Optimal Path Generation for Racing

Tim Lindenau
Munich, 07.02.2023

A

Motivation & Task Description

In racing minimum lap time is the important metric – Three path planning algorithms developed and compared w.r.t to lap time

The challenge: Planning a path that allows for the fastest lap times in autonomous racing

- Most path planning algorithms have **“time” as one of many criteria** (others e.g., fuel efficiency, safety margins …)
- **In racing, minimum lap time is the most important metric**

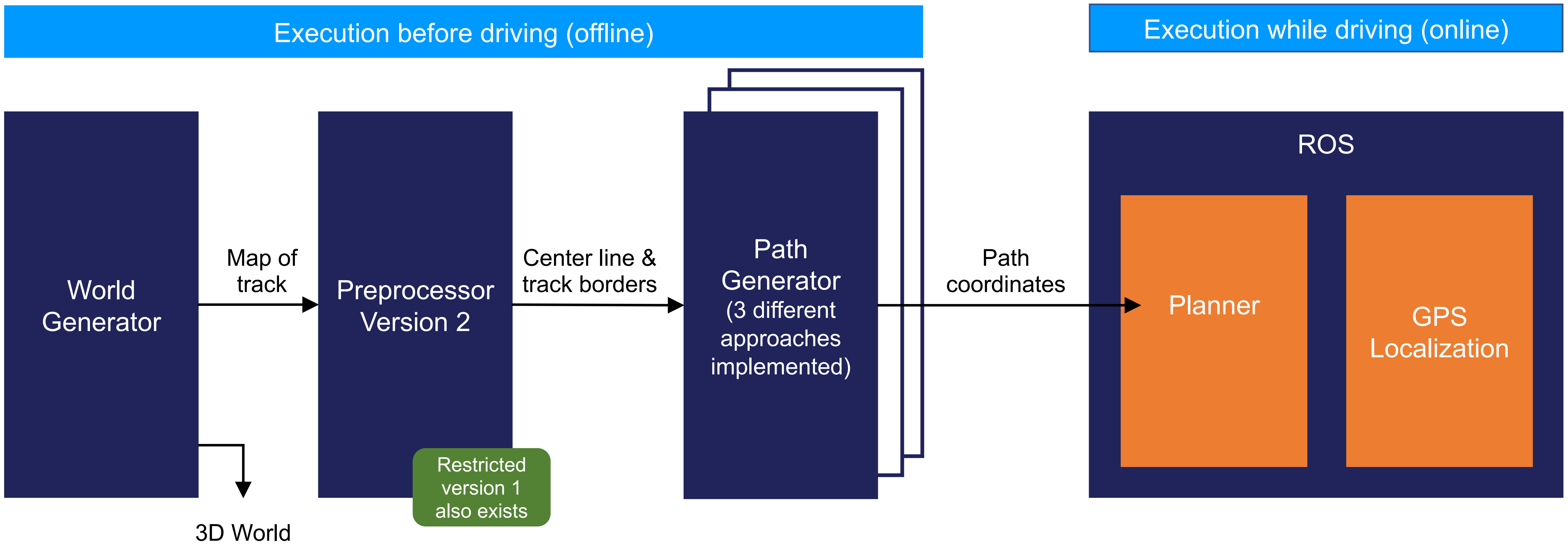
The Task: Development and comparison of different planners to minimize lap time

- Development of different algorithms for generation of fastest racing path
  - I Shortest path (Version a and version b)
  - II Minimum curvature
- Integration of these algorithms into ROS by developing a custom planner plugin
- Comparison of the three approaches in terms of lap time and possible limitations
- Implementation of an optimal MPC controller to follow the generated paths (out of scope due to reduction of team size)

B

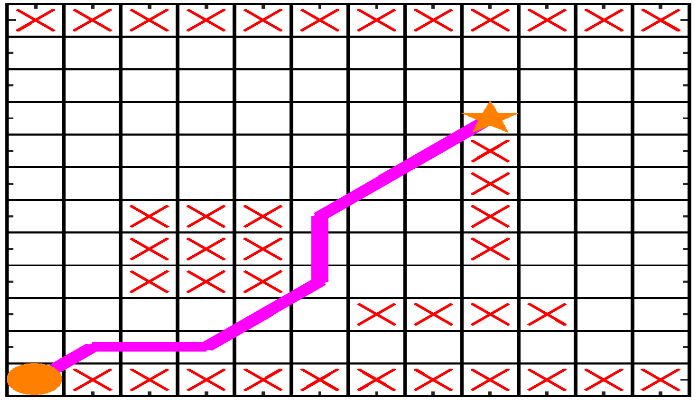
Algorithmic Overview

Software Architecture: All offline and online components developed and implemented within the project



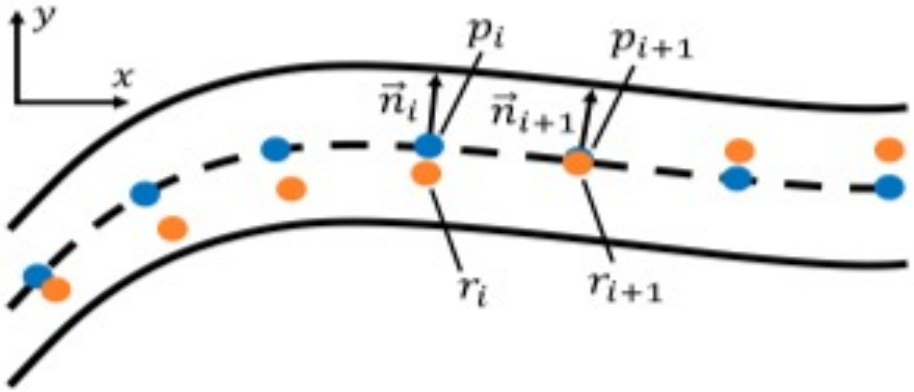
Theory in brief: Three different algorithms to find the fastest path for racing

I a Shortest path: A*



- Intuition: Shortest path is probably one of the fastest
- Efficient grid search algorithm to find minimum distance path
- Edgy paths possible ("Not smooth")

I b Shortest path: Optimization[1]



- Optimization and interpolation guarantee smooth paths
- Move points on centerline (orange) along normal vectors such that overall path length is minimized

II Minimum curvature path[1]

- Inspired by formula 1: Fastest racing path is the one that maximizes cornering speed
- Algorithm similar to I b
- Optimize for minimum curvature not path length

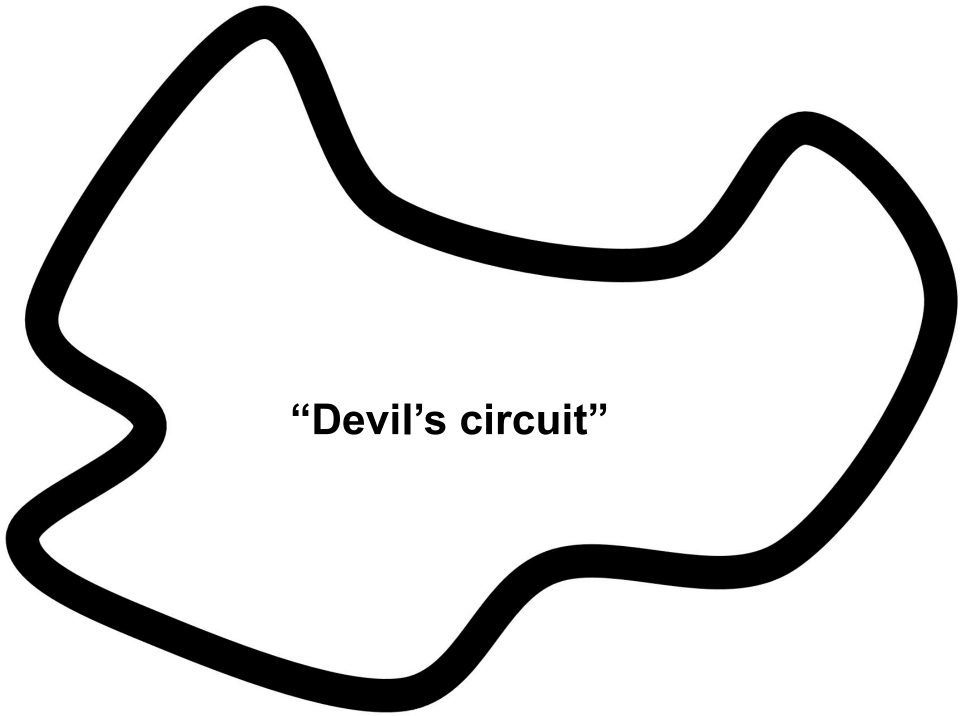
[1]Based on work of Alexander Heilmeier

C

Experiment

Experimental Setup – Comparing the lap time of all three algorithms with the TAS car in ROS and a high performance car in a standalone simulation

Preparation: Create custom designed racetrack (“Devil’s circuit”) **featuring tight and open corners**

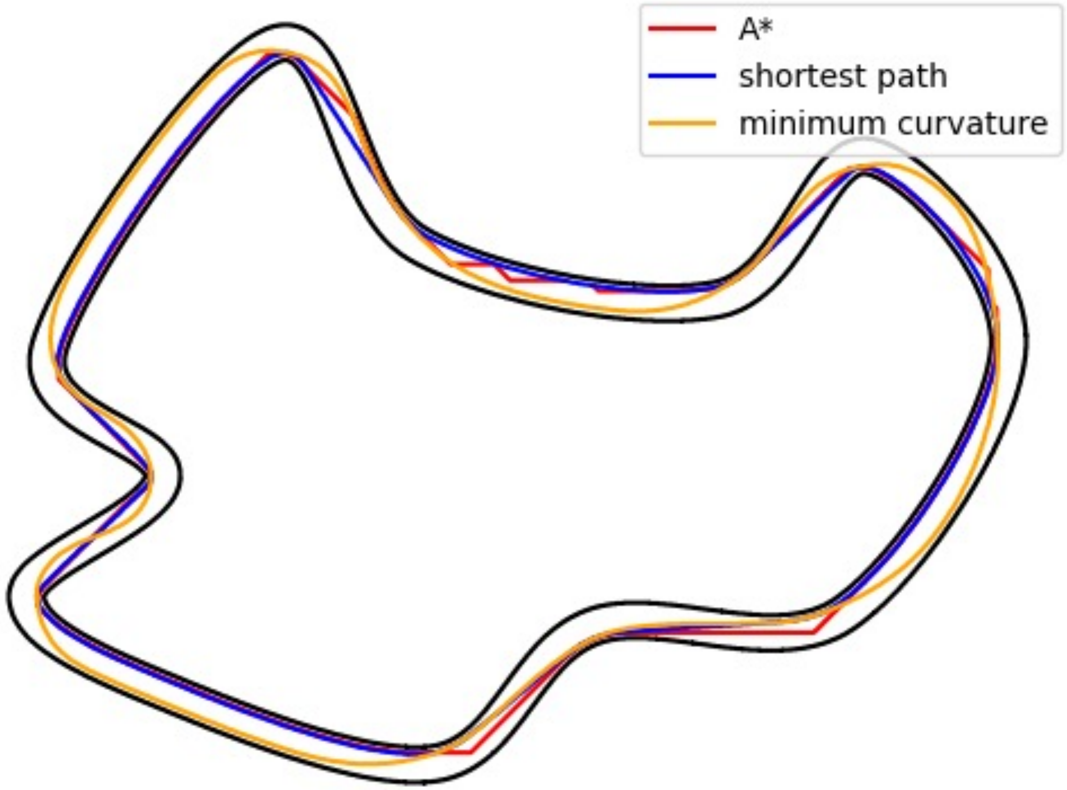


Performance evaluation of the three algorithms in two different environments

- Generate racing paths using all three algorithm
  - **I a** Shortest path: A*
  - **I b** Shortest path: optimization
  - **II** Minimum curvature: optimization
- Compared lap times of the three algorithms
  - **A** Compared lap times using the TAS car in **ROS**
  - **B** Compared lap times using custom simulator with a model of a **high performance model car** (top speed 30 kph)

Comparison of the generated paths – Optimization generated the shortest path at 241m, the minimum curvature path is 10m longer

Generated paths using all three algorithms



Path comparison

I a	**A*** generates short (248 m) but spikey path
I b	**Optimization-based shortest path** performed better. Generated path is shortest (241 m) and smooth
II	**Minimum curvature** generates smooth path where cornering radii are maximized to the detriment of overall path length (251 m)

Experimental results: With a slow car (ROS) the shortest path is the fastest; with a high performance car minimum curvature significantly faster

Comparison of lap times for all three generated paths

		A ROS	B Simulation
I a	**A***	8:05 min	50.3 s
I b	**Shortest Path** (optimization)	**7:55 min**	38.6 s
II	**Minimum Curvature**	8:20 min	**36.0 s**

Simulation by factor 10 faster than ROS

What we learned:

- Optimization always outperforms A* due to smoother and shorter path
- **Tie between "shortest path" and "minimum curvature"**
  - Slow car in ROS profits from shorter distance
  - Fast car in simulation profits from higher cornering speed
- **No approach always better,** the best path always depends on the given vehicle dynamics

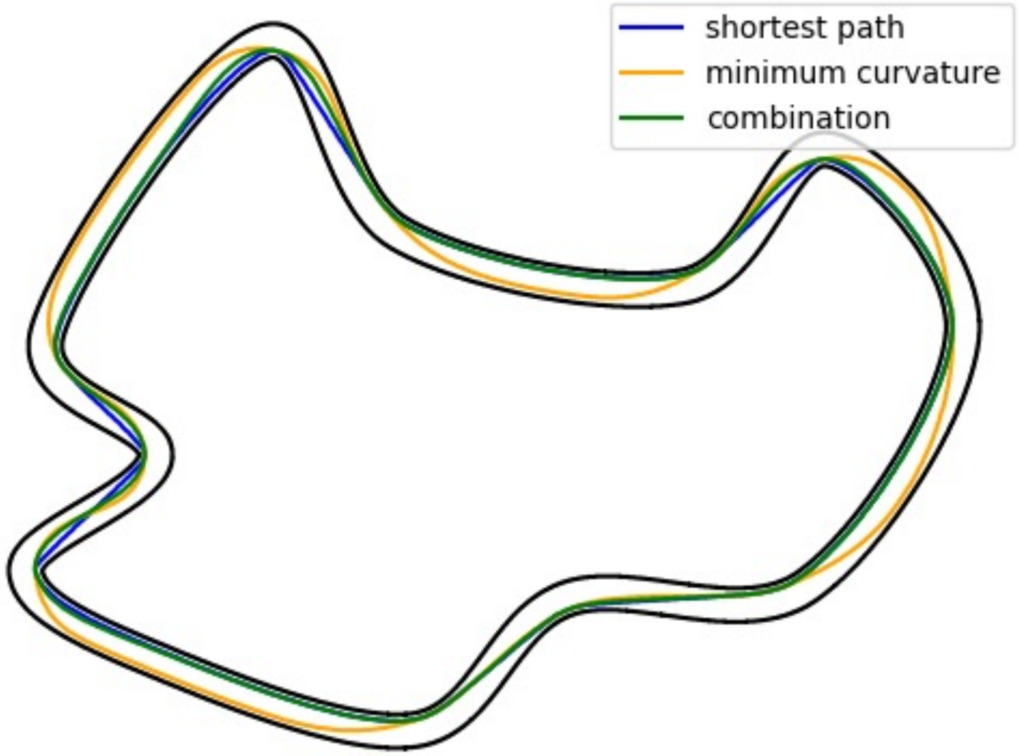
New idea:

- Can we further **reduce lap-time by combining** "shortest path" and "minimum curvature"?

Discussed on next slide

The fastest path is a combination between “Shortest path” and “Minimum curvature”: Lap time reduces to 35,6s (0,4 sec/1% faster)

The fastest path is a combination of shortest- and minimum curvature path



Lap time reduced to 35.6s: 0,4 sec faster (1%)

- Optimal path is a **combination of shortest- and minimum curvature path** depending on vehicle dynamics
- Heilmeier solved the two problems separately
- New approach **solving them together** using a weighted sum of path length and curvature as optimization objective

D

Going Beyond – Using offline generated paths in a dynamically changing environment

The biggest limitation so far – Paths are generated offline and can thus not be used in a dynamically changing environment

- **Biggest issue so far:** Paths are generated offline and cannot be changed at runtime
- Prevents application in dynamic environments like multi agent racing (i.e. multiple cars in the race)

- **"Going beyond":** Overcome this issue with a **new ROS planner** that:
  - Follows offline calculated reference path whenever possible
  - Dynamically recalculates path around objects back onto reference path

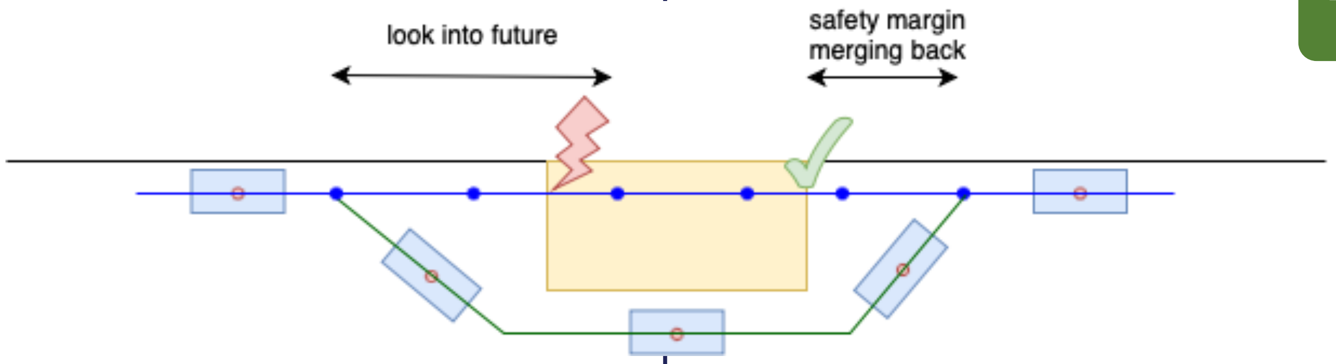
Designed concept/algorithm and developed code to solve the dynamic navigation challenge

Three challenges when implementing the dynamic planner: Collision detection, path generation and efficiency

Challenge 1: Collision detection

Challenge 2: Navigating around object

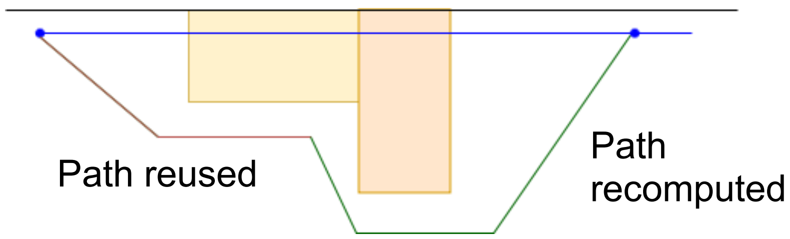
Details see appendix



- Find current position on reference path
- Extend from current position for some meters ahead
- Use laser to check for collisions of extended path

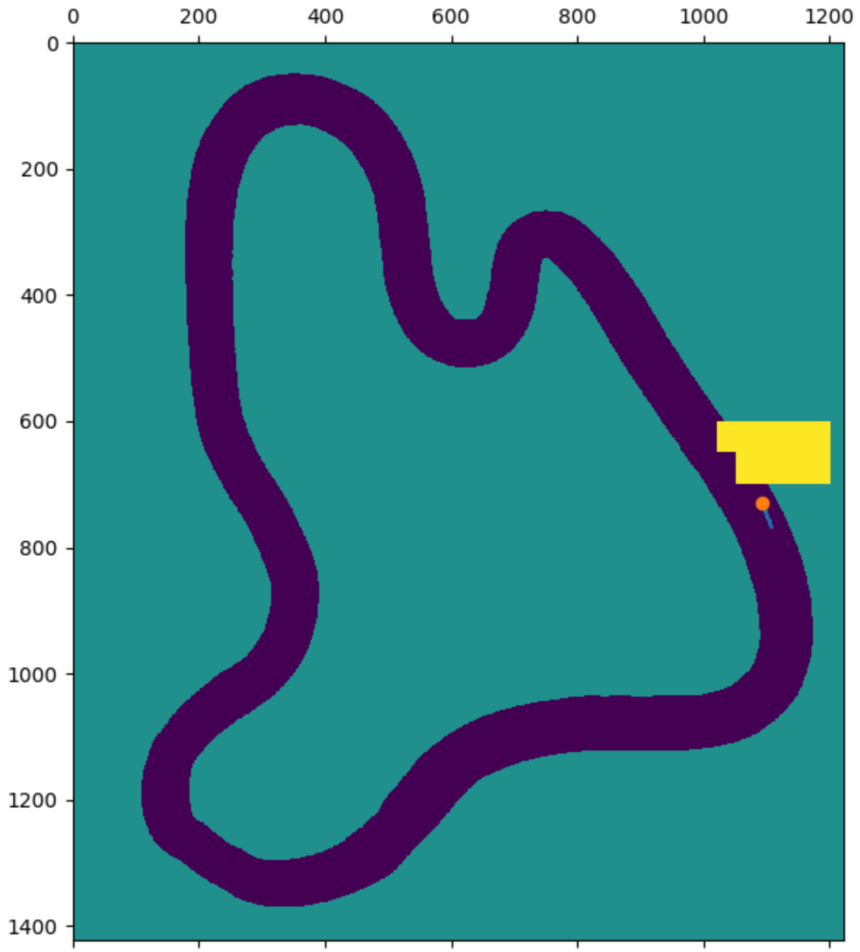
- Identify dimensions of object (requires bird eye view, not fully functional when using the laser)
- Find safe point to merge back
- Generate path around object (modified A*, soft margins and Bezier curve)

Challenge 3: Efficient Implementation



- Algorithm must be efficient and real-time capable
- A* is the bottleneck
- Reuse previously generated paths as much as possible

Animation – Dynamic path planning in action



E

Conclusion

Conclusion

Result 1:

Comparison of shortest path and minimum curvature planners

- **Curvature** and **length** are both important metrics for finding the fastest racing path
- Slow cars profit from shorter path, faster ones from minimizing curvature
- Fastest path is a **combination of both** criteria, with an optimal weighting factor dependent on the vehicle dynamics
- Created new algorithm which outperforms previously existing

Result 2:

Implementation of dynamic planner

- Extended planners from the **offline to the online setting**
- Implemented working **proof of concept**
- ROS implementation not yet fully functional due to the issue of identifying the size of an object (bird eye view)
- Possible Solution: Cavity detection by Dias et. Al.

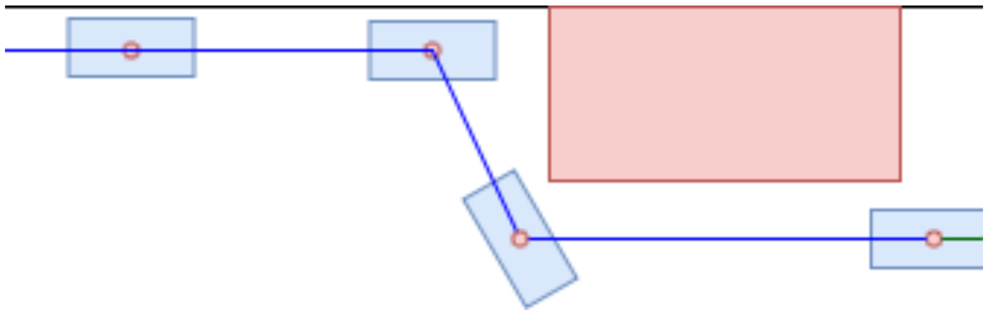
F

Appendix

Challenge 2 in detail - The asymmetric shape of the car requires adaptations to the A* algorithm. New ideas: Soft margins and Bezier curves

- Most vehicles longer than they are wide
- Minimum distance to objects that prevents collision while following a path (width) is not enough for the collision avoidance maneuver
- We cannot simply use A* to find new path around obstacles instead, 2 phase algorithm
- **Phase 1: Navigating around object**
  - A* with large safety margin to prevent collision at front
  - Soft Margin guarantees that we always find a path
- **Phase 2: Smooth merging back**
  - Use Bezier curve to construct path for merging back

Phase 1: Navigation around object



Phase 2: Smooth merging back