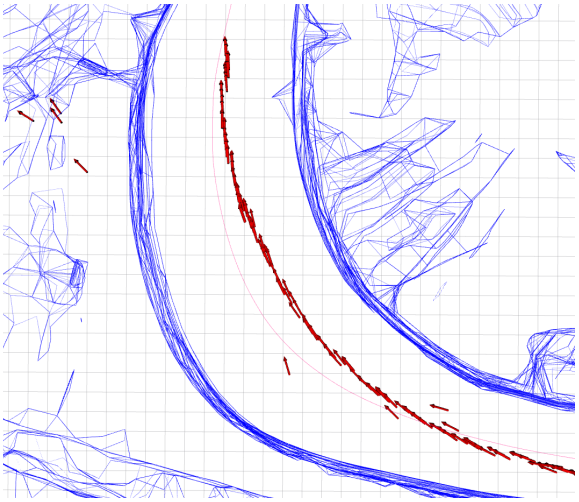


(a) This Figure shows the spread of the drive point placement, and is used to explain why using a moving average for the drive points could be useful.

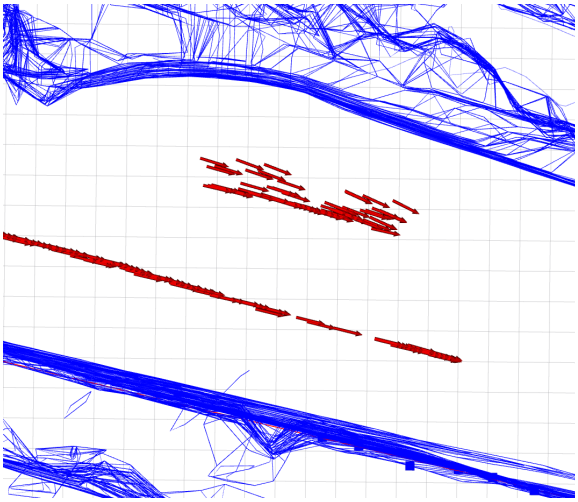
(b) This Figure shows how the single barrier navigation places the drive points differently, depending on whether the left or right barrier was used.



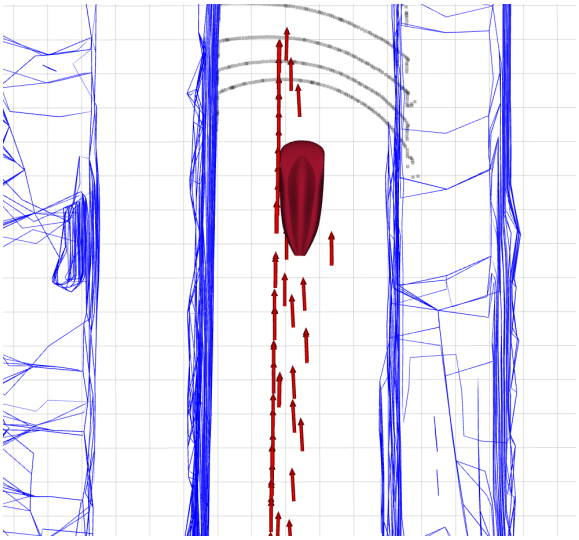
(a) This shows a corner with optimal conditions for using the single barrier navigation.



(b) This Figure shows a corner where the single barrier navigation created some wrong drive points.



(c) This figure shows how the single barrier navigation generates drive points a certain distance from the barrier, and how a change in the road width affects the generation.



(d) This Figure shows how a small road might affect the placement of drive points.

APPENDIX C

Map of competition track

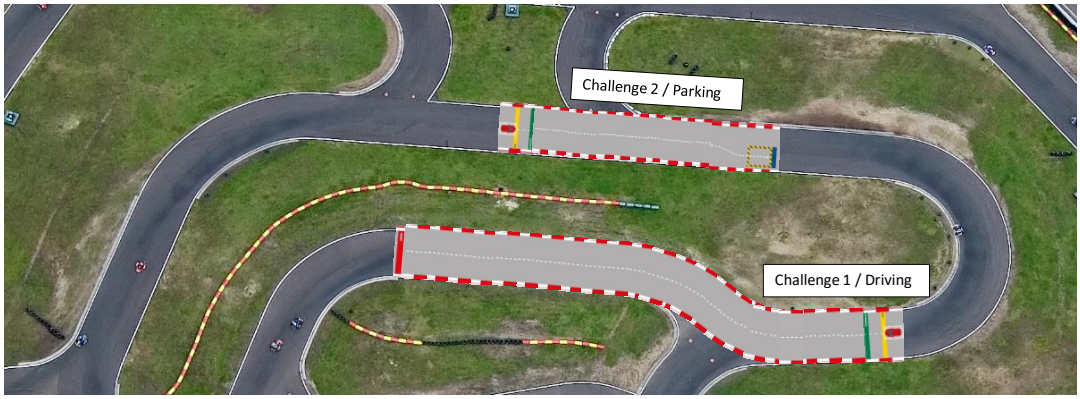


Figure C.1: Competition track for the autonomous competition at the Shell Eco-marathon 2019, as seen from above. The Figure was made by Shell

Figure C.2: Competition track for the autonomous competition at the Shell Eco-marathon 2019, as seen from above. The Figure was made by Shell

{2.7182818284}ϑωερτυθιοπσδφγηξκλ