ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

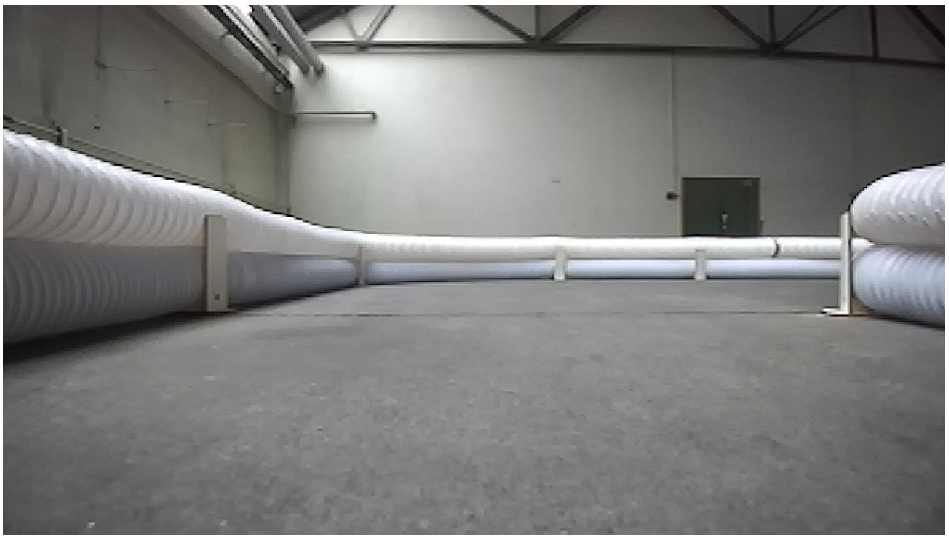
CENTER FOR PROJECT-BASED LEARNING

Department of Information Technology and Electrical Engineering

Spring Semester 2022

Track Bounds Detection for F1TENTH Autonomous Racing

Semester Thesis



Mike Boss
mboss@ethz.ch

March 2022

Supervisors: Nicolas Baumann, nicolas.baumann@pbl.ee.ethz.ch
Jonas Kühne, kuehnej@ethz.ch

Professor: Prof., Dr. L. Benini, lbenini@iis.ee.ethz.ch

Acknowledgements

I would like to thank Nicolas Baumann and Jonas Kühne for their guidance, interest, constructive advice, and the lengthy discussions held during the course of this project.

Abstract

This work tackles the problem of track-bound detection on an F1TENTH car using the on-board lidar and image sensors. At one-tenth the scale of formula racing, the F1TENTH setting offers a unique set of problems as well as fitting solutions for track bound detection for autonomous racing different to their full-scale counterparts.

This work's major contribution is a track-bound detection method based on Delaunay triangulation. The implications of an opponent on track bound detection are investigated, as well as potential solutions to the difficulties caused by the opponent car obstructing part of the visible track borders.

Declaration of Originality

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor. For a detailed version of the declaration of originality, please refer to Appendix B

Mike Boss,
Zurich, March 2022

Contents

List of Figures

Chapter 1

Introduction

1.1. Motivation

Autonomous driving has grown in popularity over the previous decade as a result of advancements in learning-based controls, utilised sensors, and available data. Autonomous racing has comparable difficulties to general autonomous driving in a quicker and more limited context, resulting in more edge-case scenarios of automobile behavior while limiting the collection of agents and probable circumstances.

While there are other autonomous racing series [6, 7], this work focuses entirely on the F1TENTH [8] racing series. The F1TENTH series features autonomous racing at one-tenth the scale of its full-scale formula counterparts. As a result of the small scale, the cars perform differently from full-scale race cars, and processing power is severely limited.

1.2. Objective

Autonomous racing software systems are primarily composed of multiple components that form a closely coupled chain of dependent sub-systems. These systems include vision, mapping, localization, and planning for the car used in this work.

This work focuses on the vision subtask of track bound detection. The track boundaries must be identified in a consistent and safe manner for route planning, resulting in left and right borders, as well as the associated center locations.

1.2.1. Outline

In chapter 3, the available sensors and systems of the utilized F1TENTH car, as well as the F1TENTH competition format, are described.

In chapter 4, the solutions to track bound detection are presented. First, the use of machine learning for track bound detection is explored. The first solution based on curve fitting is then presented. Finally, a Delaunay triangulation based solution is introduced.

In chapter 5, effects of an opponent on the solutions described in chapter 4 are described. In addition, a solution to these effects is investigated, and the consequences on track bound detection methods described.

In chapter 6, a discussion of the presented solutions and future additions is provided.

Chapter 2

Related Work

Autonomous driving is a vast research area with separate solutions based on the available sensors. For road detection, there are several solutions using 3d lidar combined with cameras [9, 10, 11, 12], as well as 2d lidar combined with cameras, either using a birds-eye-view [13, 14] or the image directly [15].

For racing, variations of SLAM[16, 17, 18] are popular for localization and mapping which in turn provide track bounds as well.

The Delaunay triangulation[19] has been used to discretize the space over which possible track paths can be searched [20].

Chapter 3

Setting overview

3.1. System

The F1TENTH cars use the Traxxas Slash 4x4 Premium Chassis and the NVIDIA Jetson Xavier NX as their compute platform.

The cars available sensors include the Hokuyo 10LX 2d-lidar sensor, as well as camera and IMU sensors. In the case of the car used by this work the camera system consists of a ZED 2i stereo camera by Stereolabs[21].

The car's compute platform runs ROS [22] publishing the sensor data as ROS messages. The lidar data is published as a LaserScan message consisting of 1080 readings over a frontal 270 degree field of view. The reading include distances and intensities of the lidar scan based on the angles right-to-left. The image data is published in color and rectified as CompressedImage messages for both the left and right camera.

3.2. Competition

F1TENTH races are a combination of time trials and head-to-head competitions. In a time trial, the highest score, calculated from a combination of the fastest lap and number of laps completed over a certain time, wins. In a head-to-head race, two cars compete against each other. Before racing, there may be an additional mapping round to allow teams to map out the course ahead of time at slower speeds.

F1TENTH race competitions are offered at different locations during the year. Race tracks depend on the location where the race is held, as the available space constrains the race and the floor is uncovered, as seen in 3.1. Visible are the unique flooring of the location as well as the two stacked standardized pipes used as track boundaries. In [2] we

can observe different flooring as well as different types of pipes used for track boundaries in the IROS 2021 competition in Prague. On the right, improvised track bounds can be observed consisting of taped sheets of some kind of paper. We can observe another set of different flooring and track bounds in 3.3 from a race held at CPSweek 2019.

Figure 3.1.: A car racing in a F1TENTH competition as seen on [1].

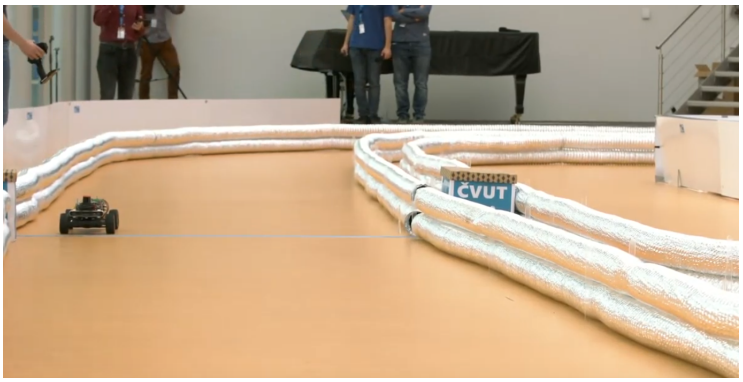


Figure 3.2.: A car racing in the IROS 2021 competition in Prague as seen on [2].

Figure 3.3.: A car racing in the CPSweek 2019 competition as seen on [3].

For the detection of opponents cars are outfitted with April-tags during head-to-head races as seen in 3.4. One car has the tags applied on the back and front, while the other only has one on the back.

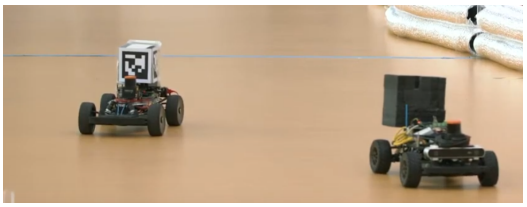


Figure 3.4.: Two cars with April-tags after a head-to-head race at the IROS 2021 competition in Prague as seen on [2].

Chapter 4

Track Bound Detection

4.1. General

This chapter first gives an overview of the filtering of the lidar in section 4.1.2 and the possibility of the use of machine learning for track bound detection in section 4.1.2. Afterwards, a first solution to detection of the track bounds by fitting curves is presented in section 4.2. Finally, a solution using the Delaunay triangulation is introduced in section 4.3.

4.1.1. Lidar scan

The lidar scan is provided as a ROS *LaserScan* message of 1080 readings over a 270 degree frontal field of view. The readings include distances and intensities based on the range of angles right-to-left. To convert the *LaserScan* into a *PointCloud2*, the *laser_filters scan_to_cloud_filter_chain* is used. The *PointCloud* format provides the values as x-y-z coordinates. As the used sensor is a 2-d lidar scanner, the z-axis can be disregarded after transforms. The original scan contains a lot of noise, which is filtered using the scan_filter_chain. A *ScanShadowsFilter*, *LaserScanRangeFilter*, and a *LaserScanAngularBoundsFilter* are used to remove this noise. The *ScanShadowsFilter* removes artifacts caused by the veiling effect when the edge of an object is scanned by the lidar scanner. This artifact is seen in 4.1 on the left, as well as the nearly completely removed version in the middle due to the usage of the filter. The *LaserScanRangeFilter* removes any lidar points that are at a distance over the maximum scan range, however none are present on the test track. The *LaserScanAngularBoundsFilter* is employed to filter points that lie behind the car which provide little informational value to the track bound detection. The result of all the filters in use, is seen on the right in 4.1. It is of notice that a lot of scan artifacts remain. The *ScanShadowsFilter* is initialized

with a minimum and maximum angle of 5 and 175 degrees, respectively. Additionally, neighbors are set to 5 and the window to 1. For the *LaserScanRangeFilter* the parameter *use_message_range_limits* is set to true. Finally, the *LaserScanAngularBoundsFilter* uses a lower and upper angle of -1.77 and 1.77 respectively.

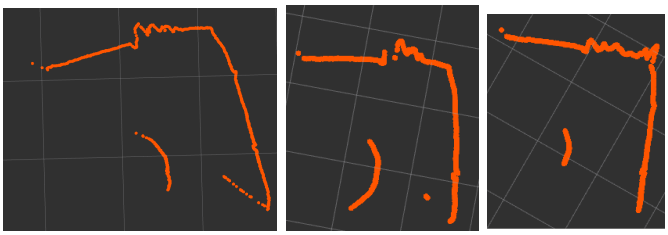


Figure 4.1.: On the left the un-processed lidar scan is shown. The middle shows the lidar scan with a *ScanShadowsFilter* used. The right shows the lidar scan with all filters enabled.

The lidar scan is down-sampled from the full scan to 300 points. This has little effect on track bound detection as, with great probability, enough important points are still included. This is done for performance, as a lot of points are redundant. The sampling is done randomly over all points, which is not ideal as sampling based on ranges or other criteria would better sample points that are far apart. Given the nature of the lidar scanner, it scans more values the closer the track bounds are to the car itself and provides fewer values at the track bound location the more the track bound varies over the scan angle. Therefore, the least amount of information is provided over straight track bounds. An example of the down-sampled lidar is shown in 4.2

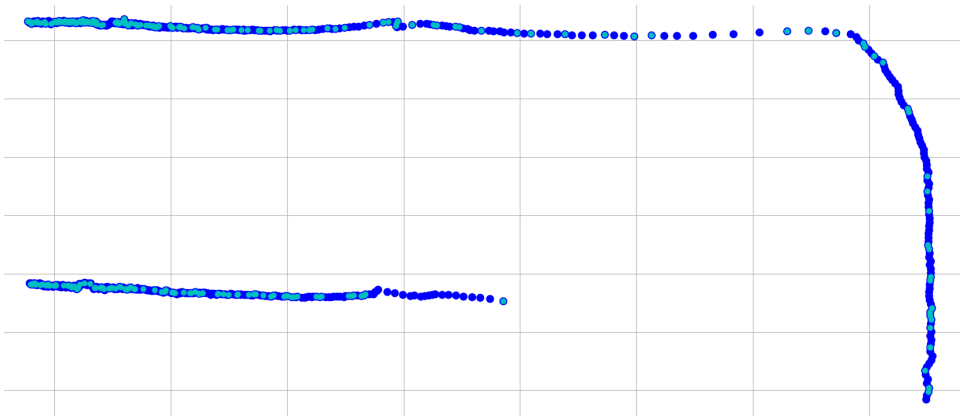


Figure 4.2.: The original lidar is shown in dark blue while the selected points of the down-sampled version are shown in cyan.

4.1.2. Machine Learning

Solutions using machine learning for track-bound detection were investigated. As there are no good datasets of enough similarity to F1TENTH racing, direct supervised learning seems infeasible. Even if data was recorded, the generalizability of such a solution may be questionable given the many different track flooring and track boundaries used as seen in 3.2. A transfer learning approach from another dataset could improve generalizability. However, no datasets of races that employ physical track boundaries higher than the car are available.

A feasible solution may be a self-supervised model that learns to predict the curves best suited to the given lidar points. It is unclear, however, how an opponent may be integrated into such a model. Given the highly specific setting, the model would be trained for its advantages over a non-learned model, which seems unclear. If the task is just track bound detection, predicting the future state for self-supervision would induce some part of learned car behavior in the model, which is unnecessary complexity for track bounds detection.

4.2. Curve Fitting

4.2.1. Overview

To detect the track bounds and center curve, first the left and right track boundaries are separated using clustering. A curve is fitted through both clusters. By sampling points on both lines at equal intervals, the center curve can be constructed as the middle of the line induced by each point pair.

4.2.2. Clustering

An AgglomerativeClustering from the Sklearn [4] library is used to separate the left and right track boundaries, with the number of clusters set to two, the linkage set to single, and the affinity set to Euclidean. As the track boundaries consist of only the left and right sides, assuming two clusters is a natural choice. When merging potential cluster candidates, single linkage ensures that the clusters with the minimum distances between all observations in the cluster pairs are merged first. This leads to clear clustering lines as the clusters' inner distances are almost always closer to other clusters in the case of a race track. The behavior of different linkages is visible in 4.3. The bottom comparison is similar to track boundary separation as seen in 4.4.

After separating the track bounds into two clusters, it is unclear which cluster corresponds to the left and right side, respectively. Either the two points with the lowest x-axis value in either cluster may be compared over their y-axis value or the median/mean of the

y-axis value of the two clusters may be compared to decide which cluster is on which side. For the used test data, both solutions performed without mistakes, however a median/mean solution may be preferred as they are more robust to potential outliers.

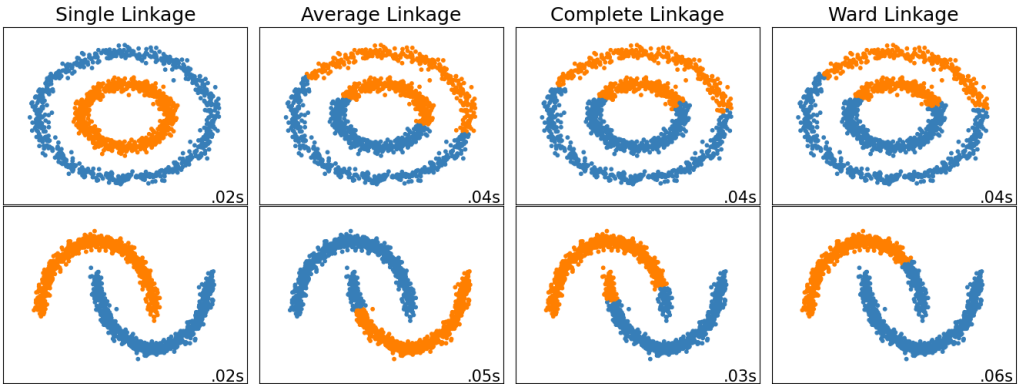


Figure 4.3.: A comparison of different linkage attributes used in the AgglomerativeClustering class by Sklearn. [4]

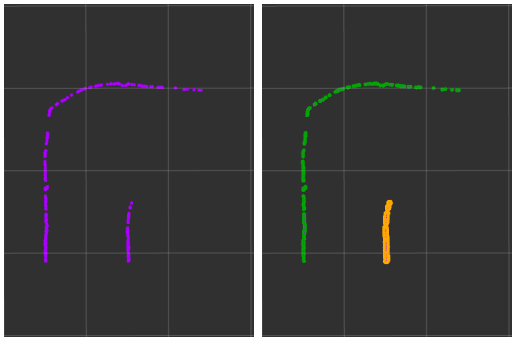


Figure 4.4.: In the left image the pre-processed lidar data is shown in purple. In the right image the left and right clusters are shown in green and orange respectively.

Incorrect clustering

Even though the assumption that the track consists of two clusters is correct, the clustering method produces incorrect clusters in certain cases. In the case of a chicane, the lidar scan will produce three distinct clusters of points instead of two. This is due to one side of the track's boundaries obstructing vision to a portion of its own side further on. Using single linkage clustering fails, as in most such cases, the other side is actually closer than the correct cluster. A failure of such a kind can be seen in 4.5. One solution would be to have an undefined number of clusters and decide, based on the direction of the line induced by the cluster points, which side they belong to. This fails for straight line clusters, which would need to be handled separately by comparing the y-axis positions of

the points. The removal of enemy lidar points further exemplifies this problem, as seen in 5.

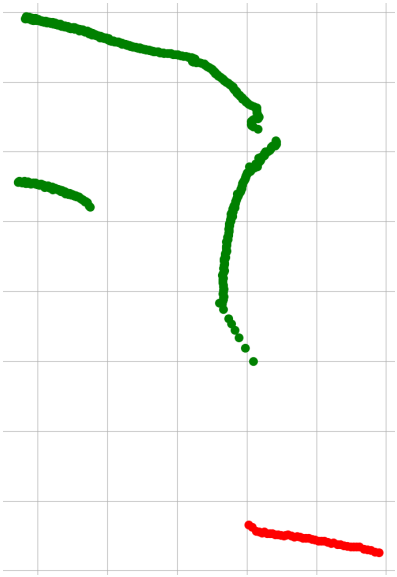


Figure 4.5.: Shown are the lidar scan points at the entrance of a chicane clustered with the described method. The first right cluster blocks vision of the lidar sensor to the rest of the right track bounds leaving a gap between it and the next right cluster. As this gap is greater than the distance to the left cluster the method misclassifies it as left track bounds.

4.2.3. Curve fitting

Latent variable

It is not possible to fit a linear model directly to the points as the track bounds do not correspond to the linear function seen in 4.4 as the corner is a 90-degree turn. The distance to the origin is introduced as a latent variable to predict the x-y-coordinates using a linear model. The origin always corresponds to the position of the car given the lidar's transformation.

Binning

After calculating the distances, the points are down-sampled using uniform binning using 100 bins over the distances to the origin. The points with the minimum distance to the origin are selected to further remove points outside of the track bounds. As seen in 4.6, this selection measure is imperfect as, for a given bin, a point may be selected that is

closer to the origin but more distant to the center-line. In 4.6, this is visible at the top, where points are selected while multiple points are closer to the center-line of the track. As the points are mostly uniformly spaced, this occurs only at lidar scanning artifacts, as seen in 4.6 at the top. Even though imperfect, the measure is computationally efficient in comparison to a more perfect measure. For example, taking only the point with the highest distance given an axis orthogonal to the linear line fitted through the points in the bin.

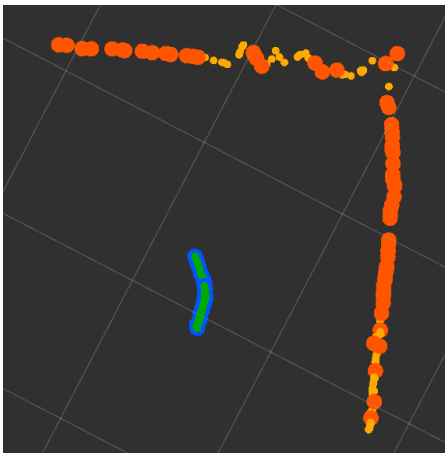


Figure 4.6.: The lidar scan of the left and right track bounds are visible in green and orange respectively. The selected binned points are visible in blue and red.

Fitting the curve

For each track-bound cluster, a curve is fitted over the distances from the origin to the x-y-coordinates of the corresponding point. A Sklearn pipeline consisting of a RobustScaler, a SplineTransformer, and a RideCV model is used to fit the curve. Given the noise present in the lidar data, the RobustScaler is the natural choice. However, compared to the StandardScaler, no great differences are obvious in the final curve. The SplineTransformer is initialized with the number of splines set to 8, the degree of the polynomial set to 2, and the sampling of the knots based on the quantiles. The number of splines is determined by testing on available data, and the best value may differ from track to track. However, as most tracks should be able to be described by a lower degree polynomial, the number is at least high enough to be able to model the track bounds quite closely. Higher spline numbers may introduce unwanted overfitting between splines. The RidgeCV model uses 10 alphas from a logspace between -6 and 6. The RidgeCV model performs well and is fast compared to more robust models.

After the model is fit, 1000 points uniformly sampled between the minimum and maximum distance are predicted as seen in the two examples in 4.7 .

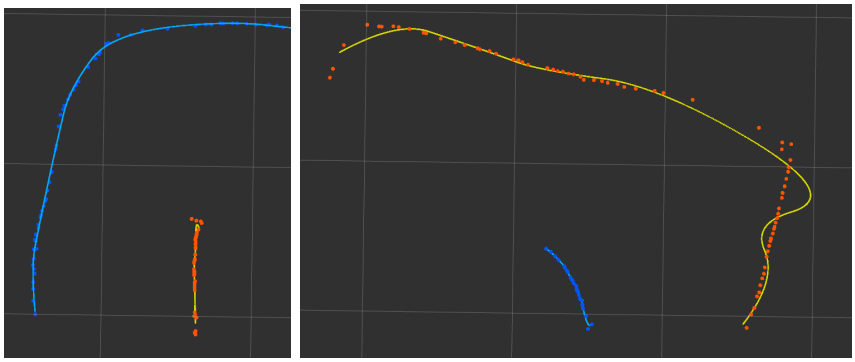


Figure 4.7.: Two examples of fitted lines. The left lines is visible in blue, the right line in yellow. The used points are visible in green for the left points and red for the right points. In the left image, the lines are fitted quite well. However, in the right image, the right line shows some erratic behavior on the right side.

4.2.4. Center line

The center line can be constructed by taking the middle of the two lines induced by the left and right points. The point pairs are chosen as the predicted points on each line. As both lines are predicted using the same number of points, each point-pair has the same relative distance on its line compared to the other point. An example of the constructed middle line can be seen in 4.8.

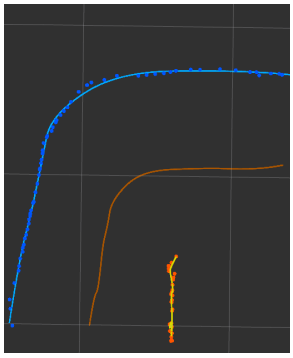


Figure 4.8.: The left and right line are shown in blue and yellow respectively. The constructed middle line, by taking the midpoints from the two lines, is shown in brown.

4.3. Delaunay Triangulation

4.3.1. Overview

Another method for determining track bounds is to first identify the center line directly using the Delaunay triangulation, and then determine the left and right points based on the triangulation's orientation.

To achieve this, first the Delaunay triangulation of the lidar scan is calculated. A search for the possible paths containing the center line is started from the simplex containing the origin. The search finds all paths starting from the origin and ending at an infinite simplex. The correct path is selected by comparing their distances to the origin. The paths are backtracked. This allows for construction of the center line as it is chosen as the midpoints of the taken Delaunay edges. As the Delaunay triangles are oriented counter-clockwise, the left and right points can be selected at the same time by choosing the first point on the edge as right and the second as left. This backtracking of the triangulation loses information in the corners, which can be reconstructed by fitting a line through the selected points.

4.3.2. Delaunay Triangulation

Description

A triangulation subdivides the space contained in the convex hull into a set of triangles. The Delaunay triangulation is a special kind of triangulation as its triangles can be described as circles with maximal radius passing through three points with the center inside the convex hull. The Delaunay triangulation is the dual of the Voronoi diagram, as seen in 4.9. Even though the Voronoi diagram is not used directly to re-construct the center line, it provides visual intuition that the center line can be re-constructed as seen in 4.10. A triangle is denoted as a simplex as the Delaunay triangulation generalizes to more than two dimensions. A triangle that extends to infinity, as its induced circle contains less than three points, is denoted as an infinite simplex.

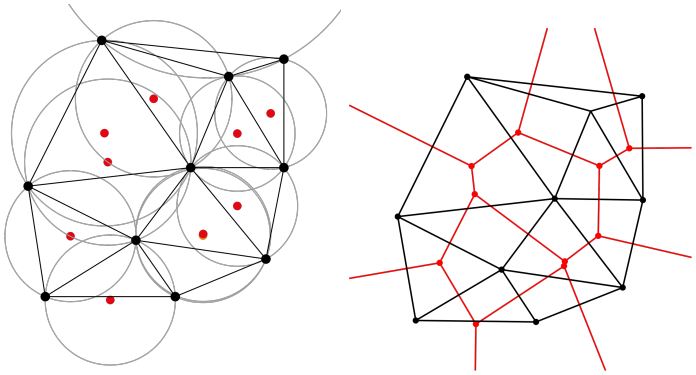


Figure 4.9.: An example of the Delaunay triangulation is shown on the left. The triangles and corresponding vertices, shown in black, make up the triangulation while the center points of the circumcircles of the induced circles are shown in red. On the right, the Voronoi diagram of the same points is shown. It is the dual of the Delaunay triangulation and can be constructed by connecting the circumcircles of the triangulation. [5]

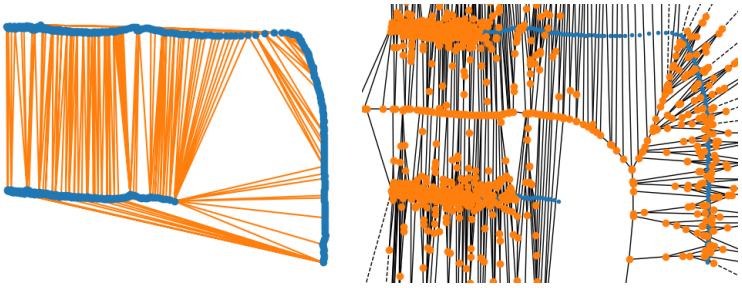


Figure 4.10.: A Delaunay triangulation of the track on the left and it's corresponding dual, the Voronoi diagram, on the right. The lidar points are visible in blue, the Delaunay triangle edges and Voronoi points in orange. On the right the center line of the track is clearly visible.

Track triangulation

To calculate the Delaunay triangulation of the track lidar scans, the SciPy [23] Delaunay triangulation is used. This implementation differs from the more commonly used CGAL implementation [24], as the triangulation is returned as dense matrices describing the neighborhoods instead of iterators. The neighborhood description follows the same standard as the CGAL implementation. As such, the vertex with index zero of a triangle is opposite to the edge with index 0. Additionally, vertices and edges are ordered in a counter-clockwise fashion, allowing edges and their corresponding edges to be accessed by simple module operations. The SciPy implementation does not contain infinite simplices, instead opting to describe infinite simplices indirectly as neighbors denoted with

-1. In testing on lidar scans from a test track, the triangulation produced approximately twice the number of simplices as lidar points. It ran in about 3-7 ms for about 800 lidar points and 1-3 ms for 300 lidar points. Delaunay triangulations for two different parts of the track are shown in 4.11.

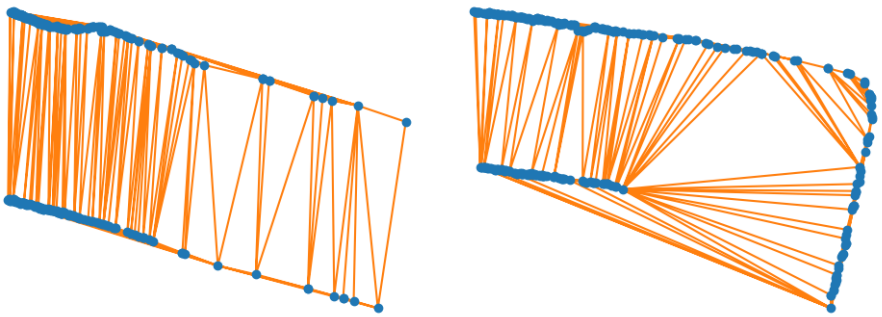


Figure 4.11.: Plots of Delaunay triangulations of two lidar scans of the race track. The lidar points are visible in blue while the triangulation is visible in orange.

4.3.3. Path

Search

Given a normal track layout, the center line of the track always leads from an infinite simplex to another infinite simplex. As the lidar scan is only over 270 degrees, there is an infinite simplex behind the car as no lidar points are recorded. Given the lidar's finite scan length and obstructing walls, another infinite vertex is at the visible end of the track if the car is not surrounded by walls.

To find the center line, all possible paths could be explored and the center line selected based on some set of criteria. However, such a search is computationally infeasible in the setting of fast paced racing. Additional information about the car's position and track can be used to reduce the search space.

The lidar scan extends behind the car, therefore the car's position, the origin, lies in a simplex on the track inside the track bounds. The search is started from the simplex containing the origin, which can easily be located given the Delaunay triangulation. A modified depth-first-search is performed; no simplex is visited twice.

A priority queue is initialized with the simplex containing the origin. For every visited simplex, all its non-visited neighbors are inserted into the queue. The priority corresponds to the length of the traversed edge of the Delaunay triangulation. As the queue is priority minimizing, the edge lengths are subtracted from a constant such that the longest edge lengths are at the front of the queue. It is clear that the search follows the track first

as it corresponds to the longest edges as long as sufficient lidar points are on the track bounds.

With no stopping criteria, this search corresponds to the construction of a maximum spanning tree. Given the many simplicies that lie outside of the track boundaries, the full construction of the tree is inefficient. It is also unclear how to efficiently select the path corresponding to the center line from such a maximum-spanning tree. The tree was constructed based on the edge length, which is not a good selection criteria on its own. The same is true for path length, as paths following smaller edge lengths may actually be longer than the true center line. Both of these selection criteria would lead to false path selections in 4.12. Selection based on the mean or sum of edge lengths would almost certainly select the path exiting at the first hole in the lidar scan. Path length as a selection criteria would select the correct center line, however the path would be extended into the last curve instead of ending at the track end. Additionally, any selection that solely focuses on the distance to the origin of either the points of the simplex or the midpoint of the traversed edge will fail in certain scenarios, such as the one seen in 4.11 on the right. There exist combinations of these selection criteria that are a feasible solution, however as all paths need to be evaluated, these are computationally infeasible.

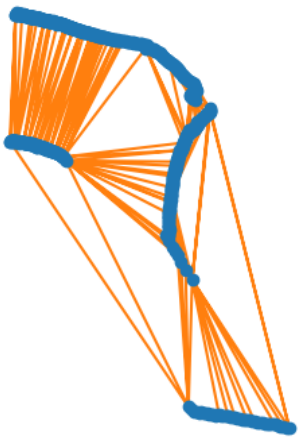


Figure 4.12.: The Delaunay triangulations of a lidar scan of the start of an S-curve on track.

A solution would be to stop the search after a certain number of infinite vertices have been found, as they directly correspond to holes in the lidar scan. Even though this solution works for 4.12 with three found infinite vertices, it would fail with the same number of allowed infinite vertices for any other scan that does not have a hole in the lidar scan, such as 4.11. It is unclear how the number of infinite vertices should be selected without evaluating the paths.

Instead of constructing the whole tree, the search only adds edges of a simplex if the length of the edge is at least a third as long as the longest edge previously seen. This criteria allows us to reduce the search space significantly as any edge through the track bounds is discarded if enough lidar points are present without limiting the search too much, as it still allows the search to continue as seen in 4.12. If the track varies greatly over distance, one may consider using only a certain number of last seen edges. However, it would be highly unlikely for a track to narrow to over a third of the broadest part. Additionally, this criteria naturally ends the search once no edges that are long enough relative to their simplex edges remain. The search runs in approximately 6-7 ms on the triangulation of 800 lidar points and in about 4-6 ms on one with 300 lidar points.

Selection

During the search, any time an edge is selected that leads to an infinite simplex, therefore escaping the track boundaries, the simplex is saved as a potential candidate for backtracking. This allows for the selection of only those paths that end at the exit of the track. From these candidates, the simplex with the point that has the maximum distance from the origin is selected. If two candidates share this point, the one with the greatest distance from its midpoint on the edge to the infinite vertex is chosen.The selection takes less than one millisecond for both the 800 and 300 lidar points.

Backtracking

Once the correct candidate simplex has been selected, it is possible to backtrack from this simplex through the use of a predecessor array saved during the search. The backtracking stops once the simplex containing the origin has been found. Given that the search starts from the origin simplex, backtracking will always lead back to the origin simplex. During this step, we can exploit the nature of the Delaunay triangulation to determine which side each track bounds lies on. Concretly, the vertices and edges of the Delaunay triangulations are oriented counter-clockwise. As the backtracking traverses the center line in reverse, the first vertex of the traversed edge belongs to the right and the second one to the left track bound. The backtracked paths can be seen in 4.13. As the triangulation makes no assumption of the underlying track layout, the end of a backtracked center line may include unwanted points, as seen in 4.14. It is unclear how one could stop the search before these points are considered, however through empirical findings, it is clear that one can easily disregard about 10 endpoints to mitigate this issue. This shortens the center line while the left and right track bound points can be kept. A similar problem point is visible in 4.13 in the right plot. The left point associated with this center point would not be included normally as it is outside the track bounds because it is a lidar anomaly. One may consider removing these left and right points as well, to mitigate such points being included in the track bounds.

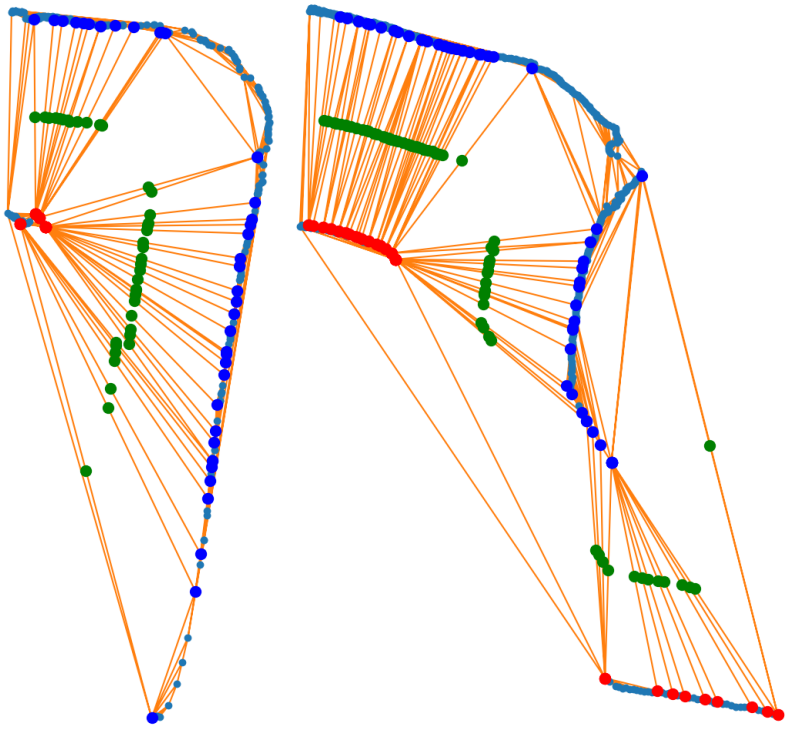


Figure 4.13.: Two examples of backtracking the search path. The selected center line points are visible in green, while the corresponding track bound points are blue and red for left and right respectively.

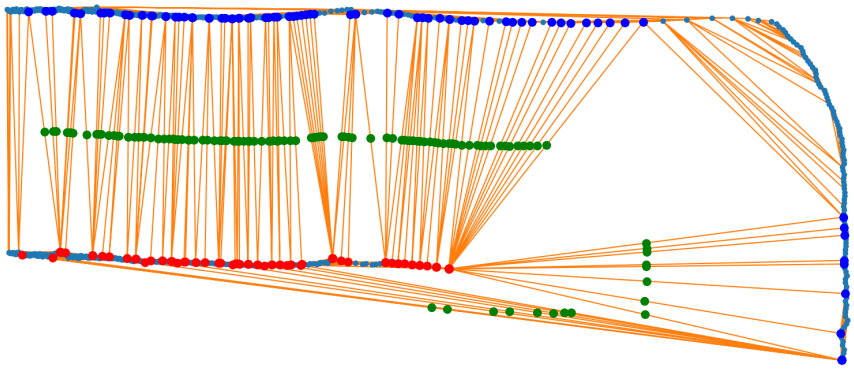


Figure 4.14.: An example of correct backtracking that includes points uncharacteristic to a center line at the end.

4.3.4. Curve fitting

The triangulation misses out on a lot of information about the track bounds in the curves because it only includes circles with the greatest radius.This behavior is visible in both 4.13 and 4.14 in the top right curve. To compensate for this, a curve similar to the first solution from section 4.2 may be fitted through the points. In contrast to the first solution these points are much better suited for curve fitting as only those points that are closest to the track are included. This is evident, as any point that would be closer would lie inside one of the induced circles of the other points.

The curves are fitted similarly as described in 4.2.3. The left and right lines are fitted separately by first calculating the distances to the origin as a latent variable. A polynomial of the fifth degree is fitted using Sklearn's PolynomialFeatures and the same RidgeCV as in 4.2.3. It is of notice that no binning has to be performed as the selected points already correspond to binned points. However, these are now the closest points to track in contrast to before.

The center line is constructed using the same midpoint sampling as before to ensure that it is actually in the center of the newly fitted lines. One may notice that this process is imperfect, as seen in 4.15 in the top right plot. This is a result of the uniform sampling from the lines instead of actually calculating a center point, which would only be possible on the shorter line. However, as seen in the top left and bottom plots, this line is normally quite centered and is always safe.

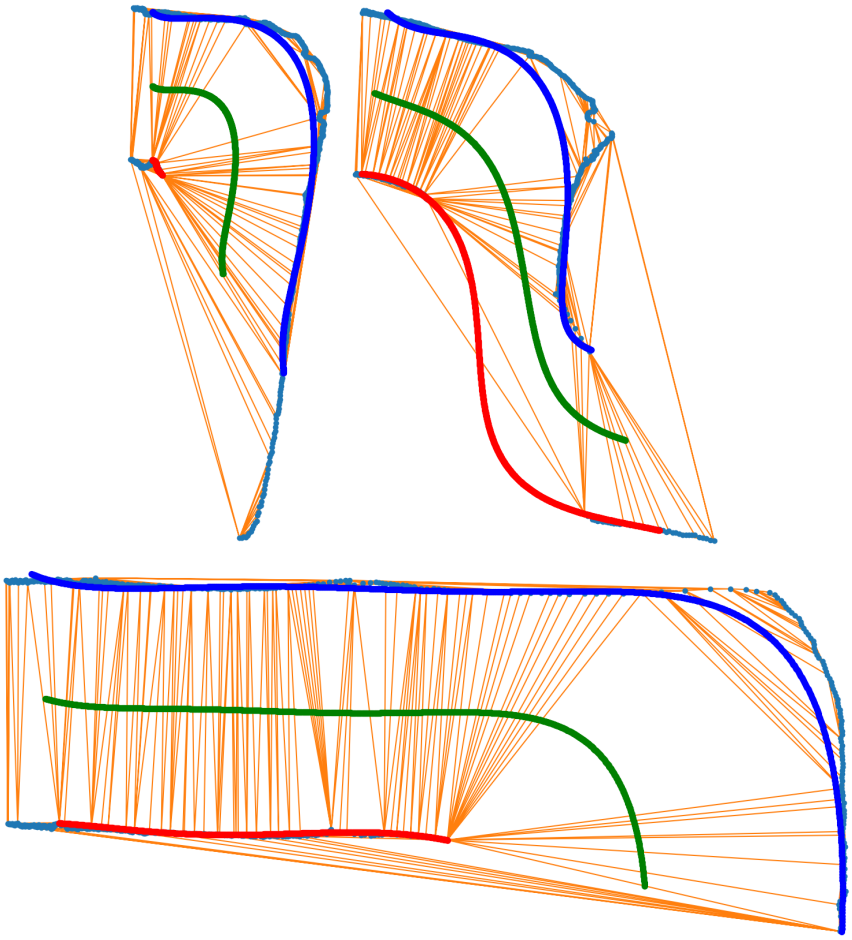


Figure 4.15.: Examples of fitted lines on the points selected on the Delaunay triangulation.

Chapter 5

Opponent Removal

5.1. Problem

In a head-to-head setting, the opponent will introduce another distinct set of lidar points to the lidar scan if it is in the visible range of the lidar sensor. Any approach that makes assumptions about clearly distinguishable track bounds in the lidar scan will find incorrect track bounds. Not only does the opponent obscure visibility to the track bounds behind it, but it also introduces points in the scan that may be difficult to distinguish from noisy track bounds. The effects of solutions that remove the opponents' lidar points on track bound detection described in the previous chapter are explored. The opponents' lidar points may be removed by filtering based on intensity, position of the points, or using the camera image to recognize the April-tag. Lidar points can be transformed into the image space and all points inside the bounding box of the detected opponent can be removed. The removal of the opponents' lidar points leaves a hole in the track bounds that may lead to incorrect track bound detection by certain solutions.

5.2. Effect on curve fitting

Removal of the opponents' lidar points may lead to wrongly clustering the track bounds if the clustering method described in 4.2.2 is employed. This is evident as a hole in the lidar scan of the track bound is closely related to the setting in the chicane where a part of the track bound is not visible. An example of a wrongly clustered track is shown in 5.1.

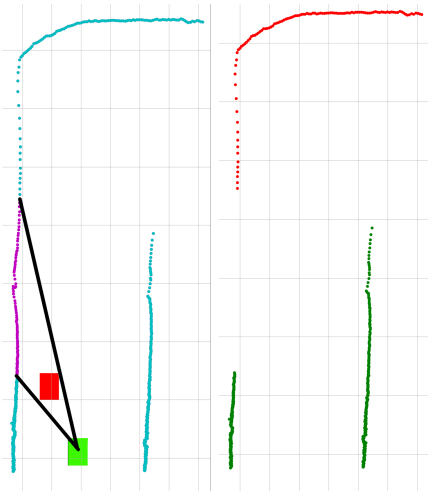


Figure 5.1.: An illustrative example of removal of the opponents lidar points and it's effects on clustering. The lidar scan is made up of both the cyan and magenta points shown in the left plot. As the opponent's car blocks part of the scan the magenta points are not visible and therefor not available to the clustering. As seen on the right, the clustering incorrectly clusters the track bounds because of the missing points.

5.3. Effect on Delaunay triangulation

The solution using Delaunay triangulation does not directly suffer from the same problems as clustering if the opponent's lidar points are removed. This can be seen using the same example track part in 5.2. As the triangulation explores all possible exits, it still finds the correct track exit with the maximum distance to the origin. The reconstructed lines cleanly fill in the produced gap even though the points are missing.

Another example is shown in 5.3 by removing the lidar points in front of the car from the entry to the chicane. This divides the track into four distinct clusters of lidar points. Using the Delaunay triangulation, it is still possible to reconstruct the actual track boundaries and corresponding center line. However, the center line is even less centered than before as a result of the missing points, leading to differently fitted curves.

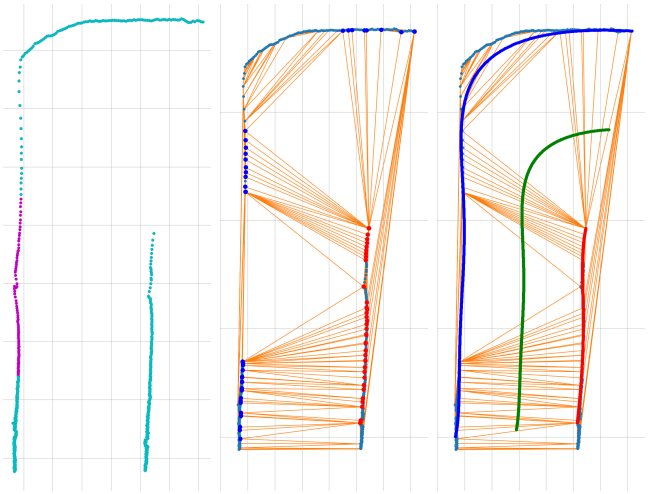


Figure 5.2.: The points in magenta are removed from the original scan and only the cyan points are left. As seen in the middle plot track bounds are still correctly separated and well constructed as curves in the plot on the right. The opponent's car would be in front to the left.

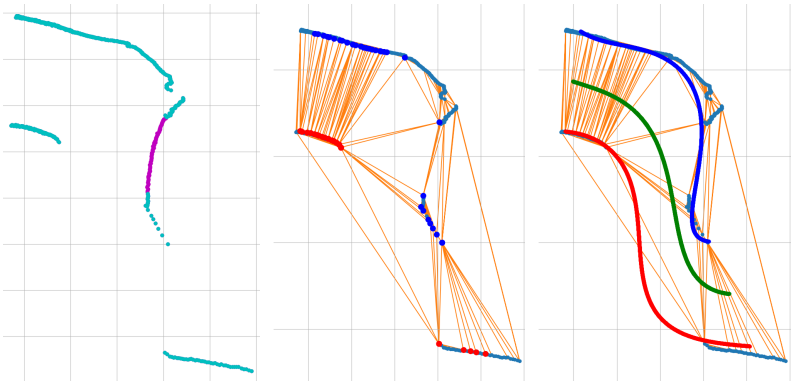


Figure 5.3.: Another example with points removed from the entry to a chicane leaving four distinguishable clusters of lidar points. The opponent's car would be right in front.

The solution is not without its failure points, as seen in 5.4. In the shown example, the detection of the track bound still works even though there are only two points visible on the right track bound. However, removing one more point leads to the curve fitting failing, which can be ignored in this case. If the whole of the right track bound is obstructed, the solution fails to find any track bound. In the case where the opponent's car is close to the right or left side of the car, it may be advisable to leave the lidar points of the car instead of removing them.

Another failure may occur if the gap created by removing the lidar points of the opponent's car has a greater distance than the track width. It is to be noted, however, that

this is not sufficient in and of itself to cause a failure, as this is also true in 5.2. The failure is caused if the gap is on the side where the track ends, as it is possible that the search finds the end before it finds it through traversing the actual track. This may be solved by, instead of simply ignoring already visited simplicies, comparing the length of the traversed edge or current path length and overwriting the predecessor accordingly.

If the opponent's car is right beside or slightly behind the car, it may lead to the origin simplex being infinite if the lidar points are removed. This would stop the search immediately as no neighbors are added to the queue. Either handling the start separately in this case or not removing the opponent's lidar points should solve this failure point.

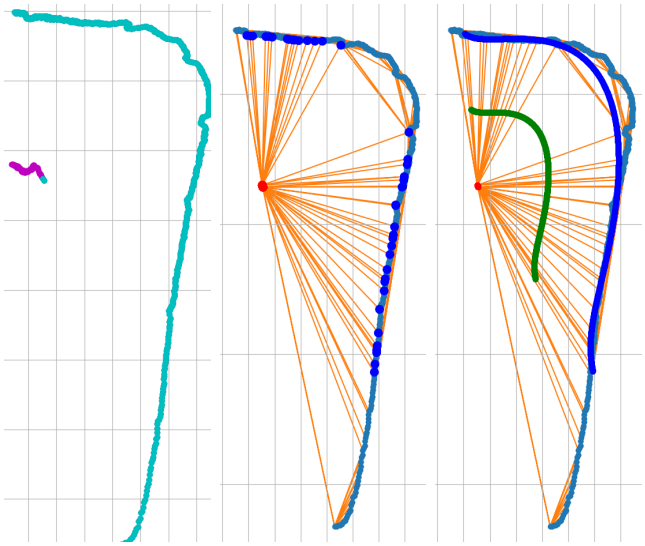


Figure 5.4.: An example of a possible failure point for the track bound detection using Delaunay triangulation. The opponent's car would be to the right. As seen in cyan there are still two points visible of the right track bound. Were the opponent's car to block all of the right track bound, no path could be found in the Delaunay triangulation. Two points are enough to reconstruct the correct center line.

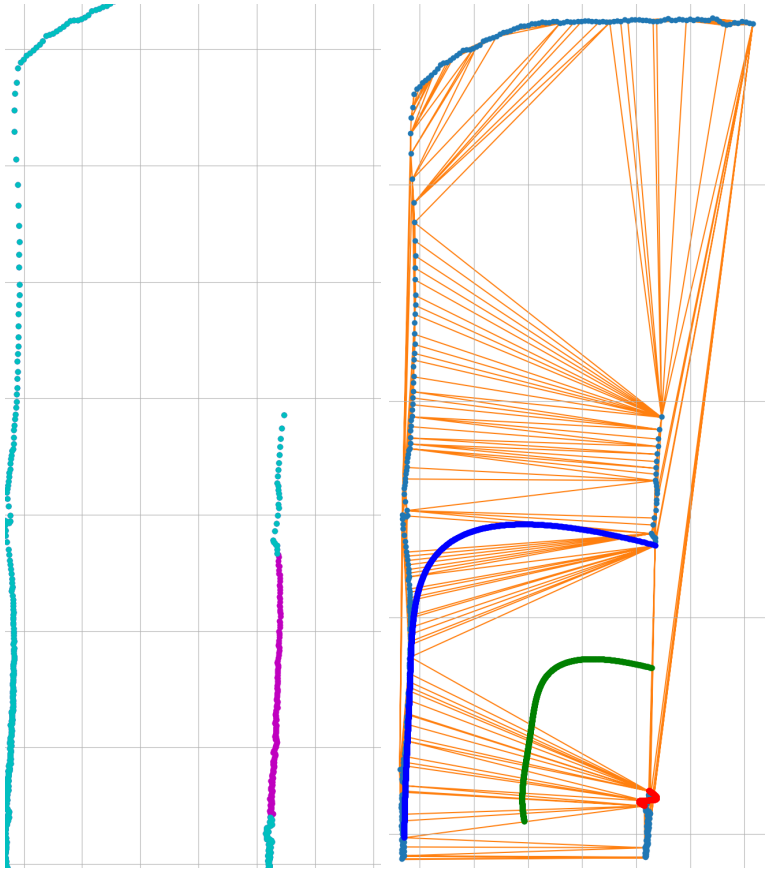


Figure 5.5.: An example of a failure of the track bound detection using Delaunay triangulation. The opponent's car would be to the right. The removed gap, shown in magenta, is bigger than the track width leading the search to find the end through this gap. As the end is already found the actual path through the track is not explored.

Chapter 6

Conclusion

In this work, solutions for track-bound detection were discussed. A solution based on the Delaunay triangulation was introduced, which is simple and robust to noise in the lidar scan. The effects on this solution of removing an opponent from the lidar scan were explored. The solution is robust for most scenarios in which an opponent is present. For some scenarios, the solution may have to be adapted or the opponent's lidar scan left in the scan.

Even though the solution runs fairly fast (un-optimized in python 35ms), an optimized version written in C++ using the CGAL library to work with the Delaunay triangulation should be significantly faster. There are multiple points where the algorithm may be made faster, such as, for example, substituting the locating of the origin's simplex as well as more track-specific search procedure optimizations.

There remain situations where the current implementation shows unstable behavior as well as some artifacts. These need to be thoroughly investigated and removed to prevent any surprises during actual racing.

All in all, the usage of the Delaunay Triangulation for track bound detection seems to be a natural fit as the search over the simplicies provides safe track boundary points, a separation of the track bounds into left and right, as well as the center line directly without intermediate steps.

For future work, an adaptive version that can be run in parallel to the opponent detection may be of interest. If no opponent is detected, the track bounds are immediately published, whereas if certain lidar points need to be removed, the adaptive version could be run again using information from the previous pass.

Appendix A

Task Description

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CENTER FOR PROJECT-BASED LEARNING

Fall Semester 2021

Task Description
Stereo Vision and LiDAR Fusion for Autonomous Car Racing

Semester Thesis

Mike Boss
mboss@student.ethz.ch

March 21, 2022

Advisors: Nicolas Baumann, nicolas.baumann@pbl.ee.ethz.ch
Jonas Kühne, kuehnej@ethz.ch
Dr. Seonyeong Heo, seoheo@ethz.ch

Professor: Prof. Dr. Luca Benini, lbenini@iis.ee.ethz.ch

Handout Date: 23.11.2021
Due Date: 15.03.2022

Project Goals

The F1TENTH Autonomous Vehicle System is a powerful and versatile open-source platform for autonomous systems research and education on a 1:10 scale. The vehicle also defines the baseline for a competition that in the last years has been held during renowned robotics conventions (e.g. IROS, IFAC world congress).

- Initially a **Mapping Round** is allowed, where the race participants are given 5 minutes to map the racing track.
- During the **Time trial - Qualification**, cars are challenged to race as quickly as possible, in a known environment without touching the walls. The fastest lap counts.
- During the **Time trial - Obstacle avoidance**, cars are challenged to race two laps as quickly as possible. The race track is known, but there are obstacles on the track which are not known a priori. Again the fastest round counts.
- During the **Head to head** race, two cars are challenged to race at the same time. The race is won by the car that passes the finish line first.

During the race, one must extract information of the stationary scenes such as the racing track and the obstacles (from **Time trials**), as well as dynamical objects such as the opponent during the **Head to Head** race. These must be represented in a spatial and temporal context relative to the autonomous agent such that one can then apply the necessary control inputs to navigate the car according to the scene information, as described in [1] or [2].

This Computer Vision (CV) thesis primarily aims to perceive and spatially reconstruct the aforementioned relative scene information by utilising camera-LiDAR fusion techniques similar to [3] and [4], by leveraging the high fidelity information of the camera and the geometric information provided by the LiDAR, relative to the car's pose. The secondary aim of the thesis (if time permits), is the temporal context of the perceived data that could be taken into consideration as well, as in [5].

Tasks

The project will be split into phases, as described below:

Phase 1: Foundation (Week 1-2)

1. Preliminary investigation and feasibility study based on current State of the Art (SotA) vision and LiDAR fusion algorithms such as [3] and [4].
2. Familiarisation with the F1TENTH system. I.e. Robot Operating System (ROS) usage and how to integrate the perception stack.

3. Identify metrics that can quantify the system requirements.

Phase 2: Rosbag (Week 3 - 4)

1. Record *rosbag* with the perception system of the car.
2. Read rosbag and convert into data-format suitable for the model.
3. Create dataloader and pre-process data.
4. Record multiple rosbags for training and testing on race track.

Phase 3: Model (Week 5-10)

1. Implement a model capable of handling the data fusion and outputting the spatial road segmentation relative to the car's pose.
2. Train model on training rosbags.
3. Evaluate model performance and adapt if necessary.

Phase 4: Adapt to real car (Week 11 - 12)

1. Adapt pipeline to run on actual rosbag data.
2. Adapt pipeline to run on car directly.
3. Make model architecture more efficient if necessary.

Phase 5: Report and clean-up (Week 13-14)

1. In-field test of system accuracy on live data.
2. Finalize and clean up code for hand in.
3. Write the report and prepare final presentation.

Milestones

By the end of **Phase 1** the following should be completed:

- The student has a solid understanding and can envision the targeted vision-LiDAR fusion Neural Network (NN) architecture.
- Metrics of the system requirements are clear and quantifiable.

By the end of **Phase 2** the following should be completed:

- A dataset has been acquired such that the student can continuously test the algorithms in subsequent phases.

By the end of **Phase 3** the following should be completed:

- An architecture has been implemented and trained on the dataset.

By the end of **Phase 4** the following should be completed:

- The pipeline has been adapted to run on the real car.

By the end of **Phase 5** the following should be completed:

- Final design and in-field test.
- Final report and presentation.

Project Organization

Weekly Report

There will be a weekly report sent by the candidate at the end of every week. The main purpose of this report is to document the project's progress and should be used by the student as a way to communicate any problems that arise during the week.

Final Report and Paper

PDF copies of the report are to be turned in. References will be provided by the supervisors by mail and at the meetings during the whole project.

Bibliography

[1] Y. Cui, R. Chen, W. Chu, L. Chen, D. Tian, and D. Cao, "Deep learning for image and point cloud fusion in autonomous driving: A review," *CoRR*, vol. abs/2004.05224, 2020. [Online]. Available: https://arxiv.org/abs/2004.05224

[2] C. Ventura, M. Bellver, A. Girbau, A. Salvador, F. Marques, and X. Giro-i Nieto, "Rvos: End-to-end recurrent network for video object segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[3] F. Yang, H. Wang, and Z. Jin, "A fusion network for road detection via spatial propagation and spatial transformation," *Pattern Recognition*, vol. 100, p. 107141, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S003132031930442X

[4] S. Gu, Y. Zhang, J. Yang, J. M. Alvarez, and H. Kong, "Two-view fusion based convolutional neural network for urban road detection," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 6144–6149.

[5] Z. Yuan, X. Song, L. Bai, Z. Wang, and W. Ouyang, "Temporal-channel transformer for 3d lidar-based video object detection for autonomous driving," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2021.

Appendix B

Declaration of Originality

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Track Bounds Detection for F1TENTH Autonomous Racing

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Boss	Mike

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
21.03.2022, Zürich	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Bibliography

[1] "F1tenth." [Online]. Available: https://f1tenth.org/

[2] "Iros-race." [Online]. Available: https://www.youtube.com/watch?v=KLOceDeegVs

[3] "Cps-week." [Online]. Available: https://www.youtube.com/watch?v=d5yzKPjikFM

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[5] Wikipedia contributors, "Delaunay triangulation — Wikipedia, the free encyclopedia," 2022, [Online; accessed 18-March-2022]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Delaunay_triangulation&oldid=1069544278

[6] "Formula student." [Online]. Available: https://www.imeche.org/events/formula-student/team-information/fs-ai

[7] "Indy-autonomous-challenge." [Online]. Available: https://www.indyautonomouschallenge.com/

[8] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," *Proceedings of Machine Learning Research*, vol. 123, 2020.

[9] J. Choe, K. Joo, T. Imtiaz, and I. S. Kweon, "Volumetric propagation network: Stereo-lidar fusion for long-range depth estimation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4672–4679, 2021.

[10] F. Yang, H. Wang, and Z. Jin, "A fusion network for road detection via spatial propagation and spatial transformation," *Pattern Recognition*, vol. 100, p. 107141, 2020.

[11] H. Liu, Y. Yao, Z. Sun, X. Li, K. Jia, and Z. Tang, "Road segmentation with image-lidar data fusion," *arXiv preprint arXiv:1905.11559*, 2019.

[12] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde, "Lidar–camera fusion for road detection using fully convolutional neural networks," *Robotics and Autonomous Systems*, vol. 111, pp. 125–131, 2019.

[13] X. Lv, Z. Liu, J. Xin, and N. Zheng, "A novel approach for detecting road based on two-stream fusion fully convolutional network," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1464–1469.

[14] Y. Yenıaydin and K. W. Schmidt, "Sensor fusion of a camera and 2d lidar for lane detection," in *2019 27th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2019, pp. 1–4.

[15] P. Yin, J. Qian, Y. Cao, D. Held, and H. Choset, "Fusionmapping: Learning depth prediction with monocular images and 2d laser scans," *arXiv preprint arXiv:1912.00096*, 2019.

[16] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[17] F. Nobis, J. Betz, L. Hermansdorfer, and M. Lienkamp, "Autonomous racing: A comparison of slam algorithms for large scale outdoor environments," in *Proceedings of the 2019 3rd International Conference on Virtual and Augmented Reality Simulations*, 2019, pp. 82–89.

[18] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, 2017.

[19] B. Delaunay *et al.*, "Sur la sphere vide," *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, vol. 7, no. 793-800, pp. 1–2, 1934.

[20] J. Kabzan, M. I. Valls, V. J. Reijgwart, H. F. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan *et al.*, "Amz driverless: The full autonomous racing system," *Journal of Field Robotics*, vol. 37, no. 7, pp. 1267–1294, 2020.

[21] "Zed 2 - ai stereo camera." [Online]. Available: https://www.stereolabs.com/zed-2/

[22] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[23] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson,

C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[24] The CGAL Project, *CGAL User and Reference Manual*, 5.4 ed. CGAL Editorial Board, 2022. [Online]. Available: https://doc.cgal.org/5.4/Manual/packages.html