Optimizacija trajektorije gibanja autonomnog trkaćeg vozila

Jurić, Marko

Master's thesis / Diplomski rad

2025

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:677559

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-22**

Repository / Repozitorij:

FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repozitory

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS No. 107

OPTIMIZATION OF MOTION TRAJECTORIES FOR AUTONOMOUS RACING VEHICLES

Marko Jurić

Zagreb, February 2025

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS No. 107

OPTIMIZATION OF MOTION TRAJECTORIES FOR AUTONOMOUS RACING VEHICLES

Marko Jurić

Zagreb, February 2025

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Zagreb, 30 September 2024

MASTER THESIS ASSIGNMENT No. 107

Student:	**Marko Jurić (0036527393)**
Study:	Information and Communication Technology
Profile:	Control Systems and Robotics
Mentor:	prof. Ivan Marković, PhD

Title: **Optimization of motion trajectories for autonomous racing vehicles**

Description:

Optimizing the vehicle's trajectory is one of the key tasks for autonomous vehicles, especially racing vehicles, where accuracy and execution speed requirements are high. This topic focuses on the development of an algorithm that optimizes the motion trajectory of an autonomous racing vehicle, which is the last part in the pipeline of an autonomous racing vehicles before the motion control itself. The goal of this thesis is to develop and test trajectory optimization algorithms that can adjust the vehicle's path, maximize speed, minimize energy consumption, and optimize other parameters of interest. The algorithm will use vehicle parameters, tire interaction with the ground surface models, and other characteristics, and the methods will be validated through simulations and compared with existing solutions.

Submission date: 14 February 2025

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 30. rujna 2024.

DIPLOMSKI ZADATAK br. 107

Pristupnik:	**Marko Jurić (0036527393)**
Studij:	Informacijska i komunikacijska tehnologija
Profil:	Automatika i robotika
Mentor:	prof. dr. sc. Ivan Marković

Zadatak: **Optimizacija trajektorije gibanja autonomnog trkaćeg vozila**

Opis zadatka:

Optimizacija trajektorije gibanja vozila jedan je od ključnih zadataka za autonomna vozila, a posebice u slučaju trkaćih vozila, gdje su zahtjevi za točnošću i brzinom izvođenja visoki. Ova tema fokusira se na razvoj algoritma koji optimizira trajektoriju autonomnog trkaćeg vozila što je posljednji dio sustava autonomnog trkaćeg vozila prije samog upravljanja gibanjem. Cilj ovog diplomskog rada je razviti i testirati algoritme za optimizaciju trajektorije koji mogu prilagoditi putanju vozila, maksimizirati brzinu, minimizirati potrošnju energije i optimirati ostale parametre od interesa. Algoritam će koristiti parametre vozila, modeliranje interakcije guma s podlogom i slične karakteristike, a metode će biti validirane kroz simulaciju te uspoređene s već postojećim rješenjima.

Rok za predaju rada: 14. veljače 2025.

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS No. 107

OPTIMIZATION OF MOTION TRAJECTORIES FOR AUTONOMOUS RACING VEHICLES

Marko Jurić

Zagreb, February, 2025

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 107

OPTIMIZACIJA TRAJEKTORIJE GIBANJA AUTONOMNOG TRKAĆEG VOZILA

Marko Jurić

Zagreb, veljača, 2025.

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Zagreb, 30 September 2024

MASTER THESIS ASSIGNMENT No. 107

Student:	**Marko Jurić (0036527393)**
Study:	Information and Communication Technology
Profile:	Control Systems and Robotics
Mentor:	prof. Ivan Marković, PhD

Title: **Optimization of motion trajectories for autonomous racing vehicles**

Description:

Optimizing the vehicle's trajectory is one of the key tasks for autonomous vehicles, especially racing vehicles, where accuracy and execution speed requirements are high. This topic focuses on the development of an algorithm that optimizes the motion trajectory of an autonomous racing vehicle, which is the last part in the pipeline of an autonomous racing vehicles before the motion control itself. The goal of this thesis is to develop and test trajectory optimization algorithms that can adjust the vehicle's path, maximize speed, minimize energy consumption, and optimize other parameters of interest. The algorithm will use vehicle parameters, tire interaction with the ground surface models, and other characteristics, and the methods will be validated through simulations and compared with existing solutions.

Submission date: 14 February 2025

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 30. rujna 2024.

DIPLOMSKI ZADATAK br. 107

Pristupnik:	**Marko Jurić (0036527393)**
Studij:	Informacijska i komunikacijska tehnologija
Profil:	Automatika i robotika
Mentor:	prof. dr. sc. Ivan Marković

Zadatak: **Optimizacija trajektorije gibanja autonomnog trkaćeg vozila**

Opis zadatka:

Optimizacija trajektorije gibanja vozila jedan je od ključnih zadataka za autonomna vozila, a posebice u slučaju trkaćih vozila, gdje su zahtjevi za točnošću i brzinom izvođenja visoki. Ova tema fokusira se na razvoj algoritma koji optimizira trajektoriju autonomnog trkaćeg vozila što je posljednji dio sustava autonomnog trkaćeg vozila prije samog upravljanja gibanjem. Cilj ovog diplomskog rada je razviti i testirati algoritme za optimizaciju trajektorije koji mogu prilagoditi putanju vozila, maksimizirati brzinu, minimizirati potrošnju energije i optimirati ostale parametre od interesa. Algoritam će koristiti parametre vozila, modeliranje interakcije guma s podlogom i slične karakteristike, a metode će biti validirane kroz simulaciju te uspoređene s već postojećim rješenjima.

Rok za predaju rada: 14. veljače 2025.

I would first like to thank my family, especially my parents, sister, grandma, uncle, and aunt for their great emotional and financial support throughout my studies.

I extend my gratitude to my mentor, Prof. Dr. Ivan Marković, and the study assistant, Vlaho-Josip Štironja for their help in executing tasks and practical advices.

Also, I would like to thank my colleagues from the FSB Racing Team. Especially the Autonomous module and Vehicle Dynamics module for their time, advice, and all necessary information and data. Additionally, I appreciate the help from my work colleagues who helped me define the vehicle model.

Finally, big thanks to my friends I met through college and my dear girlfriend.

Contents

1 Introduction **3**

1.1 Formula Student competition 3

1.2 Motivation and context 4

1.3 Autonomous system pipeline 5

2 Theoretical background **7**

2.1 Path planning and trajectory optimization 7

2.1.1 Graph based 7

2.1.2 Sample based 7

2.1.3 Optimization based 8

2.1.4 Receding horizons 8

2.2 Path geometry 9

2.2.1 Spline generation 9

2.3 Optimization method 10

2.4 Velocity profiles 13

2.5 Vehicle model 14

2.5.1 DoF in the CoG 14

2.5.2 Dynamic bicycle model 14

2.5.3 Dynamic two-track model 15

2.6 Summary 16

3 Model formulation and resolution **17**

3.1 Vehicle parameters 17

3.2 Vehicle model implementation 18

3.2.1 Friction constraint 18

3.2.2 Engine constraint 21
3.3 Velocity profile 23
3.3.1 First-pass 23
3.3.2 Forward integration 24
3.3.3 Backward integration 25
3.4 Path representation 26
3.5 Objective functions 28
3.6 Escaping the direct optimization method 31

4 Implementation analysis 34
4.1 FSG track example 34
4.2 Output data for MPC 38

5 Results 40

6 Conclusion 42
6.1 Achievements 42
6.2 Future work and improvements 43

References 45

Abstract 48

Sažetak 49

1 Introduction

Race trajectory optimization is an essential part of any racing team whether they use an autonomous system or not. Creativity and innovation enable teams to gain an edge over opponents, and trajectory optimization has the advantage of being extremely flexible with a wide number of different approaches that will be discussed.

Nevertheless, a well-optimized trajectory has to undergo a number of vehicle and track constraints to keep the vehicle within the feasible state space. It is done using the engine and tire parameters, proportions of the vehicle, iterative sensor, mass and other constants calculated through simulation.

The goal of this thesis is to develop a trajectory optimization prototype that processes artificial track inputs from SLAM (*Simultaneous Localization and Mapping*) and generates an optimized path and velocity profile for use in the autonomous vehicle's control algorithms.

In addition, prototype has a purpose to prove the chosen approach while keeping the development curve sharp. Latter development heading is to optimize the computational load and integrate it inside the current autonomous pipeline.

1.1 Formula Student competition

Formula Student competition is an engineering competition where teams present concepts, structural solutions and build one-seater cars in accordance with Formula Student rulebook. Competitions have three categories: CV (internal combustion engine), EV (electric vehicle) and DV (*Driverless*), which includes autonomous cars regardless of the type of propulsion.

Each discipline consists of two types of event categories: static and dynamic. In this work the focus will be on the dynamic events of DV category that test the vehicle and autonomous system capabilities, performance, robustness and security. Dynamic events are listed:

- **Acceleration** in which the car from a stationary state must travel a distance of 75 meters and thus his goal is to test the time needed for acceleration.
- **Skidpad** which serves to check the maneuverability, and the track is in the shape of number *eight* where it is necessary to make two left and two right circles.
- **Autocross** which serves to test the effectiveness and computational performance of autonomous software over two sets of one lap as well as mechanical capabilities of a car.
- **Trackdrive** is the the most complex discipline that tests the performance and endurance of the autonomous vehicle. It consists of one run with 10 laps. Because of its complexity it is rewarded with the most points.

Trajectory optimization ensures vehicle will not exceed track limits in any of the given events and tries to produce the fastest trajectory while making sure the vehicle constraints are met. Besides that, algorithm shall converge and adapt the trajectory when the track input changes in reasonable time so the vehicle can actually use its benefits throughout majority of the event.

1.2 Motivation and context

Motorsport is a leading inovator in the automotive industry, from mechanical and control achievements such as all-wheel drive or active suspension to fuel and battery efficiency [1]. Recently, autonomous driving competitions such is *Formula Student Driverless* are also creating, using and testing newest technologies that are yet to be used in the future of automotive industry.

Furthermore, the nature of track is simplified to be a series of blue and yellow closed-loop cones. Yellow cones are right edge of the circuit while blue cones are left edge of the circuit. Rules and regulations are simple: get as best time as possible while hitting as little

cones possible. In this simplified environment there is a huge space to test incredibely complex mapping, sensor fusion, control, perception and other systems and algorithms. That enables transfer and adoption of new innovations into a real, complex environment such is traffic on the road.

Thesis is based upon the configuring the team's EV car - *Vulpes* to be a prototype version of autonomous vehicle. It's purpose is to be the proof of concept for latter integration of EV and DV car into one to compete on future competitions.

Currently, optimization is done using the minimum time optimization implementation [2] which is proven to be computationally inefficient with average execution time of $15s$ for basic Formula Student Driverless track. Although the actual execution time is low for the direct optimization, vehicle model is on the simpler side. Goal is to find computationaly less expensive method and using all vehicle and tire parameters available while sacrificing some performance. Heuristic optimization methods are proposed as the potential solution.

Figure 1.1: FSB Racing Team's first EV car adjusted for Driverless category - *VulpesD*.

1.3 Autonomous system pipeline

The Autonomous system pipeline consists of three major components based of the work of multiple Driverless Cup winners AMZ Driverless Team [3]:

- **Perception** Its goal is to perceive cones and estimate their color and position as well as the associated uncertainties.

- **Motion estimation and Mapping** Its goal is to map the track, detect the boundaries and estimate the physical state of the car.

- **Control** Its goal is to safely operate the car within the limits of handling while minimizing lap time. Taking the track boundaries and the state estimates as inputs, the control actions (steering, throttle and brake) are computed.

Inside the FSB Racing Team, the pipeline is adjusted and divided into more sections. Parts of pipeline are specific to team's hardware and optimal approach based on budget, experience, development time and knowledge:

- **Odometry** - Fusing multiple sensors such as vehicle encoders, GPS, IMU (*Inertial Measurement Unit*) including a magnetometer, accelerometer and gyroscope, to define vehicle odometry model.

- **Perception** - Same use case as mentioned above consisting of two perception sensors: LiDAR and stereo camera detecting and extracting landmarks (blue and yellow cones) from its measurements.

- **Localization and mapping** - Integration of odometry output with perception output to iteratively update the current vehicle location and the landmark map, implemented through FASTSLAM 2.0 [4] algorithm which is based on Extended Kallman filter and particle filter.

- **Path planning and Trajectory optimization** - Finding the central path of the track based on SLAM track map. After that, optimizing the trajectory while keeping the velocity profile in bounds of vehicle and track constraints.

- **Control** - Control of throttle, braking and steering to follow given trajectory implemented through MPC (*Model Predictive Control*) described in following thesis [5] [6].

2 Theoretical background

This chapter discusses different approaches to path planning and trajectory optimization in racing domain. In addition, it will describe mathematical optimizer, track and geometry approaches to formulate the objective function and constraints for minimization problem. Furthermore, vehicle models are discussed regarding the available data.

2.1 Path planning and trajectory optimization

This section will take a look into most used approaches for optimizing the race trajectory. This is the central point for picking the suitable algorithm for the given problem construct.

2.1.1 Graph based

Based on generating a search space (directed graph) that satisfies the differential constraints of the graph creation. Graph resembles state lattices that are connected based on a certain heuristic i. e. minimal path and constraints. Constraints could be obstacles (cones) or vehicle limits. Graph creation is performed offline, taking approximately 4.5 seconds, with a parallelized search requiring around 20 milliseconds. The approach is best suited for vehicles with limited turning capabilities, such as truck with trailers in narrow environments. [7]

2.1.2 Sample based

Sample based algorithms revolve around generating random or heuristic trajectoires that are then evaluated based on the cost function. Typical implementations would be Ant-Colony algorithm, Evolutionary algorithms or RRT (*Rapidly-exploring Random Trees*). This approach is not suitable when fast computation is required such is the nature of

Formula Student dynamic events. However, it is proven that small enhancements and disregard of unfavorable nodes, while dividing the problem for each corner and later connecting, can improve computation time significantly. [8]

2.1.3 Optimization based

The optimization methods revolve around minimizing or maximizing the cost function within given parameter bounds. Widely spread solution is MPC (*Model Predictive Control*) which integrates optimization with control outputs because of integrated constraint values of the vehicle within. Besides that, there are solutions such as minimizing the curvature, minimizing the path length, maximizing corner exit velocity or the direct method - lap time based optimization. Minimizing curvature and maximizing the corner exit velocity [9] are approaches that use general meta heuristics that vehicle and track layouts have - velocity is proportional to square root of a corner radius or corner exit velocity gives the highest average velocity through following straight. Here are the examples of some cost functions, firstly for minimizing the lap time:

$$J = t_f$$

and secondly for maximizing the out of the corner velocity:

$$J = \sqrt{\dot{x}_f + \dot{y}_f}$$

where f denotes the timestamp of interest. In the first denoting the last timestamp of the track and in the second the timestamp of the each corner exit.

2.1.4 Receding horizons

As defined in the article [10] , the optimal motion planning problem is in general hard to solve by directly applying optimal control techniques, since the problem in general is nonconvex due to obstacle-imposed constraints and nonlinear system dynamics. Therefore, approximate methods are proposed combining the trade-off between time duration and other measures such as smoothness of a motion. To solve the complexity of global planning and optimal control of the vehicle, iterative sliding time window is used to future trajectory states and optimal control of it reducing the problem space.

2.2 Path geometry

Firstly, common approach is to use arcs and clothoids where turns are defined as two concentric circle arcs and clothoids on the start and exit of a corner with straights defined as two parallel lines. On the other hand, spline generation from control points is a better alternative. Between the two, the approach of spline generation is more suitable for Formula Student tracks because of arcs and clothoids approach inability to describe well corners of variable radius that tracks have a plenty of.

2.2.1 Spline generation

The problem formulation begins with the interpolation of track edges. After that, connected interpolated points represent the mid path. To escape the sharp edges and discontinuities in curvature, forming cubic splines is proposed. When the cubic spline along the path is formed, path is discretized to the desired interval. [11]

Formally, discretization interval is inversely proportional to the number of parameters that will be optimized, meaning that denser discretization will produce more precise trajectory estimation with the cost of higher computing complexity.

Furthermore, convenient way to store the given path is in this form: left and right track edges - A and B coordinates (x, y) with α value that is in range $[0-1]$. Parameter α represents the ratio where the midpoint lays on the $\overrightarrow{AB}$ vector. Next step is optimization function that will optimize those alpha values bounded by $[0-1]$ interval. Now, w_{in} is the ratio from mid to left part of the circuit and w_{out} ratio from mid to right part of the circuit.

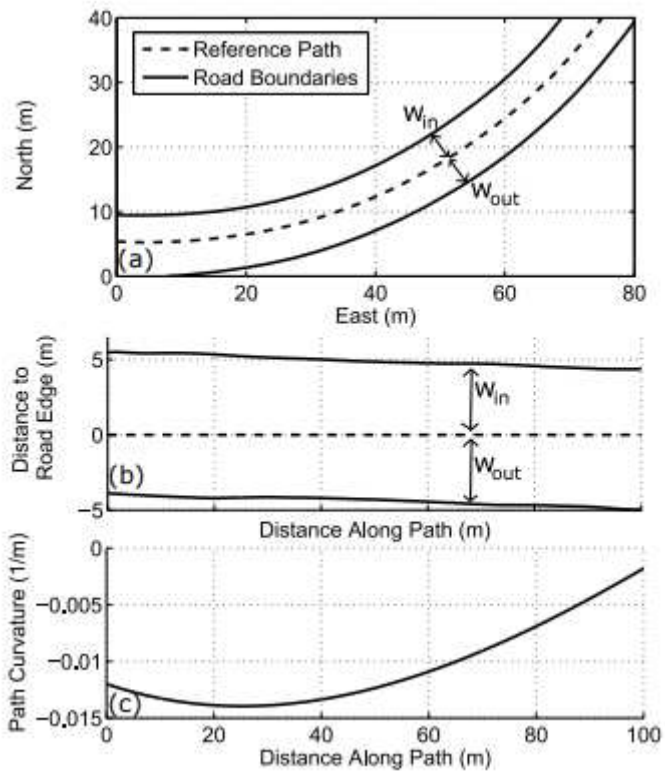


Figure 2.1: Midline representation with smooth curvature from spline generation. [11]

2.3 Optimization method

Calculating the solution in closed form is computationally infeasible, so some other approaches are necessary. The choice of the appropriate optimization method depends on the specific problem and its formulation. Problem formulation starts with the spline generation method. It generates cubic splines that are third-order functions [12]:

$$g_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \quad i = 0, 1, \dots, n - 1 \tag{2.1}$$

where a_i, b_i, c_i, d_i denote the polynomial coefficients.

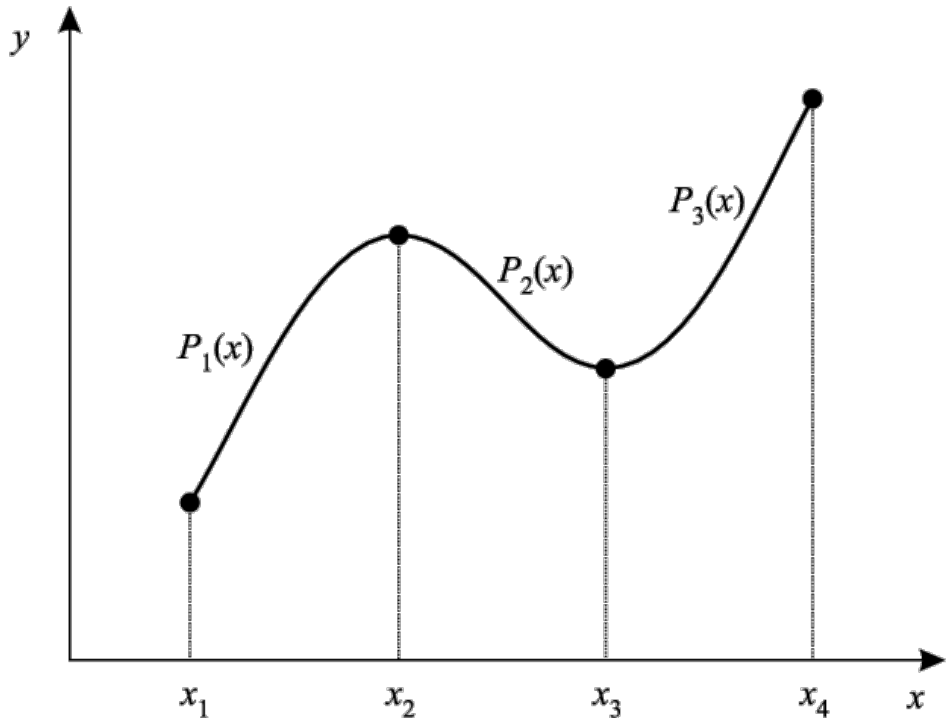


Figure 2.2: Cubic splines with 4 points.

To achieve a smooth interpolation we impose that g(x) and its first and second derivatives are continuous. The number of unknown parameters is $4n$, resulting in need for twelve conditions. The First 6 equations are gotten from spline continuity condition:

$$g_i(x_i) = y_i, \quad i = 0, 1, \dots, n-1 \tag{2.2}$$

$$g_i(x_{i+1}) = y_{i+1}, \quad i = 0, 1, \dots, n-1 \tag{2.3}$$

We are equalizing the polynomial function at each point to be the same as the y value of the point. Four more conditions are retrieved with the assumption that interior points the first- and second-order derivatives of a spline need to match. Now, the first- and the second-order derivatives of a spline in the i and $i+1$ function need to match. It ends up with these two conditions also being met:

$$g_i'(x_{i+1}) = g_{i+1}'(x_{i+1}), \quad i = 0, 1, \dots, n-2 \tag{2.4}$$

$$g_i''(x_{i+1}) = g_{i+1}''(x_{i+1}), \quad i = 0, 1, \dots, n-2 \tag{2.5}$$

This ensures that curvature is continuous - path does not have any sharp edges. Two more conditions are missing to solve this linear equation problem and they are named clamped curvature conditions:

$$g_0'(x_0) = \alpha \tag{2.6}$$

$$g_{n-1}'(x_n) = \beta \tag{2.7}$$

These constraints determine that the first and the last value first-order derivatives should be known beforehand. Values are calculated by iterating this generation process on all control points of the spline.

From this equation excerpt it is evident that concatenated cubic splines are not producing exclusively convex functions throughout its length leading to a conclusion that standard gradient descent method will probably not lead to a global minimum.

Candidate optimization functions are derivatives of Newton's method which is based on approximating the current point of objective function with quadratic one using the second-order Taylor's approximation:

$$g(x) = \frac{1}{2}(x-q)^T A(x-q) + (x-q)^T \nabla f(q) + f(q), \tag{2.8}$$

where A stands for Hessian matrix. Hessian matrix is consisted of the second-order derivatives. The matrix's order depends on the input size parameters which, in this case, are control points directly dependent on the number of landmark pairs (cones). Directly computing the Hessian matrix in each optimization step is a computationally expensive task. Besides that, Hessian might not be positive semidefinite matrix, meaning the quadratic approximation might not have a minimum value leading the algorithm to fail.

Firstly, the problem of time-consuming direct calculation of Hessian matrix is solved by approximating the Hessian matrix. Now, previous equation 2.8 switches to this one:

$$g(x) = \frac{1}{2}(x-q)^T B(x-q) + (x-q)^T \nabla f(q) + f(q), \tag{2.9}$$

with B being the approximation matrix.

Secondly, the problem of that matrix not being semidefinite is solved using the BFGS (*Broyden–Fletcher–Goldfarb–Shanno*) algorithm [13] where B meets three constraints making it semidefinite. Explanation of those constraints is out of the scope of this thesis.

To conclude, L-BFGS-B version of algorithm will be used to solve the minimization problem which is just the memory friendly version. It erases old B matrices from the memory that were used many iterations before the current step.

Still, it should be noted that this is a gradient descent method used on a nonconvex and it might not converge to a global minimum but a local one. This optimization algorithm ensures that in the each step gradients are descending.

2.4 Velocity profiles

After defining the path geometry and optimization methods of this problem, velocity profiles can be discussed. Velocity profiling is determining maximum speed the vehicle can have in a path segment. It should be clear that separating the path optimization formulation from velocity profiling an old, time and memory optimization tool is used under the hood: *"Divide et impera"*. Giving intuitive cost functions to optimization problem that are already proven approaches in the real world motorsport solves the huge complexity problem. Unlike this approach, lap time-based optimization directly uses time as minimization factor which leads to longer computation time because of two interdependent optimization problems.

In 2008, for the first time is proposed the *three-pass* process for generating velocity profiles. [14]

- **First pass** - Determine maximum velocity based solely on corner radius r, theroretical maximum vehicle velocity v_{max} and tire friction coefficient μ_y.
- **Forward integration** - Determine the maximum acceleration limits at the given moment.
- **Backward integration** - Determine the maximum deceleration limits.

The first-pass velocity output is forwarded to the second and the third filters and the

final output is derived from this equation:

$$v_{out} = \min(v_{\text{acclim}}, v_{\text{declim}}) \tag{2.10}$$

This method iterates through the starting point of the circuit to the last forming the loop of connected velocities for each point of discretized interval. Each filter is based on the vehicle model constraints.

2.5 Vehicle model

Lastly, vehicle model's purpose is to describe the vehicle's design and behavior. Models can vary in complexity. How complex model will be depends on the available vehicle parameters, vehicle's state data, desired computing efficiency and problem's need for precision.

Roughly, models can be divided based on the DOF (*Degree of freedom*) position and number in vehicle dynamics system, thereunto some parameters can be defined at equilibrium state where the system is at rest. For instance, slip angle of the tire can be an optimal one derived from simulations. That assumption will be used throughout friction coefficient data extraction for this vehicle model.

2.5.1 DoF in the CoG

The simplest way to describe the vehicle with as little data as possible is to center the whole vehicle mass vehicle inside the CoG (*Center of gravity*). This way slip angle, tire model, lateral and longitudinal load transfer are neglected. Modeling revolves solely around vehicle mass m, longitudinal acceleration a_x, lateral acceleration a_y and fixed friction coefficient μ.

2.5.2 Dynamic bicycle model

The bicycle model simplifies vehicle's axles by combining two wheels into single midpoint. Steering rotates the front axle of the model giving the steering angle δ. When steering angle and wheelbase L is known the corner radius R can be calculated:

$$R = L/tan(\delta). \tag{2.11}$$

The slip angle can be obtained using simple geometry. The angle between the R vector and the vector from the rear axle to the center of rotation provides the same $delta$, unless the distance from the rear axle to the center of gravity l_r is known. Angle between rear axle to center of rotation vector and CoG to center of rotation vector is named slip angle β.

$$tan(\beta) = l_r/(L/tan(\delta)) \tag{2.12}$$

With input parameters being longitudinal velocity and steering angle, slip angle can be obtained using just simple geometry of the simplified model.

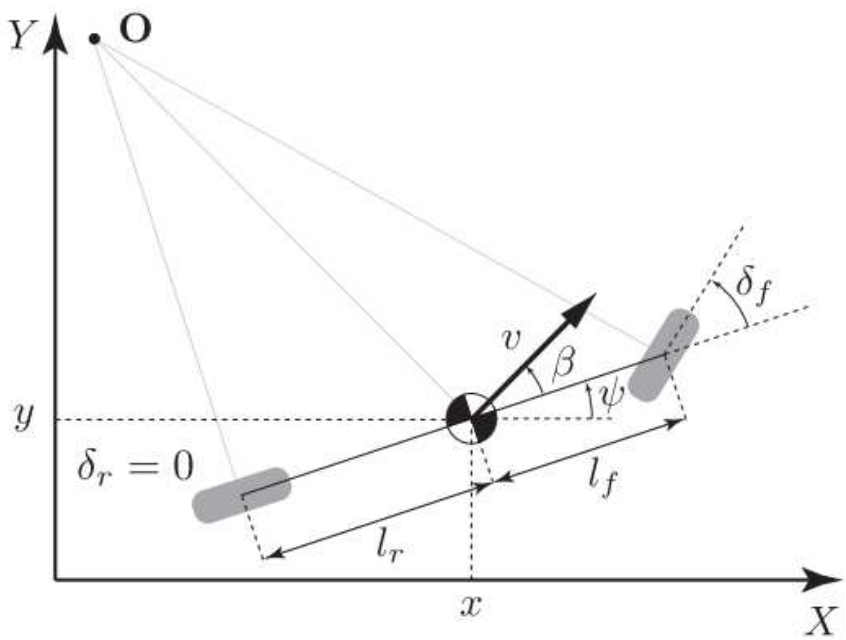


Figure 2.3: Bicycle model geometry. [15]

2.5.3 Dynamic two-track model

The dynamic two-track model can most precisely describe the vehicle. It can obtain vertical forces information, consequently calculating the load transfer with lateral and longitudinal forces of the each wheel. Combining this model with a good tire model

produces valid constraint estimate.

The model will be described in more detail in chapter 3.

2.6 Summary

This chapter looked into available solutions in each technical crossroad and argued which approach should be used to solve the thesis' problem.

In summary, the problem is formulated around finding the heuristic approach to optimize trajectory instead of direct lap-time optimization. There was a quick overview of common path planners where optimization-based methods were the most suitable ones. After that, spline generation and storage are explained in detail. For the given spline construct, adequate optimization is proposed: BFGS method. In spite of non-convex minimization object function, method is able to find the global minimum. Next, the velocity profiling is defined as the separate problem from path planning when direct method is not used which reduces the time complexity significantly. Lastly, the dynamic two-track model is proposed, which is the best description of the vehicle based on the available data.

3 Model formulation and resolution

This chapter will explain how each each part of the trajectory optimization process is formulated, implemented and connected. Firstly, it will describe the vehicle model with its constraints in depth. Next, the connection between vehicle model and velocity profile will be explained. For the profile to have a basis for concatenation, the exact implementation of spline generation will be explained. After that, the optimization problem will be formulated with candidate objective functions. Lastly, the solution to the high time complexity of optimization will be proposed with offline parameter estimation.

Complete model is implemented using the *Python* programming language with some practical libraries such as *numpy* and *scipy*. It is used for quick prototyping and to confirm algorithm suitability and correctness. The idea is to later implement the whole algorithm in *C++* language and wrap it with ROS (*Robot Operating System*).

3.1 Vehicle parameters

Firstly, it is appropriate to list all parameters used in the vehicle model in one place.

Symbol	Description	Value
m	Vehicle mass without the driver	206 kg
ρ	Air density	1.225 kg/m^3
g	Gravity constant	9.81 m/s^2
I_{zz}	Moment inertia around z-axis	116.0175 kg/m^2
t_f, t_r	Front track width	1.29 m
t_r	Rear track width	1.24 m
w_B	Wheelbase	1.53 m
l_f	Distance from front axle to center of gravity	0.842 m
l_r	Distance from rear axle to center of gravity	0.689 m
h_{cg}	Center of gravity height	0.327 m
r_w	Wheel radius	0.2286 m
R	Transmission ratio between engine and wheels	8
x	Friction utilization ratio	0.66
$\mu_{x_{rolling}}$	Rolling friction coefficient	0.01
C_d	Aero drag coefficient	1.39
A	Frontal vehicle area	1.285 m^2

Table 3.1: Vehicle parameters.

3.2 Vehicle model implementation

The vehicle model used is the dynamic two-track model already mentioned in section 2.5.3. This section will go into detail about how it is implemented.

3.2.1 Friction constraint

The model is based on lateral and longitudinal load transfer, where the vertical force is estimated at each discretized point of the circuit based on the lateral and longitudinal acceleration. Knowing the vertical force F_z of each maximum theoretical longitudinal

force F_y can be calculated. To calculate it, the tire model must be included, Pacejka's magic formula:

$$F = D \sin\left[C \arctan\left\{B\alpha - E\left(B\alpha - \arctan(B\alpha)\right)\right\}\right] \tag{3.1}$$

where B, C, D and E represent fitting constants and F is a force or moment resulting from a slip angle α. To make vehicle model simpler, there is an assumption made that the vehicle is always driven on the optimal slip ratio for the given vertical force, giving the vehicle maximum longitudinal force available from the tire. Given the steady state condition of slip ratio the vehicle model is put into the equilibrium state for it.

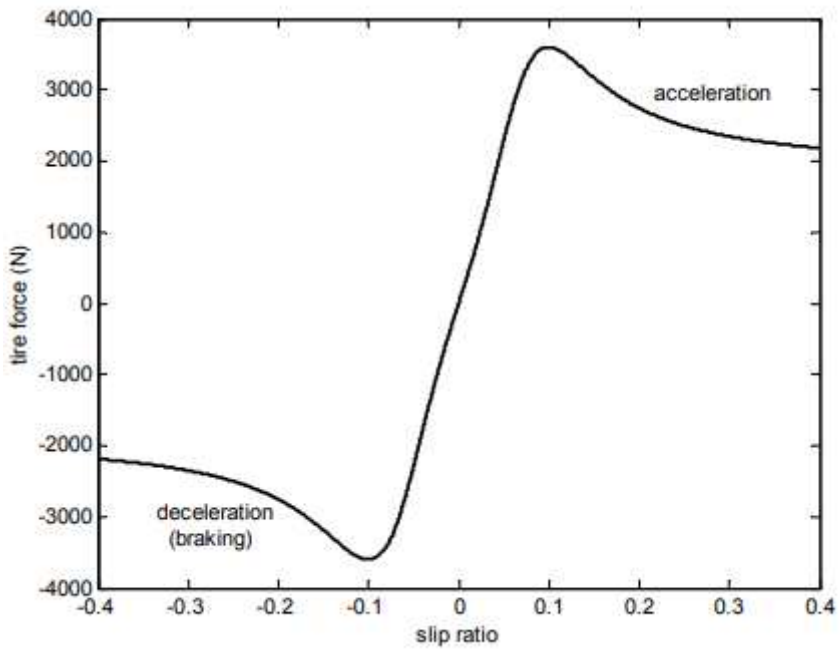


Figure 3.1: Longitudinal force depending on slip ratio. Optimal slip values are maximum and minimum points of the graph. [16]

Slip ratio is a value without a unit of measure:

$$\sigma_x = \frac{r_{\text{eff}}\omega_w - v_x}{v_x}, \tag{3.2}$$

where r_{eff} denotes the effective radius of a wheel and ω_w its angular velocity. In this specific case, it represents the measure of longitudinal slip. In lateral case, slip angle can be derived from this equation:

$$\alpha = \arctan\left(\frac{v_y}{v_x}\right), \tag{3.3}$$

which denotes the angle between the orientation of the tire and the orientation of the velocity vector of the wheel.

The simulations of the tire *Hoosier 18"* longitudinal and lateral friction coefficients μ_x and μ_y) with respect to the vertical force on the tire are estimated with the optimal slip assumption.

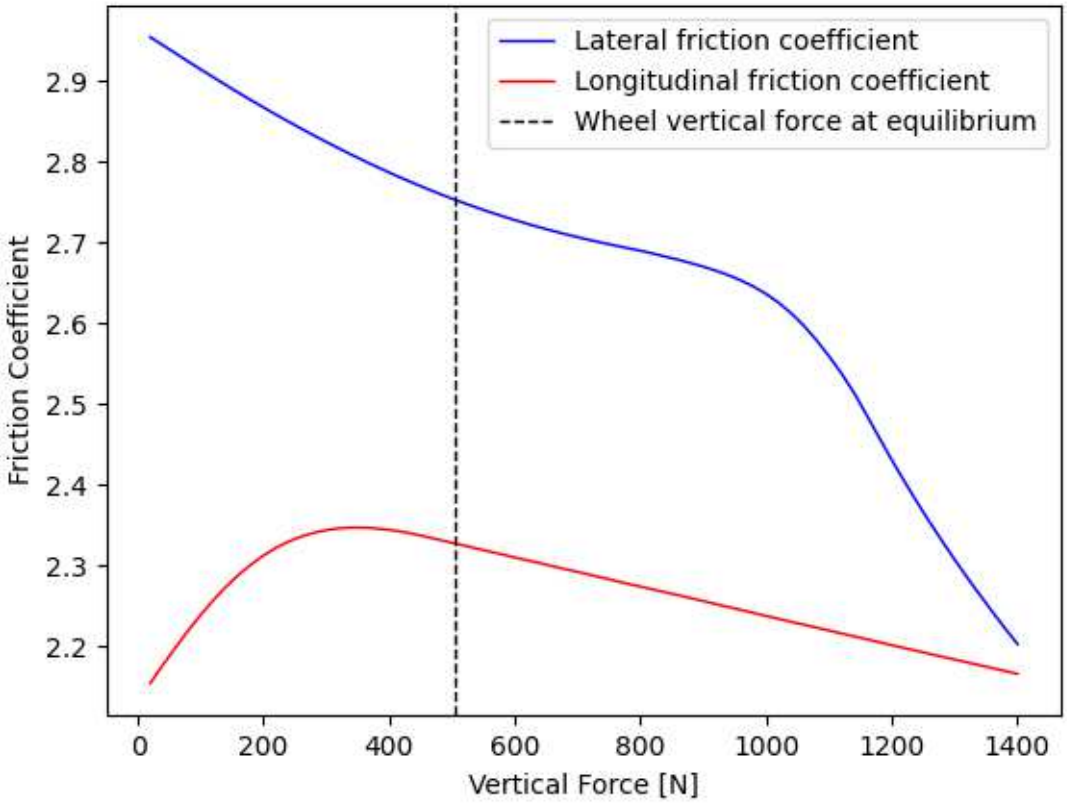


Figure 3.2: Friction coefficients μ_y and μ_x depending on vertical force.

Next, when lateral and longitudinal velocity are available, lateral and longitudinal force transfer can be calculated:

$$F_{z,0} = \frac{F_{z,static}}{4} - \frac{\Delta F_{z,front_lateral}}{2} - \frac{\Delta F_{z,longitudinal}}{2} \quad \text{(Front left)} \tag{3.4}$$

$$F_{z,1} = \frac{F_{z,static}}{4} + \frac{\Delta F_{z,front_lateral}}{2} - \frac{\Delta F_{z,longitudinal}}{2} \quad \text{(Front right)} \tag{3.5}$$

$$F_{z,2} = \frac{F_{z,static}}{4} - \frac{\Delta F_{z,rear_lateral}}{2} + \frac{\Delta F_{z,longitudinal}}{2} \quad \text{(Rear left)} \tag{3.6}$$

$$F_{z,3} = \frac{F_{z,static}}{4} + \frac{\Delta F_{z,rear_lateral}}{2} + \frac{\Delta F_{z,longitudinal}}{,}2 \quad \text{(Rear right)} \tag{3.7}$$

where lateral and longitudinal change of vertical force is:

$$\Delta F_{z,front_lateral} = \frac{m \cdot a_y \cdot h_{cg} \cdot l_r}{w_B \cdot t_f} \tag{3.8}$$

$$\Delta F_{z,rear_lateral} = \frac{m \cdot a_y \cdot h_{cg} \cdot l_f}{w_B \cdot t_r} \tag{3.9}$$

$$\Delta F_{z,longitudinal} = \frac{m \cdot a_x \cdot h_{cg}}{w_B}. \tag{3.10}$$

Each tire has its different vertical forces depending on the state of the vehicle. Vertical forces are calculated in each step to determine the maximum traction the vehicle can produce at that control point. The maximum traction is divided in both lateral and longitudinal orientation and presented as maximum lateral force and longitudinal force of the tire ($F_{y,\max}$ and $F_{x,\max}$).

$$F_{x,\max,i} = mu_x \cdot x \cdot F_{z_i} \tag{3.11}$$

$$F_{y,\max,i} = mu_y \cdot x \cdot F_{z_i} \tag{3.12}$$

where i denotes the wheel index and x friction utilization ratio which is around $\frac{2}{3}$ for the dry asphalt surface. It is necessary to multiply the friction with the ratio because simulations are done on the abrasive sandpaper. With that, first constraint is defined, the tire friction constraint.

3.2.2 Engine constraint

Besides friction constraint, engine configuration and capability need to be modeled. The longitudinal acceleration is limited by the engine torque. It can be derived from the torque curve. It is a graph of engine torque with respect to engine speed.

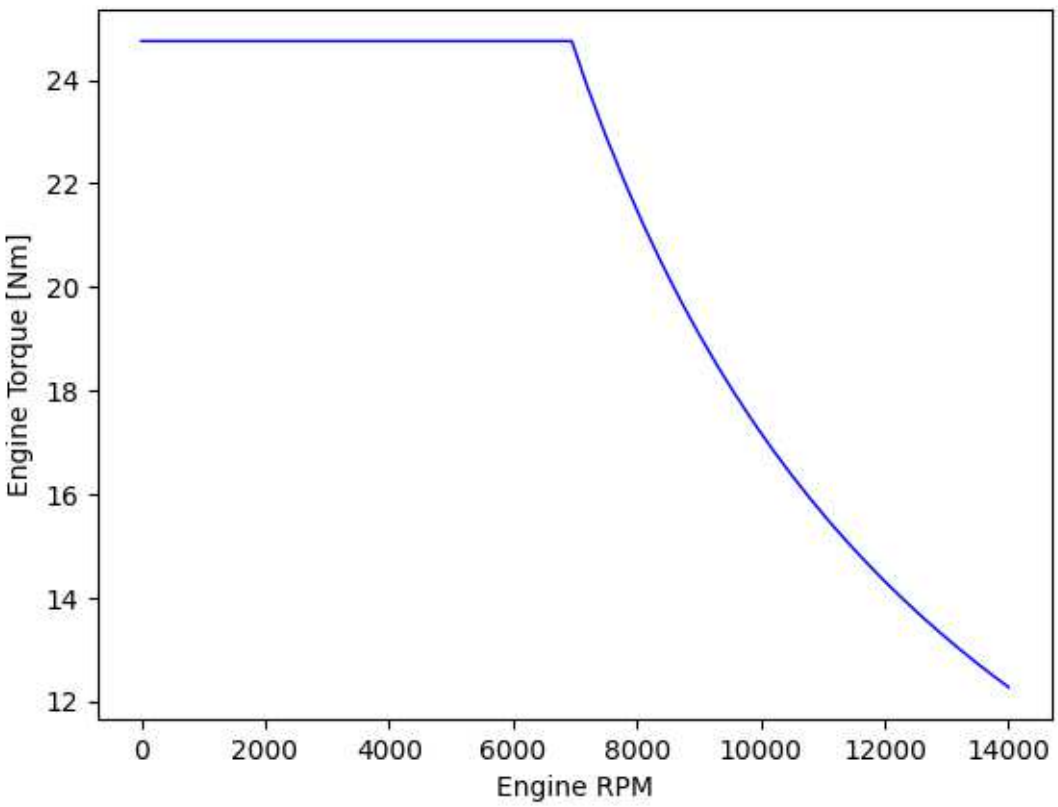


Figure 3.3: Engine torque curve.

To calculate maximum power engine can output for the target velocity, firstly, wheel velocity needs to be calculated:

$$v_{wheel} = \frac{(v_{target} \cdot 60)}{2 \cdot \Pi \cdot r_{wheel}} \tag{3.13}$$

The measurement unit used is rounds per minute. After that, the wheel velocity is used to estimate the engine velocity v_{engine} based on the transmission ratio between its speed and wheel velocity.

$$v_{engine} = R \cdot v_{wheel} \tag{3.14}$$

Now, the longitudinal force can be derived without resistances. Corresponding engine torque is interpolated from the engine torque curve with respect to engine speed which leads to the equation:

$$F_{x,e} = \frac{T_e \cdot R}{r_{wheel}} \tag{3.15}$$

Finally, to achieve the closest estimation of the longitudinal force vehicle can transfer to the ground resistances are included. Air resistance and rolling resistance are the major ones, besides them there are numerous other that are neglected in this case.

$$F_{aero} = \frac{1}{2} \cdot C_d \cdot \rho \cdot A \cdot v_{target}^2 \tag{3.16}$$

$$R_x = \mu_{x_{rolling}} \cdot m \cdot g \tag{3.17}$$

Resistances are subtracted from the 3.15 to calculate the final longitudinal force vehicle can achieve at the given control point.

3.3 Velocity profile

Velocity concatenation to the given path is done using the three-pass filter. To successfully use the filter, the theoretical maximum velocity of the vehicle must first be estimated. That velocity will serve as the upper bound for local velocity calculations, which will be explained in the first pass.

The theoretical maximum velocity is calculated using the engine speed (rpm) at the maximum torque value from the torque-engine speed curve. Final equations are:

$$v_w = \frac{v_{e_{max}}}{R} \tag{3.18}$$

$$v_{x_{max}} = \frac{v_w \cdot 2 \cdot \pi \cdot r_w}{60} \tag{3.19}$$

The maximum theoretical longitudinal velocity is then fed into the first-pass filter.

3.3.1 First-pass

First-pass filter focuses on the lateral tire limit based on the curve radius. For each discretized path segment, there is also a curve's radius. Assuming that lateral force is equal to the friction force:

$$F_{lat} = F_{friction} \rightarrow \frac{mv^2}{r} = \mu_y ng \rightarrow v = \sqrt{r\mu_y g}, \tag{3.20}$$

local velocity can be derived and it depends solely on the lateral friction coefficient and radius. To find the adequate μ_y the minimum one is used from the load transfer already explained in section 3.2.1.

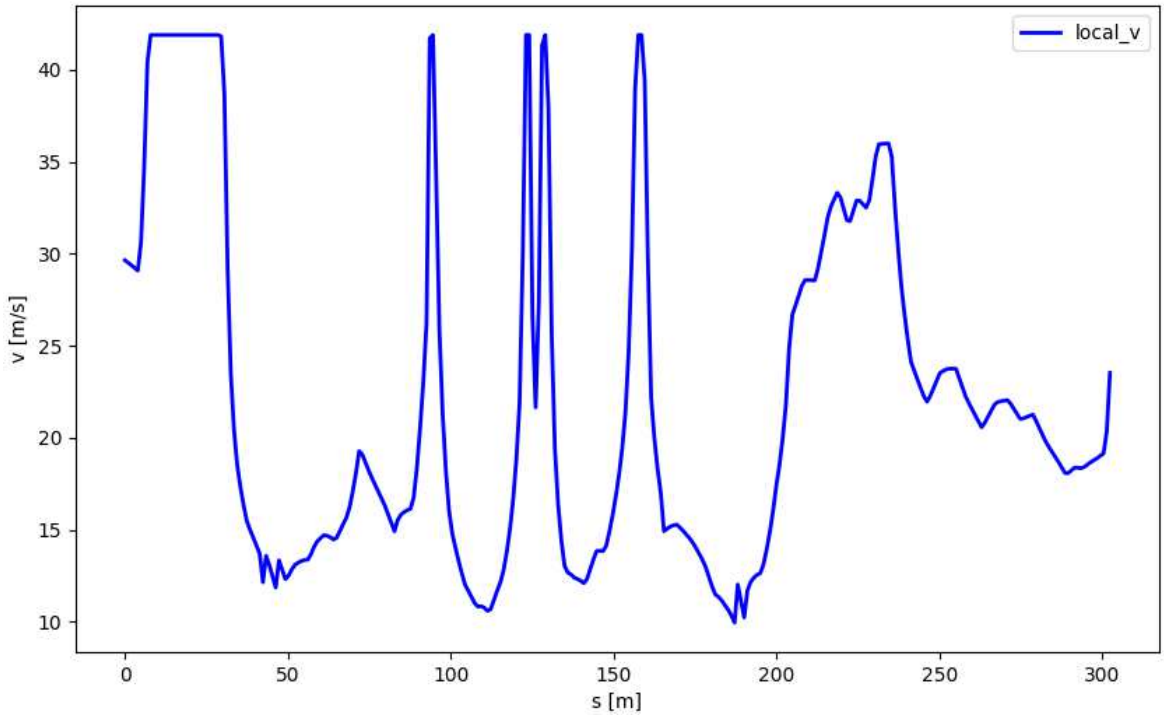


Figure 3.4: Local velocity profile for FSG track.

3.3.2 Forward integration

Forward integration goes through the local velocities and updates the velocities where there is positive acceleration between the two. Longitudinal acceleration, path's segment velocity and corner radius are fed into the engine and traction constraints.

The remaining lateral traction is equal to the:

$$F_{traction} = F_{y_{max}} - \frac{v_x^2 \cdot m}{r}, \tag{3.21}$$

where $F_{y_{max}}$ is the sum of all lateral forces available from the tires based on the load transfer, local longitudinal acceleration and the corner radius. The remaining longitudinal traction is similarly calculated as the lateral:

$$F_{traction} = F_{x_{max}} - a_x \cdot m. \tag{3.22}$$

Lastly, the only remaining constraint is the engine limit, which is the maximum F_x the vehicle's engine can produce at the given local velocity, minus the resistance forces.

The minimum of these forces is taken into account to compute the filter's velocity:

$$a_x = \frac{F_{min}}{m} \tag{3.23}$$

$$v_{acclim} = min(v_{local}, \sqrt{v_0^2 + 2 \cdot a_x \Delta s}), \tag{3.24}$$

where v_0 is the velocity from the previous path segment.

3.3.3 Backward integration

The opposite of the forward integration, backward is updating the velocities where there is negative acceleration between the path's steps. Here, only longitudinal friction is considered, as braking depends solely on the longitudinal vehicle dynamics. The maximum F_x is the sum of all longitudinal forces again depended on the load transfer. That dependence is important because it provides the model with the different friction coefficients based on vertical forces. The equation is same as above 3.22. The minimum velocity is taken again:

$$v_{declim} = min(v_{local}, \sqrt{v_0^2 + 2 \cdot a_x \Delta s}). \tag{3.25}$$

The final velocity profile is the minimum value of the output velocities v_{acclim} and v_{declim} from forward and backward integration.

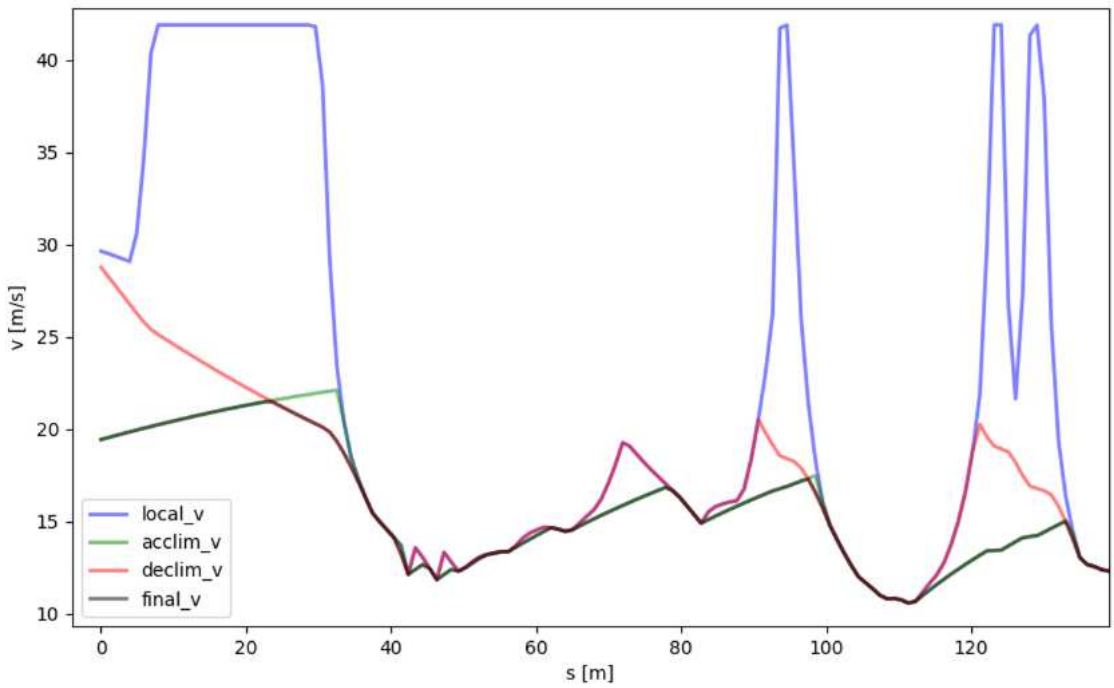


Figure 3.5: Velocity profile with all filters and final velocity.

3.4 Path representation

The spline path geometry used in implementation is described in section 2.2. The left and right constraints of the circuits are blue and yellow cones. The cones do not have to come in a pair, it depends on the circuit layout. For instance, if a part of the track is a hairpin, inner part of the turn will have less cones because of the track layout rule that cones should be at the minimum five meters apart. To solve this problem, line is created that connects the one cone to the closest cone of the contrary color.

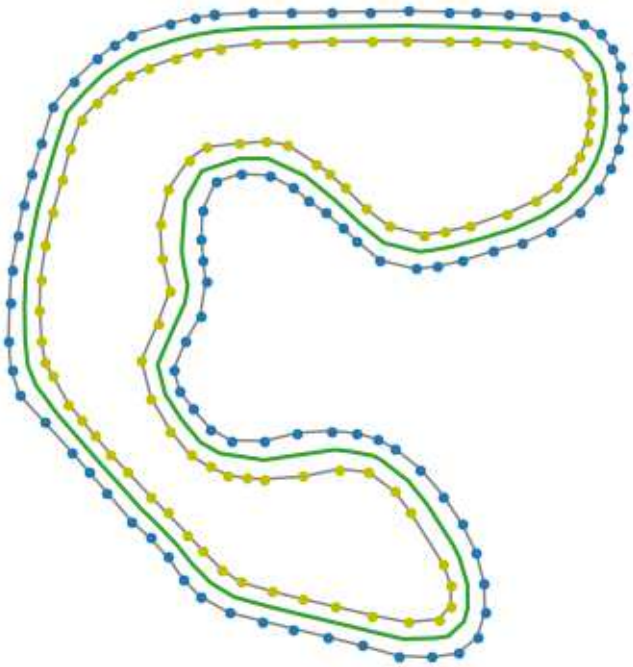


Figure 3.6: FSD track layout with midline.

Iterating the process, midline is created at the each center point of that line. The midline path is now a list of sequential data saved as:

Value	Description
Left cone position	x and y local coordinates.
Differences	x and y difference between left and right cone.
Alpha value	Distance ratio from the left cone inside the interval $[0-1]$, initial alpha value is 0.5.

Table 3.2: Control point data.

Now, when the α value is zero or one, the optimized path could theoretically be directly on the edge of the track. Constraints to the left and right cone position value have to be made so the vehicle would not go over the cones. Based on the orientation vector

from the left cone to the right cone and *vice versa* left constraint and right constraint is made. By adding the car's width and cone's width in the direction of that orientation vector and normalized with the vector norm.

$$x_{new} = x + \frac{w_{car} + w_{cone}}{2} \cdot \frac{\overrightarrow{LR}}{||\overrightarrow{LR}||} \tag{3.26}$$

$$y_{new} = y + \frac{w_{car} + w_{cone}}{2} \cdot \frac{\overrightarrow{LR}}{||\overrightarrow{LR}||} \tag{3.27}$$

These equations give the updated position of the left cone - α now starts from that position.

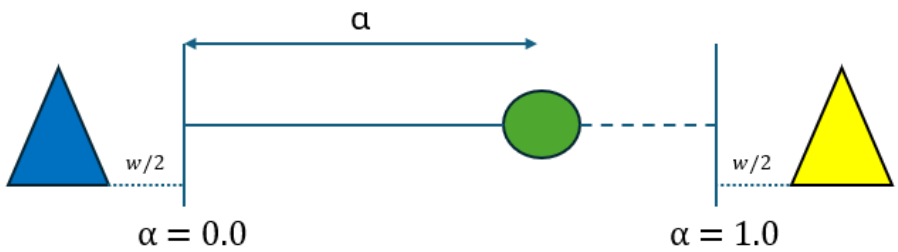


Figure 3.7: Cone pair and midline with α value.

After constraints are made, control points are prepared for spline generation following the explanation in section 2.3. During optimization of the α values, cubic spline is generated at each step. After the optimal values are found, connected cubic splines are discretized every meter in order to calculate the features of each point and to calculate the maximum potential velocity at that point.

3.5 Objective functions

Lastly, the only remaining part of this implementation are candidate objective functions. The objective function is prominently used to represent and solve optimization problems.

As mentioned in chapter 1, the objective functions will not be used directly to minimize time, but rather will be of a heuristic nature. As is commonly known inside motorsport, the less the driver steers the vehicle, the more speed he will carry throughout the corner making him spend less time. When this conclusion is applied to the whole

circuit, the minimum-curvature objective function is an obvious choice. This objective function ensures the least amount of steering. The global curvature can be described as follows:

$$\Gamma^2 = \sum_{i=1}^{n_s} \kappa_i(s)^2 = \sum_{i=1}^{n_s} x_i''(s)^2 + y_i''(s)^2 \tag{3.28}$$

where index i denotes the discretized track segments of the connected cubic splines. As it is previously derived in the section 2.3, cubic splines have continuous second derivation (curvature) which enables the optimization problem to be defined:

$$\begin{aligned} &\text{Minimise} && \Gamma^2(\alpha) \\ &\text{Subject to} && 0 \leq \alpha_j \leq 1 \quad \text{for } j = 1, \dots, n \end{aligned}$$

where j denotes the index of the control point.

However, through testing the minimise curvature objective function, in some cases it is not the optimal way to pass some corners or a sequence of them. The most extreme case scenario is a hairpin turn.

Figure 3.8: Famous hairpin turn in the Formula 1 Monaco GP.

From this example 3.9, it can be assumed that for the small cost of the higher curvature, less time can be spent on the corner by just "hitting the apex" and significantly reducing the traveled path.

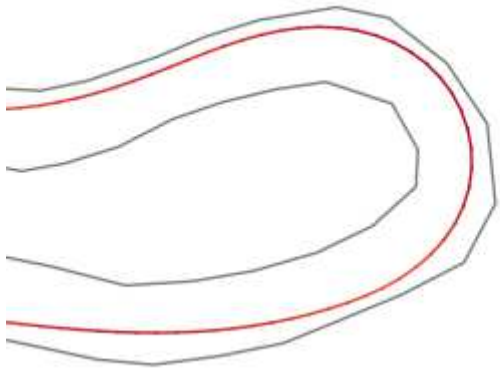


Figure 3.9: Hairpin turn example with minimum curvature.

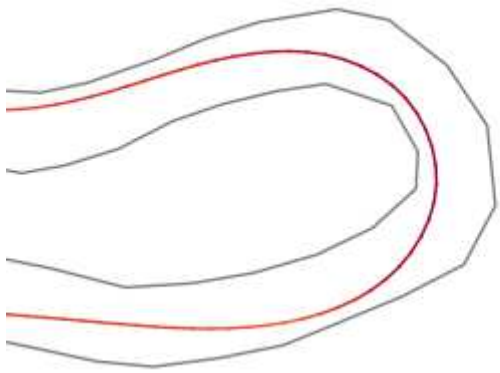


Figure 3.10: Hairpin turn example with the apex hit.

A more optimal approach would be to hit the apex in that corner, which is visible in the figure 3.10, which leads to a new objective function which is the global minimum path:

$$\begin{aligned} \text{Minimise} \quad & S(\alpha) = \sum_{i=1}^{n_s} s_i \\ \text{Subject to} \quad & 0 \leq \alpha_j \leq 1 \quad \text{for } j = 1, \dots, n \end{aligned}$$

where index i denotes discretized path segments and j the control point index.

Finally, to achieve the desired heuristically optimal result, final solution will be to use

the combination of them, as it is proposed in this work [17] to be the suitable approach. Because of the obvious flaw minimum curvature has and the under-performance of the minimum path approach solely has, combining them together solves both function cons. The parameter ϵ determines the ratio between usage of minimum curvature and minimal path. Now, that parameter will be the center of optimization problem and subproblem will be to determine the α values mentioned above. Under the hood, ϵ parameter will be optimized based on lap time. The lap time is calculated using the velocity profile based on the vehicle model.

$$\begin{aligned} \text{Minimise} \quad & F(\varepsilon) = (1-\varepsilon)\cdot\Gamma^2 + \varepsilon\cdot S \\ \text{Subject to} \quad & t(s) \end{aligned}$$

The minimization method implemented within the *scipy* library - *minimize_scalar* is used. It minimizes the scalar value within the given bounds of input parameter.

It seems that again the direct optimization method is being used but it can be avoided. The solution to it is in the next section 3.6.

3.6 Escaping the direct optimization method

To reduce the computational time required to compute the optimal weight between the minimum path and minimum curvature, an offline weight calculation is proposed. The main idea is to find a metric with the best p-value for correlation with the weight ϵ. The p-value determines whether there is a meaningful conclusion that the dataset correlation coefficient differes from zero, based on the sample observations. It ranges from 0 to 1, and correlation is considered meaningful (not random) if the p-value is less than 0.05. In other words, if the p-value is less than 0.05 there is a high probability that the calculated correlation is not random.

The metrics considered include the longest straight, number of turns, sharpest turn curvature, track length, and percentage of turns in the track, etc. During the testing, the best one turned out to be the mean curvature through turns metric with the lowest p-value for a linear regression fitting. Linear regression was used to prevent overfitting,

as the dataset was not large enough for more complex approaches.

Furthermore, the metric was estimated using the corner detection algorithm [18]. It consists of three inputs: minimum curvature threshold, proximity, and length. First, midline path points of the spline are tested to determine if their curvature exceeds the threshold. Then, the corner proximity parameter is applied to detect corners. That parameter checks if there are in proximity more curvatures above the threshold and merges them. After that, the continuity in curvature has to be longer than the length parameter; otherwise it is not classified as a corner. Figure 4.6 represents detected corners using: $\kappa_{min} = 0.03, \Delta s = 7m, l = 5m$.

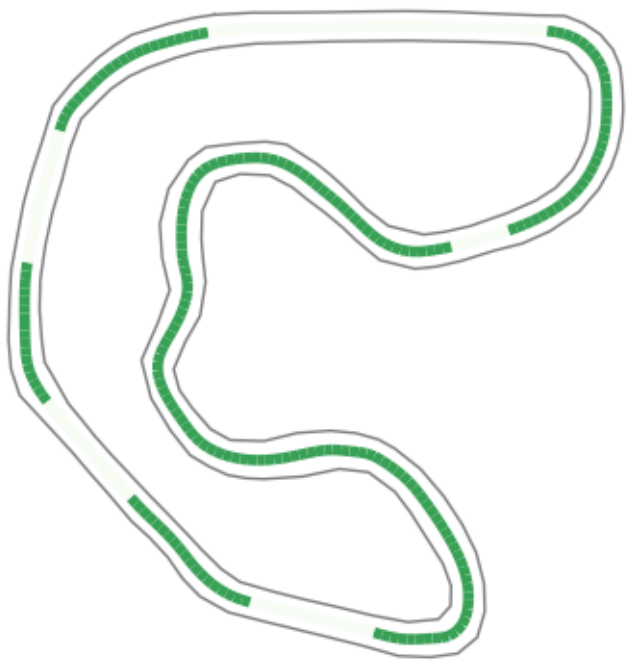


Figure 3.11: Detected corners for FSG track.

Based on the generated mask, the mean curvature through turns can be calculated. Meanwhile, the compromised optimization method calculates the optimal weight ϵ. Now, pairs of mean curvature and optimal weight are stored. Linear fitting is performed using the *sklearn* library and its linear module.

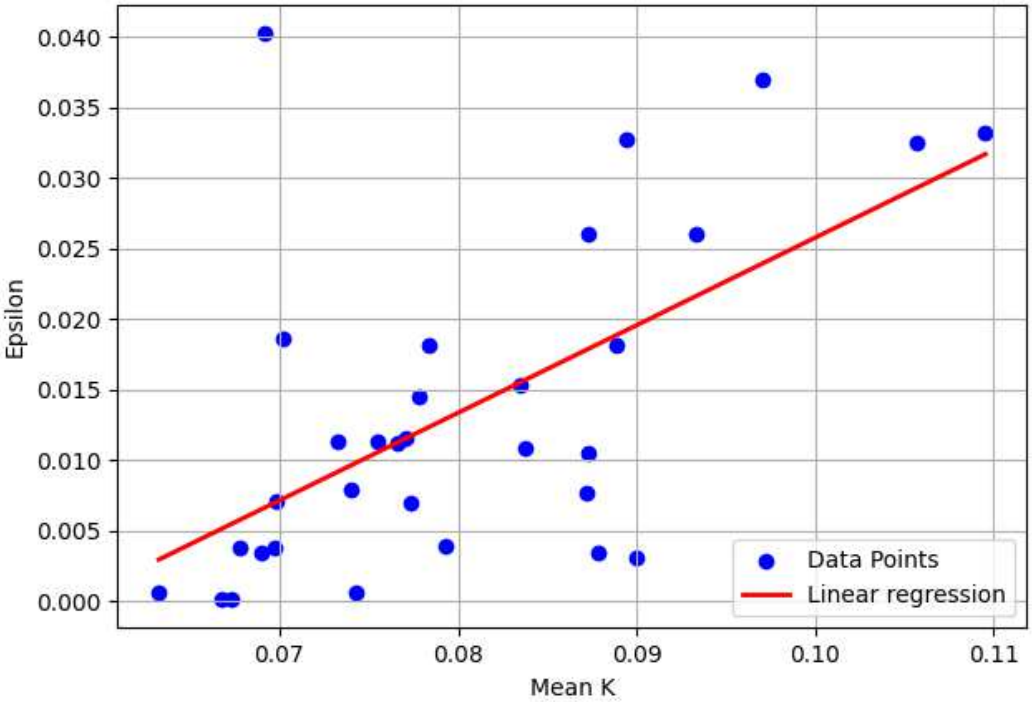


Figure 3.12: Linear regression with r = 0.6020 and p-value = 0.00026732.

This regression gives a candidate function for precomputed weight ϵ:

$$\epsilon^* = 0.62026 \cdot \bar{\kappa} - 0.03625 \tag{3.29}$$

Now, the compromise weight will not be calculated for a specific track;instead, it will depend solely on the mean curvature through turns.

Lastly, the dataset consists of three types of tracks: drawn tracks, randomly generated tracks, and tracks directly estimated from the Trackdrive events. The dataset consists of one FSG track used in most Formula Student simulators, 12 hand-drawn tracks [19], 10 randomly generated [20] and 9 tracks produced by SLAM of StarkStrom Augsburg team [21]. Hand-drawn tracks are created using the drawing to FSD track layout software, which creates cone boundaries while making sure the track layout rules are followed. Randomly generated tracks are made by the *python* implementation of Formula Student Team Delft using a bounded Voronoi diagram created from a uniformly sampled set of points.

4 Implementation analysis

In the analysis chapter, emphasis will be on the computational time of the objective functions and the lap time. FSD track from simulator will be shown with estimated lap time, computational time, weight, and trajectory visualization.

4.1 FSG track example

The performance of the each approach on the track will be listed. The lowest lap time optimization method will serve as the reference point. Its lap time will be labeled as 100%, while other methods will be expressed as a percentage ratio - t_{method}/t_{best}.

This track will serve as the main example. Each method will have its corresponding trajectory visualized. Subsequent tracks will serve only for computing the mean computational time and lap time.

For the record, the direct time-based optimization method (L-BFGS-B) is limited to 15,000 iterations, so the method might not converge within the given number of iterations. Work [11] explains that for extremely nonlinear problems, such as the time-based optimization, computational complexity increases exponentially.

Method	**Lap Time**	**Run Time**	**Lap Time vs. Best**	ϵ
Min path	27.40 s	0.41 s	141.46%	-
Min Γ^2	19.43 s	2.49 s	100.31%	-
Time based	21.68 s	1242.59 s	111.93%	-
Compromise optimal	19.41 s	44.13 s	100.21%	0.037
Compromise offline	19.37 s	2.89 s	100.00%	0.019

Table 4.1: Lap and run times for each method on the FSG track.

From the processed data, it is evident that the time-based optimization method did not converge within the given iteration limit. Moreover, the method was unable to change α values from the midline path.

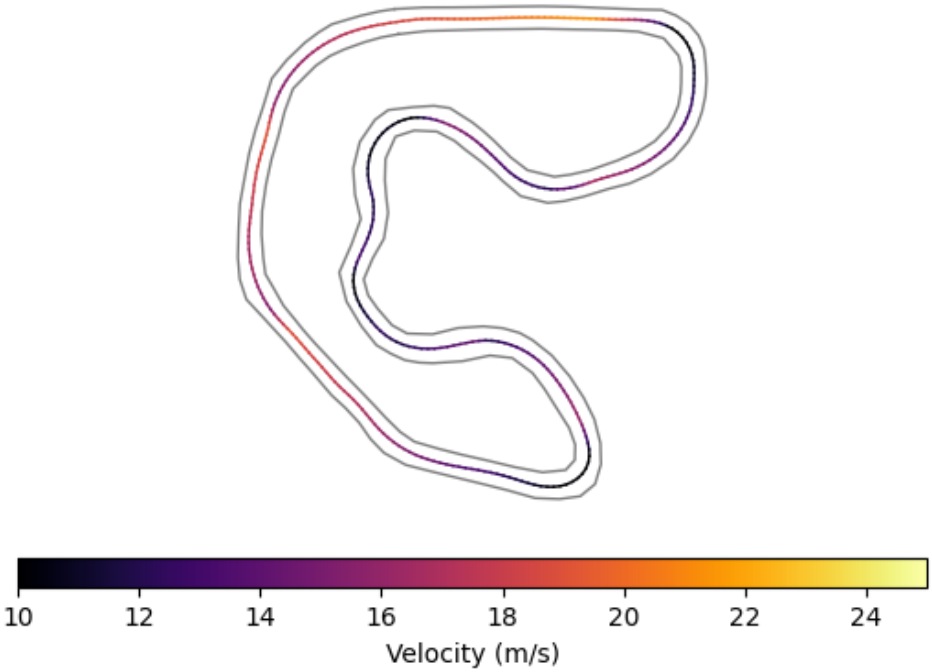


Figure 4.1: Time-based optimization trajectory.

To reduce the long computational time *Tufast Racing Team* from Munich used the curvilinear abscissa approach for track description, algorithmic differentiation using the software framework CasADi, and smoothing the track input data by approximate spline regression. An alternative approach would be warm-starting the optimization with a heuristic optimization method. That solution would be fed into the time-based optimization, giving it close to optimal starting point.

Furthermore, from Table 4.1 interesting conclusion can be made. The offline compromise method achieves a faster lap time than the so-called 'optimal' method. The reason lays in the *optimization_scalar* function from *scipy* library. The optimal solution got stuck in the local minima leading to offline method's weight ϵ perform better for the given track. The function is using Brent's method to find the local minima which is

highly influenced on the starting optimization point of the algorithm and can easily end up in the local minimum.

To solve this, before finding the optimal weight ε^*, grid search is used to shrink the upper and lower weight's bounds. The potential global minimum is now in the close neighborhood of the best performing ε from grid search. Now, *optimization_scalar* function is called with bounds expressed as ϵ-neighborhood. This function is going to find a local minimum which is close, or is the actual global minimum while reducing the time complexity that global minimization method would have.

For visualization purposes, all trajectories will be shown below.

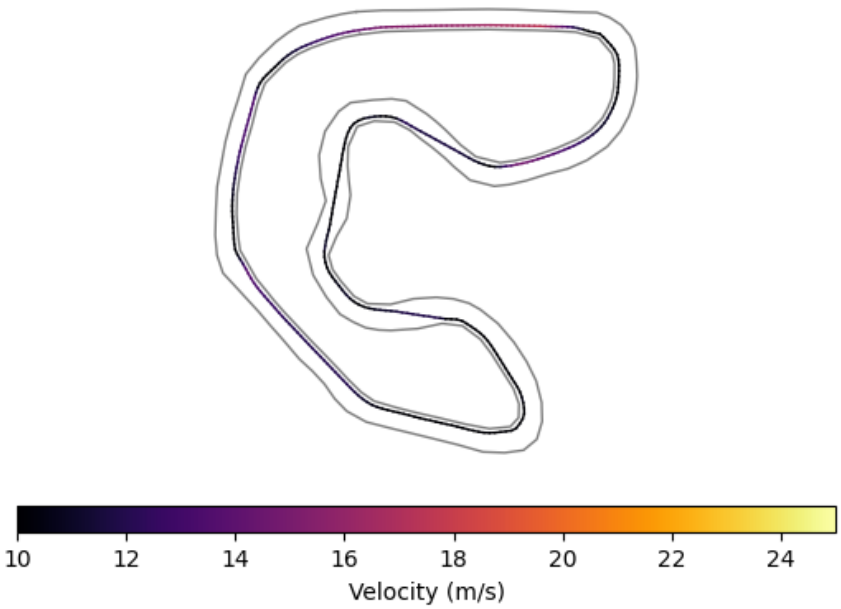


Figure 4.2: Minimum path trajectory.

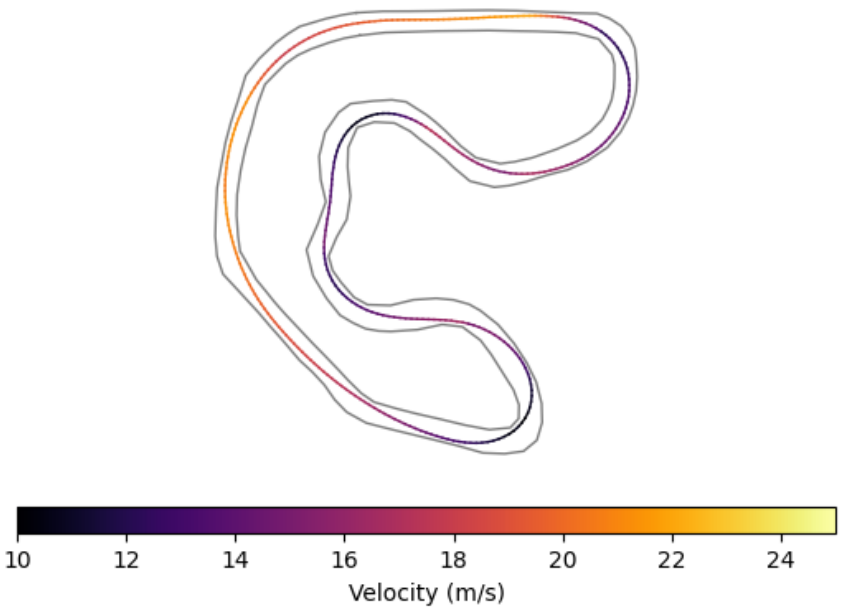


Figure 4.3: Minimum curvature trajectory.

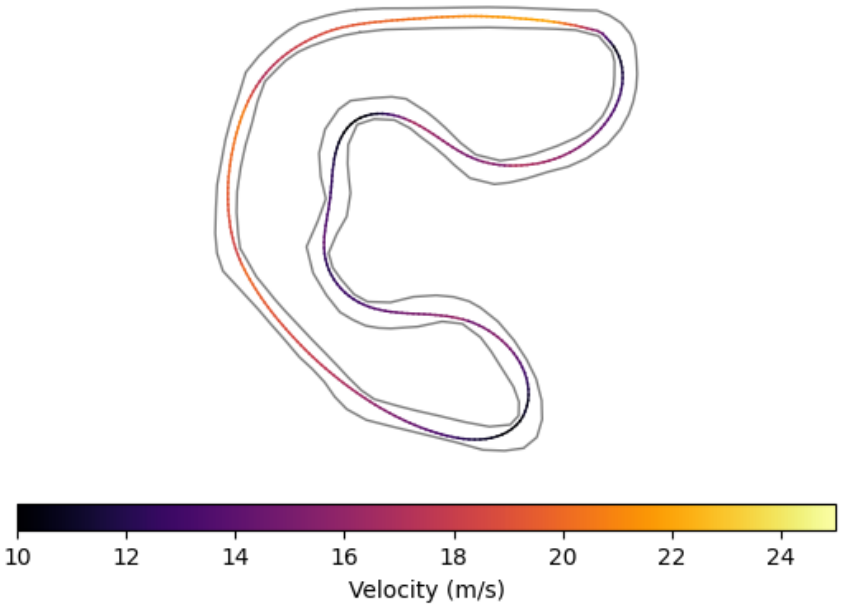


Figure 4.4: Compromise with optimal weight.

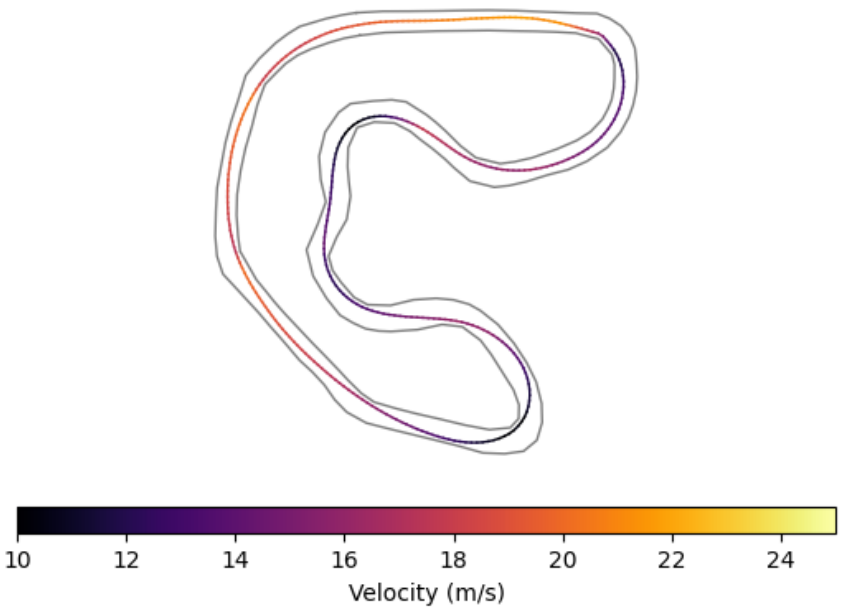


Figure 4.5: Compromise with offline estimated weight.

4.2 Output data for MPC

Vehicle control is implemented through MPC (*Model Predictive Control*). As defined in [6], the control variables are longitudinal velocity v, slip angle β, and yaw rate ω_z. In addition, there are also path variables MPC has to follow: distance from the reference path n and difference between vehicle orientation and tangent line of the reference path. Velocity and path segments have already been explained and defined as output variables but slip angle and yaw rate must also be included as output variables to provide the controller with all necessary execution data.

Yaw rate is calculated using the segment's curve radius and the estimated longitudinal velocity throughout the entire path. Yaw rate is the same as angular velocity around the z-axis.

$$\omega_z = \frac{v_x}{r} \tag{4.1}$$

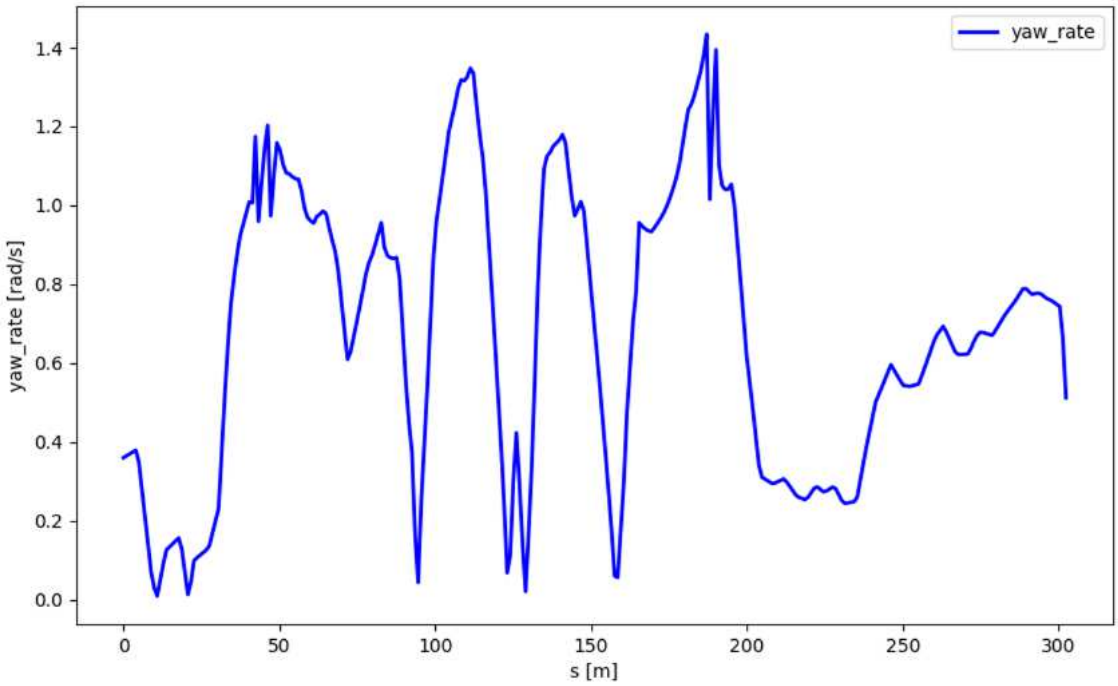


Figure 4.6: Absolute yaw rate values.

The slip angle cannot be obtained using the proposed vehicle model because it is based on the static slip ratio assumption. The optimal slip angle estimated through tire simulations is 12° and this constant will also be included in the output.

The final output consists of x and y path segment coordinates, maximum longitudinal velocities v_x, yaw rates ω_z, and constant slip angles β.

5 Results

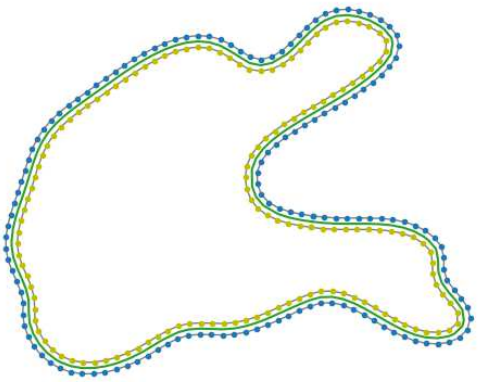
To present the data concisely and intuitively, the estimated performance of the compromise weight will be shown relative to the optimal weight ϵ^* and the minimum curvature method $\epsilon = 0$. Different track datasets will be grouped together to showcase their potential influence on the optimization.

Track	**Lap Time vs.** $\varepsilon = 0$	**Lap Time vs.** ε^*
FSG	100.00%	100.32%
Hand-drawn	98.72%	100.22%
Random	99.91%	100.21%
FSD	98.63%	100.53%
Mean	99.10%	100.31%

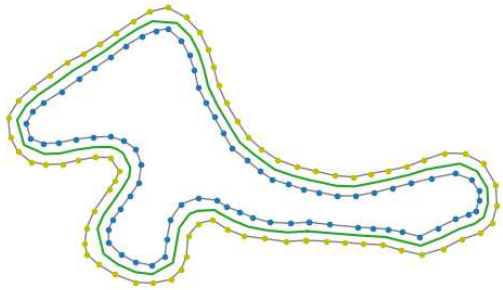
Table 5.1: Mean lap times for curvature and optimal compromise method relative to offline compromise .

From Table 5.1 it is clear that by adding a bit weight to the minimum path, faster times can be achieved. On average, the lap time for the optimal method is 1.21% lower than for the minimum curvature method. It would seem that it is not that much of a difference, but if it is put in the perspective of actual lap times, it is significant. The average lap time taken for the vehicle to drive through these tracks is $30s$. The difference in output ratio multiplied by $30s$ equals 0.363 faster lap times.

Moreover, it can be seen that the biggest difference is in the FSD tracks that are generated using the collected data from actual competitions. The least difference is in the randomly generated tracks, which is less than 0.5%. That leads to a conclusion, the main



(a) Randomly generated track.



(b) FSD track.

reason difference is not that significant on the randomly generated tracks is that the curvature is consistent throughout the corner and there are little to no sharp or hairpin turns, which have been shown to be affected the most by compromise optimization in Section 3.5. That is a direct consequence of the algorithm's inability to recreate sharp turns and inconsistent curvatures through corners. To support the conclusion, below are figures of the one FSD track and the one randomly generated.

In addition, run times of the offline compromise method are on average, similar to the minimum curvature method. The run time duration is 4s for both methods and depends solely on the number of optimizable α values. The mean run time if the optimal compromise method is 80s and is disregarded as optimal optimization method for this particular use case.

6 Conclusion

6.1 Achievements

The developed trajectory optimization solution successfully accomplished the set goals. The goals were to find a heuristic approach that would reduce time and computation complexity while keeping the lap times close to the direct optimization method.

Candidate optimization functions were minimum curvature, minimal path and a compromise between each. On average, the compromise method yielded the best results but with a high runtime cost. This led to the offline compromise weight estimation using the simplest approach - linear regression.

To collect enough *Trackdrive* track data which is sparse, synthetic track generation was used with various methods to prevent overfitting.

Finally, the formulated optimization problem was nonlinear and non-convex. It was decided not to use global optimization methods due to the high computational cost but instead to settle for nearly optimal results. The optimization method for objective functions was L-BFGS-B, a quasi-Newton method which is suited for nonlinear convex problems. It is used because of its high convergence, robustness in gradient descent, and good performance with a large number of parameters.

The choice of the right vehicle model regarding the complexity and available data has proven to be a key factor in accurately describing the vehicle. In this way, the vehicle model is closely matched with vehicle's capabilities in the real-world scenario. Another reason this is important is the performance of the control algorithm, if the vehicle is poorly described, output trajectory will either underperform or exceed the capabilities of a real vehicle. In that case, the control algorithm will struggle to follow the given

trajectory and may either overshoot or undershoot it.

In conclusion, this solution combines the rather complex vehicle model with simple heuristic objective functions and optimization methods.

6.2 Future work and improvements

This is only the prototype version written in *python* for the purpose of fast implementation and prototype presentation.

This concrete solution can be improved with better midline estimation and denser α values of the spline. Currently, the corner's inner boundaries are connected to the closest matching opposite boundary which leaves less values than it can be derived.

In the future, trajectory optimization should be rewritten in C++ within the ROS wrapper. That way, computational time will reduce significantly and make the implementation competitive.

The currently implemented version expects the correct track boundaries with no duplicate, missing cones or incorrect cone color classification. In that case, the algorithm would not be able to find the midline path and consequently the optimal one. Midline generation will be replaced with a more robust method because the SLAM map output does not have the exactly correct boundary map.

Next, the reference yaw rate that is the input for the MPC controller should be updated to receive positive and negative values instead of absolute. That would require to change how curvature is stored in the memory. Positive derivations would represent the right rotation and negative the left rotation. Currently, only the absolute Euclidean value is stored.

The track dataset can be filled with actual recorded track data and additional synthetic data for better weight parameter estimation.

Once the complete autonomous pipeline is set up, the next step would be practical testing. Then the influences from other autonomous system parts will evaluate the robustness of the software. An important aspect to observe will be the interaction between

the trajectory output and the control system's behaviour.

To extract maximum potential from this trajectory optimization, it should be adapted to work also on incomplete tracks. Currently, this method can only optimize complete global trajectory because it needs the spline closure, later it can be easily updated, which would slightly reduce the lap time of the first lap and have an instant complete optimal trajectory for the second lap and laps onward.

References

[1] E. Lawrence, "Redbull racing: Tech that changed road cars," https://www.redbull.com/int-en/motorsport-tech-that-has-changed-road-car-production, accessed: 2025-02-07.

[2] T. I. of Automotive Technology, "Global race trajectory optimization," https://github.com/TUMFTM/global_racetrajectory_optimization, accessed: 2025-02-09.

[3] R. V. e. a. A. d. Kabzan J, Miguel de la IV, "The full autonomous racing system," *arXiv.org*, 2019.

[4] S. Thrun, "Fastslam: An efficient solution to the simultaneous localization and mapping problem with unknown data association," *Computer Science Department Stanford University*, 2003.

[5] D. Đurašinović, "Slijeđenje zadane putanje cestovnog vozila zasnovano na modelskom prediktivnom upravljanju," *Master thesis, University of Zagreb*, 2023.

[6] M. Lovreković, "Upravljanje gibanjem trkaćeg bolida koristeći modelsko prediktivno upravljanje," *Master thesis, University of Zagreb*, 2024.

[7] R. A. K. Mihail Pivtoraiko and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Carnegie Mellon University Pittsburgh, Pennsylvania*, 2009.

[8] S. C. P. S. K. E. F. P. T. J. hwan Jeon, R. V. Cowlagi and K. Iagnemma, "Optimal motion planning with the half-car dynamical model for autonomous high-speed driving," *American Control Conference*, 2013.

[9] E. Velenis and P. Tsiotras, "Minimum time vs maximum exit velocity path optimization during cornering," *Georgia Institute of Technology*, 2005.

[10] T. G. D. A. Kristoffer Bergman, Oskar Ljungqvist, "An optimization-based receding horizon trajectory planning algorithm," *Linkoping University, Sweden*, 2020.

[11] J. G. J. C. Kapania, Nitin R; Subosits, "A sequential two-step algorithm for fast generation of vehicle racing trajectories," *Cornell University Library*, 2019.

[12] J. R. Chasnov, *Numerical methods*. Hong Kong University of Science and Technology, 2024.

[13] Y.-H. Dai, "Convergence properties of the bfgs algoritm," *SIAM Journal on Optimization*, vol. 13, pp. 693–701, 01 2002. https://doi.org/10.1137/S1052623401383455

[14] E. Velenis and P. Tsiotras, "Minimum-time travel for a vehicle with acceleration limits: Theoretical analysis and receding-horizon implementation," *J Optim Theory Appl*, 2008.

[15] A. Rastogi, "Two to four: Bicycle model for car," https://architrstg.medium.com/two-to-four-bicycle-model-for-car-898063e87074, accessed: 2025-02-09.

[16] R. Rajamani, *Vehicle dynamics and control*. Springer, 2012.

[17] F. Kolarić, "Design of yaw rate controller for electric vehicle," *Master thesis, University of Zagreb*, 2023.

[18] J. Davison, "Racing line optimisation for autonomous vehicles," *Bachelor Thesis, University of Bath*, 2020.

[19] P. Papalotis, "Drawing to fsd layout," https://github.com/papalotis/drawing-to-fsd-layout/tree/main, accessed: 2025-02-23.

[20] M. van Löben Sels, "Random track generator," https://github.com/mvanlobensels/random-track-generator, accessed: 2025-02-23.

[21] I. Ivanov, “Fsd racetrack dataset,” https://github.com/iv461/fsd_racetrack_dataset, accessed: 2025-02-23.

Abstract

Optimization of motion trajectories for autonomous racing vehicles

Marko Jurić

Race trajectory optimization is an essential component for achieving the fastest possible lap. It is part of the whole autonomous system pipeline of a Formula Student single-seater. This work is implementing a heuristic approach to trajectory optimization with objective functions such as minimal curvature, the shortest path, and the combination of both. Using this approach, path planning is separated from trajectory optimization, which significantly reduces the computation time. To further reduce the computation, a compromise weight between minimal curvature and the shortest path estimation is calculated offline. The dataset for offline weight estimation is based on synthetically generated tracks. The single-seater mentioned is described with a dynamic two-track vehicle model using the torque with respect to the engine speed curve, the friction coefficient with respect to the vertical force in both lateral and longitudinal directions, and basic vehicle parameters. The generated trajectory compromises some lap time, in regards to the direct optimization, to reduce time complexity.

Keywords: trajectory optimization; race car; Formula Student; minimum curvature; heuristics; synthetic data; non-convex optimization; autonomous vehicle

Sažetak

Optimizacija trajektorije gibanja autonomnog trkaćeg vozila

Marko Jurić

Optimizacija trkaće trajektorije ključan je dio za postizanje najbržeg mogućeg kruga. Ovaj je rad dio cjevovoda autonomnog sustava Formula Student bolida. Korišten je heuristički pristup optimizacije trajektorije gdje su objektivne funkcije minimalna zakrivljenost, najkraći put i njihova kombinacija. Koristeći taj pristup, planiranje putanje odvojeno je od optimizacije trajektorije što uvelike smanjuje vrijeme izvođenja. Kako bi se još ubrzalo izvođenje, težina koja određuje kompromis između estimacije korištenja minimalne zakrivljenosti i najkraćeg puta računa se unaprijed. Skup podataka za unaprijedno estimiranje težine temelji se na umjetno generiranim stazama. Već spomenuti bolid opisan je pomoću dinamičkog dvotračnog modela vozila koristeći graf ovisnosti okretnog momenta o brzini motora, graf ovisnosti bočnih i uzdužnih koeficijenata trenja o vertikalnim silama i osnovnim parametrima vozila. Generirana trajektorija nešto je sporija nego ona izravna, ali smanjuje vremensku složenost.

Ključne riječi: optimizacija putanje; trkaći automobil; Formula Student; minimalna zakrivljenost; heuristika; sintetički podaci; nekonveksna optimizacija; autonomno vozilo