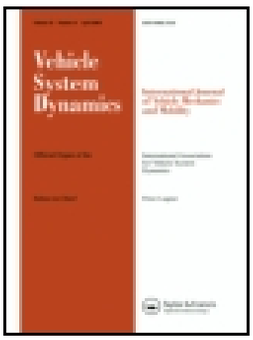

Taylor & Francis
Taylor & Francis Group

Vehicle System Dynamics
International Journal of Vehicle Mechanics and Mobility

ISSN: 0042-3114 (Print) 1744-5159 (Online) Journal homepage: https://www.tandfonline.com/loi/nvsd20

Minimum curvature trajectory planning and control for an autonomous race car

Alexander Heilmeier, Alexander Wischnewski, Leonhard Hermansdorfer, Johannes Betz, Markus Lienkamp & Boris Lohmann

To cite this article: Alexander Heilmeier, Alexander Wischnewski, Leonhard Hermansdorfer, Johannes Betz, Markus Lienkamp & Boris Lohmann (2019): Minimum curvature trajectory planning and control for an autonomous race car, Vehicle System Dynamics, DOI: 10.1080/00423114.2019.1631455

To link to this article: https://doi.org/10.1080/00423114.2019.1631455

VEHICLE SYSTEM DYNAMICS
https://doi.org/10.1080/00423114.2019.1631455

Minimum curvature trajectory planning and control for an autonomous race car

Alexander Heilmeier [a], Alexander Wischnewski[b], Leonhard Hermansdorfer[b], Johannes Betz[a], Markus Lienkamp[a] and Boris Lohmann[b]

[a]Chair of Automotive Technology, Technical University of Munich, Munich, Germany; [b]Chair of Automatic Control, Technical University of Munich, Munich, Germany

ABSTRACT

This paper shows a software stack capable of planning a minimum curvature trajectory for an autonomous race car on the basis of an occupancy grid map and introduces a controller design that allows to follow the trajectory at the handling limits. The minimum curvature path is generated using a quadratic optimisation problem (QP) formulation. The key contributions of this paper are the extension of the QP for an improved accuracy of the curvature approximation, the introduction of curvature constraints and the iterative invocation of the QP to significantly reduce linearisation errors in corners. On the basis of the resulting raceline, a velocity profile is calculated using a forward-backward-solver that considers the velocity dependent longitudinal and lateral acceleration limits of the car. The advantages and disadvantages of the proposed trajectory planning approach are discussed critically with respect to practical experience from various racetracks. The software stack showed to be robust in a real world environment as it ran successfully on the Roborace DevBot during the Berlin Formula E event in May 2018. The lap time achieved was within a tenth of a second of a human driver and the car reached about 150 km/h and 80% of its acceleration limits.

ARTICLE HISTORY
Received 9 October 2018
Revised 21 May 2019
Accepted 27 May 2019

KEYWORDS
Trajectory planning; path planning; control; minimum curvature; autonomous driving; race car

1. Introduction

Regarding the multi-stage autonomy of vehicles based on the SAE categorisation, level 5 will enable a completely self-driven vehicle without a driver. This means that all tasks, which have previously been accomplished by a driver, must now be performed using algorithms alone. This includes perceiving the environment, planning trajectories and following them.

To benchmark state-of-the-art software for autonomous cars at the physical limits of a vehicle, a team from the Chair of Automotive Technology and the Chair of Automatic Control of the Technical University of Munich (TUM) takes part in the Roborace competition. Roborace provides an electrically powered, automated level 5 race car called DevBot,

CONTACT Alexander Heilmeier alexander.heilmeier@tum.de

which serves as a platform for practical testing of the programmed software. The TUM team developed the complete autonomous driving pipeline for controlling the vehicle [1]. The software was publicly presented in May 2018 at the Formula E Event in Berlin [2]. On this event, the DevBot drove three autonomous laps on the racetrack with a maximum velocity of 150 km/h. The lap time achieved was within a tenth of a second of an average human driver [2].

The general workflow to let the vehicle drive autonomously is as follows: Firstly, the environment around the car on the racetrack must be perceived based on walls and free spaces. We focused on the 2D-LiDAR data of the DevBot to create an occupancy grid map of the racetrack. This is done using the SLAM (Simultaneous Localisation And Mapping) algorithms gmapping [3] or Google Cartographer [4]. Next, the map is processed to obtain the centreline of the track. This is the input for the trajectory generation presented in more detail below. Finally, the vehicle controller is fed with small trajectory parts with a defined time horizon while the car is driving around the track.

This paper presents the planning and control parts of the software stack allowing an autonomous race car to drive a racetrack at the handling limits. The key contributions are the theoretical formulation of the minimum curvature path optimisation problem including its extension for an improved accuracy of the curvature approximation, the introduction of curvature constraints originating from a real car's steering design and the iterative invocation of the QP to significantly reduce linearisation errors in corners. Furthermore we introduce several extensions to the planning algorithm to guarantee robust operation in real world application. All these have increased the quality of the solution significantly on the narrow Formula E tracks compared to state-of-the-art approaches.

The rest of the paper is structured as follows: Firstly, related work for the relevant topics is presented. Then creation of the centreline is introduced just before trajectory generation is discussed in more detail. The methodology section ends with a description of the controller design. Afterwards the results are presented and the advantages and disadvantages of the approach are discussed.

2. Related work

The field of autonomous driving has been the focus of research for several years. Various full vehicle concepts have been proposed, e.g. during the DARPA Grand Challenge [5] and the DARPA Urban Challenge [6–12]. Both focused on rather low speed scenarios but the latter included several navigational aspects and decision-making processes in complex, dynamic environments.

Racing scenarios have not been highly investigated although they are highly relevant for autonomous driving on motorways and in the urban environment, e.g. in emergency evasion situations. An exception is an autonomous Audi TTS that is operated by Stanford University [13–15] reaching competitive performance levels on various tracks. Here, the racing problem is separated into trajectory planning and trajectory control. It was shown that their approach is capable of nearly achieving the vehicle limits. In contrast, Liniger [16] and Liniger et al. [17] present multiple optimal control strategies to solve the trajectory planning and control problem within a single algorithm. The main advantage of this approach is the physically meaningful parametrisation via the nonlinear vehicle-model

and the fact, that a single algorithm takes care of both problems. This allows to incorporate tire force constraints in a straight-forward way. However, we believe that integrated trajectory planning and control approaches are difficult to scale to complex scenarios due to the non-convex nature of the constraints imposed by multiple vehicles. Furthermore, these algorithms are considered to be more complex during the tuning process due to the mixture of tasks. This point of view is in line with the split of tasks proposed by many researchers discussing the architecture for autonomous driving systems in road scenarios [18–20].

2.1. From map to centreline

To be able to use the information gained by LiDAR scanning of the racetrack, the resulting data must be processed in such a way that it can be used for trajectory planning. In our case, the inputs of the planning module are the racetrack's centreline and the corresponding track widths. There are several approaches for extricating the road from LiDAR data and generating the corresponding centreline. Common approaches are Support Vector Machine Classification [21], the use of kernel density estimation [22], canny edge detection and subsequent hough transformation [23] or different applications of neural networks, e.g. [24].

The above mentioned methods work for a variety of scenarios and are able to deal with huge street networks, but sometimes fail when it comes to small scales, e.g. single roads. Therefore, some other approaches have to be considered with regard to the special environment in which the centreline generation has to be applied in our project. A unique feature of many Formula E racetracks is having walls right next to the tarmac. Therefore, the drivable area can be obtained by considering the walls as track limits. Consequently, the resulting 'free of obstacles' area has a tube-like shape. The task of finding the centreline of such a shape is very common in clinical procedures when visualising human blood vessels. On the one hand there are hand-tuned filters designed to respond to certain structures and on the other hand classification techniques using machine learning. Guedri et al. [25] utilises consecutive image-processing methods (e.g. binarisation, skeletonisation) and a simple classification for tracking the centreline of human blood vessels. Sironi et al. [26,27] reformulate the centreline detection in terms of a regression problem.

2.2. Trajectory planning for autonomous cars

A trajectory defines the path a moving object follows in the space-time domain. Both, path and trajectory planning methods for autonomous cars can be found in literature. As velocity profile planners such as [15,28–30] can generate the missing time domain information based on a given path, both types may be suitable for attaining our goal. Therefore, we do not distinguish between path and trajectory planning.

Survey papers [19,31] serve as an entry point to the topic as they compare different algorithms, e.g. in terms of optimality, completeness and time complexity. The solution approaches can usually be categorised into three classes [19], even though the assignment is often not clearly defined:

- Variational methods

- Graph search methods
- Incremental search methods

In variational methods, the path is usually represented by splines, whose parameters are optimised in terms of a cost function. This includes optimisation techniques such as optimal control [32–36] and geometrical optimisation [30].

Graph search methods discretise the possible configuration space of the car and use heuristic information to search for a cost optimal path through the graph. The best known example of this type is Dijkstra [37]. It has been improved by the A* family [38,39]. Dynamic environments with moving objects can be considered in D* algorithms [40–42].

Incremental search methods are similar to the graph search methods but are based on random sampling of the configuration space. The samples are then connected by an algorithm. The cost optimal path to the target point therefore continues to improve the longer the algorithm runs. A widely known approach is the Rapidly-exploring Random Tree (RRT) [43]. It has been succeeded by RRT* [44] and became suitable for dynamic environments as RRT^X [45]. Another relative is EST [46].

Combined approaches are found in the context of the DARPA challenges that were mentioned above. They represent a spectrum of approaches, e.g. optimal control and lattice planners in [6] and Hybrid A* in [7].

Most of the algorithms can only be applied for low to medium lateral accelerations and at low velocities. The amount of research into high performance cars and race cars is much smaller. A major problem here is to plan a valid trajectory that meets all the constraints at high velocity and accelerations while keeping desirable smoothness properties. Several authors [16,17,47] propose to use optimal control for this purpose. Gerdts et al. [48] extend an optimal control approach by a moving horizon to reduce the computational effort of the optimal control problem. Another approach is the offline optimisation of pre-planned manoeuvres [49], which can then be composed online according to traffic and driving situation. In the context of minimising lap time on the basis of optimal control, there are indirect and direct approaches. The former are based on Pontryagin's minimum principle to find the solution, e.g. [35]. Perantoni and Limebeer [34] is a representative of the latter category. Here, the optimal control problem is transformed into a nonlinear programming problem. Dal Bianco et al. [36] gives an overview of this field and compares the different approaches. Typical disadvantages of optimal control problems are high computation times or high complexity of the parametrisation and implementation. Jeon et al. [50] and Arab et al. [51] are based on RRT* motion planners. While the former separates the problem into path and velocity profile generation, the latter combines a sparse version of RRT* with model predictive control (MPC). Rizano et al. [52,53] present path planning approaches for autonomous race cars based on the concatenation of circular arcs and straights. However, the description of the track imposes some limitations in this approach. The Audi TTS in [54] drives with a similar approach. Here, the pre-computed path is composed of four parts per corner: straight, entry clothoid, constant radius arc and exit clothoid. Braghin et al. [30] uses a geometrical optimisation of the path along a racetrack to minimise the curvature. Kapania et al. [15] divides the trajectory generation into two sequential sub-problems to reduce computational effort. At first a velocity profile is generated before a convex path optimisation problem is solved

which minimises the resulting path curvature while taking the vehicle's handling limits into account.

2.3. Vehicle control

Nowadays, lateral and longitudinal control systems are widely applied in series-production vehicles. Conceptually they are treated as independent systems using the lateral deviation from a path and the vehicle velocity as reference variables. Due to its more complex nature, the former receives more attention in research activities. In its basic form, it usually relies on an output feedback controller for a look-ahead point placed in front of the vehicle [14,55,56]. Several nonlinear modifications have been proposed in recent years, using Lyapuov [57], Exact-Linearisation [58,59] or Sliding-Mode [60,61] design techniques. Another design direction is utilising optimisation based controllers [17,62]. However, their use is sensitive to model quality, numerical effects and they impose heavy computational demands. For this reason, this design direction usually involves higher development efforts. A detailed comparison of several concepts can be found in [63].

3. Methodology

This section first introduces the architectures of the hardware used in Roborace and the software developed by the TUM team. Then the software parts are described in more detail starting with the centreline generation and ending with the controller implementation. The plots in this section mostly show the same characteristic corner combination of the Berlin Formula E track to be able to follow the progress from one step to the next. The segment is a good benchmark for the algorithms, as it includes the end of a long straight as well as corners with different narrowness.

3.1. Autonomous driving system components

During the Roborace competition, we used the so-called DevBot, a race car based on an LMP3 chassis [64]. It was designed as a rapid prototyping platform for autonomous driving algorithms and can therefore be driven by a human driver as well as autonomously. It provides various sensors, of which the following are relevant for this paper: Four LiDAR sensors around the front wheels, an OxTS 4002 inertial measurement unit including GPS, a Kistler SFII P optical velocity sensor and four wheelspeed sensors. Two control units can be utilised to run the algorithms: a Nvidia Drive PX2 for planning and decision tasks and a Speedgoat Mobile Target Machine for real-time control tasks. For more information about the vehicle and the holistic autonomous driving pipeline used in the car, we refer to [1].

3.2. Autonomous driving software architecture

The software is divided into three main modules. The perception module generates a representation of the environment. This includes the detection of track boundaries and objects as well as tarmac recognition. These informations are handed over to the planning module, which is responsible for making decisions at long-term level (race strategy decisions

and generation of a global race trajectory) and short-term level (evasion and overtaking manoeuvres). It generates local trajectory snippets for the control module. The latter is aimed at tracking the local target trajectories.

The focus of the Berlin competition in 2018 was to set the best possible lap time in an environment free of opponents and obstacles. This allowed the perception and trajectory optimisation parts shown in Figure 1 to be carried out offline. Three manually driven laps were used to generate an occupancy grid map of the track by fusing GPS, odometry and LiDAR measurements. This map was then post-processed. Run-off areas and curbs required manual reworking because it is almost impossible to detect them with the used LiDAR. The step also includes the generation of the centreline required for the subsequent optimisation algorithm. The actual driving trajectory is obtained in a two-step procedure: first finding a minimum curvature path and then generating a suitable velocity profile that considers the vehicle's handling limits.

Figure 2 depicts the software modules which are active during the driving sessions. The Planning Unit (PU) performs two main tasks. One of them is monitoring the system mission (Behaviour State Machine). This includes launching the car, monitoring the subprocesses, counting the number of laps and adjusting the velocity profile based on given scale factors, e.g. to introduce a tire warm-up lap. The other task is the extraction of high fidelity local trajectory parts from the global trajectory that can be used for control (Local Trajectory Generation). To cover enough distance along the track, we adapt the step size of the trajectory snippet sent to the trajectory tracking controller dependent on the current velocity.

A major advantage of the local trajectory concept is that it allows an extension of the functionality in future work, e.g. to implement the handling of static and dynamic objects that are not considered in the offline computed global race trajectory. To ensure safety in the event of a PU or network failure, much longer foresight emergency trajectories are

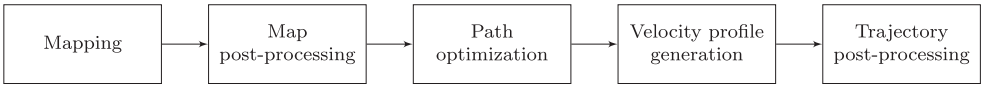

Figure 1. Process flow chart for the generation of a race trajectory (offline).

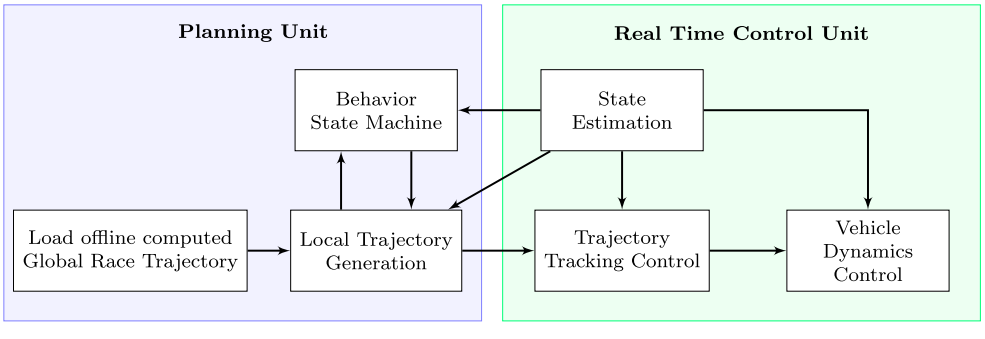

Figure 2. Autonomous driving software architecture of the TUM team (online).

generated simultaneously. These allow the Real Time Control Unit (RTCU) to stop the car safely on its own.

The RTCU implements a two-degree of freedom controller for lateral path and velocity tracking. Both rely heavily on a feed forward control capturing the main vehicle characteristics. The feedback control is based on a state estimate obtained from sensor fusion. This controller interfaces the vehicle via the vehicle dynamics control module using a velocity and curvature set point interface. The latter converts the requested values into steering angles and force requests.

3.3. Generation of a centreline

Since we use 2D-LiDAR data to generate the circuit map, the post-processed LiDAR data is represented as an occupancy grid map with three possible states: 'occupied', 'unknown', 'free of obstacles'. It can be interpreted as an image, which allows the use of certain image-processing methods. In addition to the previously mentioned methods used in clinical procedures for the visualisation of human blood vessels, several additional functions have been implemented. These are necessary to obtain viable results for the path optimisation. The four main steps for generating the centreline and its corresponding track widths are visualised in Figure 3:

- Preprocessing of the raw map data
- Smoothing and filtering of the raw map data
- Applying the Euclidean distance transform and extricating the centreline
- Smoothing of the centreline

Firstly, the original grid map is binarised. The resulting grid map consists of a single binary value for each cell (occupied (true), free (false)). The 'unknown' space is declared as 'occupied' to be on the safe side. The result is shown in Figure 3(a).

Secondly, single pixels and accumulations of a small, predefined number of pixels with state 'occupied' are considered as 'false positives' and are discarded by the algorithm. In addition, small 'free' areas (e.g. single LiDAR beams as seen in Figure 3(a)), which are not relevant for the vehicle due to their small spatial extent, are closed, i.e. considered as occupied. Furthermore, the GPS data of the vehicle during mapping is used to clear the grid map of false positives. The result is a smoothed, tube-like shape that represents the drivable area of the racetrack (Figure 3(b)).

Thirdly, the morphological skeleton method [65] is applied to output the track boundaries. Using the euclidean distance transform [66] allows calculation of the maximum distance from every cell to its nearest track boundary cell. All distance values that are outside the actual racetrack are discarded (Figure 3(c)). The watershed algorithm [67] (usually used for image segmentation) is applied to detect the maximum distance to track boundaries while considering the closed-loop nature of the centreline. The resulting centreline has an edgy, discontinuous shape depending on the track layout. These edges cause a sudden change in direction, which is difficult to handle in the subsequent path optimisation algorithm.

Lastly, a Savitzky-Golay filter is used [68] to smooth the centreline obtained in the previous step. It is separately applied to the x- and y-coordinates. Finally, the algorithm outputs

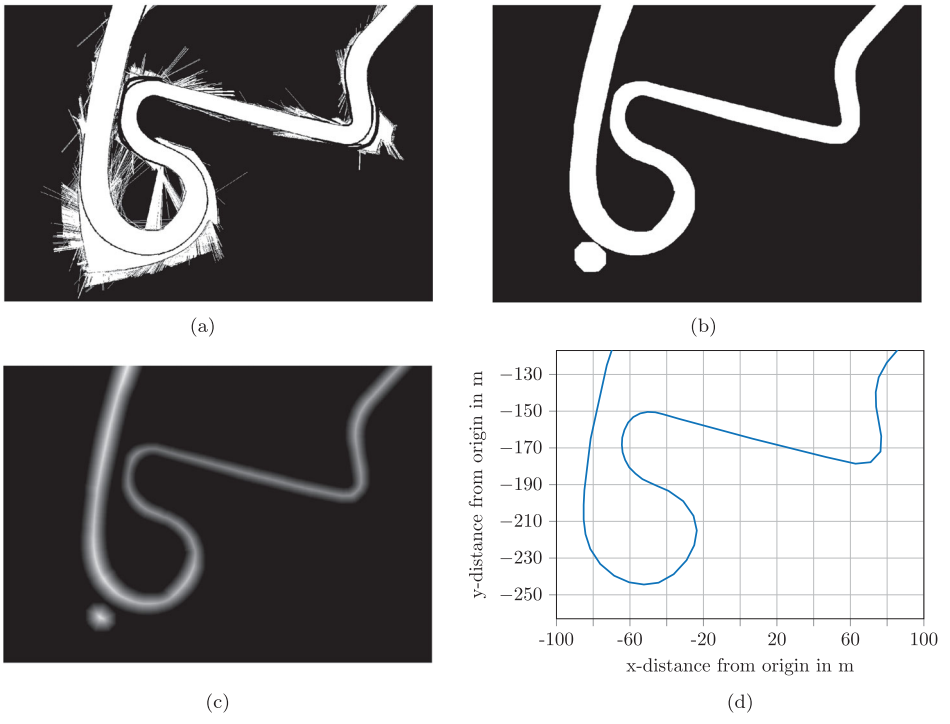

Figure 3. Four main steps of centreline generation (top left to bottom right). (a) Preproccessing of the raw map data (b) Smoothing and filtering of the raw map data (c) Applying euclidean distance transform (d) Smoothing

the xy-coordinates of the centreline and the corresponding track widths (i.e. distance to track boundaries is equal to both sides) of each centreline point (Figure 3(d)).

3.4. Obtaining a minimum curvature path

For our application, a trajectory consists of the seven variables curvilinear distance s, coordinates x and y, heading ψ, curvature κ, velocity v_x and longitudinal acceleration a_x: $[s,\ x,\ y,\ \psi,\ \kappa,\ v_x,\ a_x]$. We had several requirements concerning the trajectory generation. Firstly, it should achieve the best possible lap time. Secondly, the implementation of the algorithm should be robust and reliable to reduce testing time. Thirdly, the calculation time should not exceed a few seconds. This would enable us to use the same approach for replanning of the raceline online on the car in the future. Considering the different concepts presented in Section 2.2, we decided to split the trajectory planning problem into path optimisation and velocity profile generation. Therefore, we calculate the first five dimensions of the trajectory within this section. The velocity and longitudinal acceleration profiles are generated afterwards based on the raceline to obtain a complete trajectory.

The path optimisation approach applied significantly extends the work of Braghin et al. [30]. Kapania et al. [15] is quite similar and has also been considered as a basis. The difference is that this approach first generates a velocity profile and then optimises the path by solving a convex optimisation problem on this basis. The process is repeated until the

calculated lap time no longer improves. We opted for the approach in [30] because [15] shows a slightly large computing time of approximately 90 s (requiring an average of three iterations with 30 s each) and gives no guarantee of convergence.

The idea of the approach presented in [30] is to vary the path on the track in such a way that the globally summed quadratic curvature is minimised. This is not the minimum time solution, which is generally a compromise between minimum curvature path and shortest path [15]. However, the minimum curvature path is reasonably close to a minimum time path because it allows the highest cornering speeds at a given maximum lateral acceleration as given by

$$v_{\max} = \sqrt{\frac{a_{\mathrm{y}}}{\kappa}}. \tag{1}$$

Figure 4 compares the paths and lap times t_{lap} resulting from shortest path optimisation, our final implementation of the iterative minimum curvature optimisation and minimum time optimisation (based on [34,69]) for the Berlin track segment. All solvers consider a vehicle width of 2.0 m. The discretisation step size is 3.0 m. The computation times from centreline import to trajectory output are 4 s for shortest path, 18 s for iterative minimum

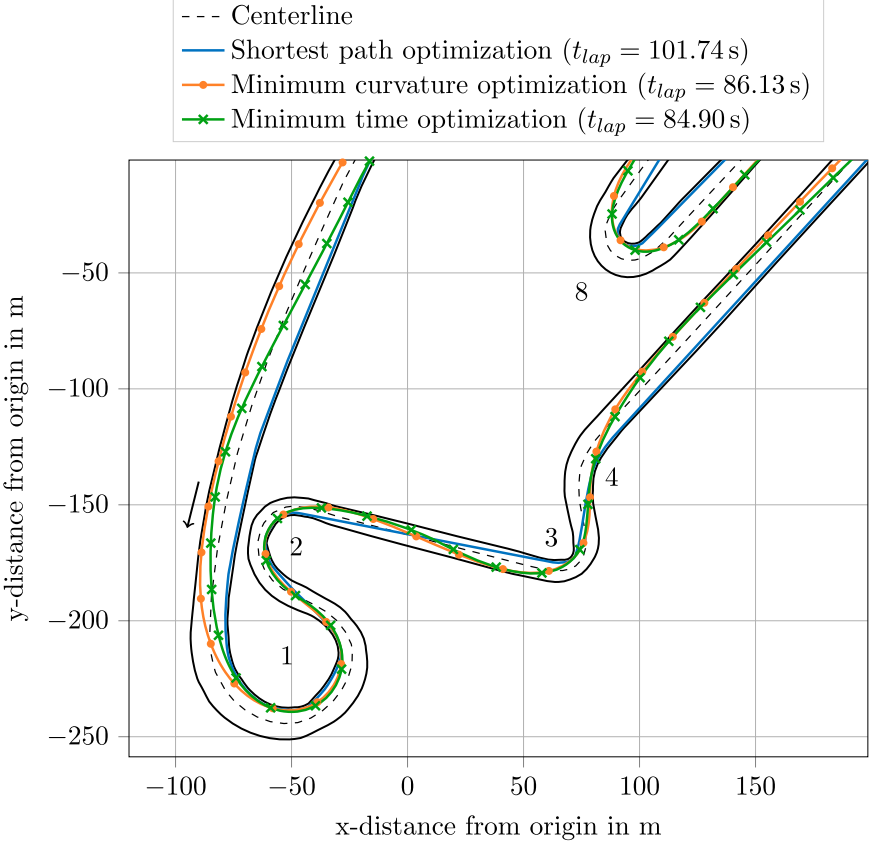

Figure 4. Comparison of the paths resulting from shortest path optimisation, (iterative) minimum curvature optimisation and minimum time optimisation.

curvature and 151 s for minimum time optimisation on a laptop computer (Intel Core i7 2.7 GHz, 16 GB RAM).

The mentioned minimum time optimisation result is based on a two-track model to show the actual time optimal path. However, the velocity profile calculations for the shortest path and minimum curvature results are based on a point mass model. To obtain comparability in terms of the computation time, we also calculated the minimum time result based on a point mass model. In this case, the computation time is 37 s.

As can be seen, the path of the minimum curvature solution is quite close to the solution obtained in the minimum time problem within the corners. On the straights, the minimum time result stays closer to the shortest path as soon as the tires are not fully exploited. The lap time advantage is 1.4% while the computation time of the minimum curvature optimisation problem is 50% smaller (comparing the point mass models). For an offline application, the longer computation time can easily be tolerated to obtain the better lap time. For online applications, however, the lower computation time of the minimum curvature problem can be a decisive argument, especially since car computers such as the Nvidia Drive PX2 often have quite slow CPUs in comparison to the Intel Core i7 in the laptop. Further advantages of the minimum curvature path optimisation are a very low parametrisation effort and the independence from vehicle dynamics parameters as well as the fact that the resulting raceline shows a very smooth curvature profile. All this eases the implementation on a real car significantly. The shortest path solution is primarily interesting for cars that are limited by a low top speed instead of their lateral acceleration capabilities.

3.4.1. Description of the optimisation problem

The approach is formulated as a QP that can be solved quickly and robustly. The inputs for the optimisation problem are the centreline points and the corresponding track widths obtained in Section 3.3. Within the optimisation problem, we switch from centreline definition to reference line definition. The difference is that the reference line may have different track widths on the left and right sides of the line, i.e. it does not have to be in the middle of the track.

The optimisation problem is based on a variation of the raceline points r along the reference line normals, i.e. along the track widths, to minimise the curvature. The notation of the ith raceline point reads

$$\vec{r}_i = \vec{p}_i + \alpha_i \vec{n}_i \tag{2}$$

where $\vec{p}_i = [x_i \;\; y_i]^{\mathrm{T}}$ is the reference line point and $\vec{n}_i$ the unit length normal vector. The independent parameter α_i is used to move the point $\vec{r}_i$ along the normal vector between the track boundaries as visualised in Figure 5. The track boundaries are transformed into boundaries on α_i given by combination of the left and right track widths $w_{\mathrm{tr,left},i}$ and $w_{\mathrm{tr,right},i}$ and vehicle width w_{veh} as

$$\alpha_i \in \left[-w_{\mathrm{tr,left},i} + \frac{w_{\mathrm{veh}}}{2}, w_{\mathrm{tr,right},i} - \frac{w_{\mathrm{veh}}}{2} \right]. \tag{3}$$

The raceline is defined by third order spline interpolations of the points r in x and y coordinates. This enables us to calculate first and second order derivatives explicitly from the spline representation. In the remainder of the section only the x-part is described for the sake of brevity. The y-part follows by straightforward extension of the proposed concepts.

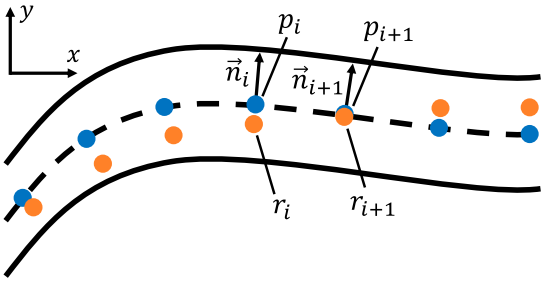

Figure 5. Point notation for the reference line points p and the raceline points r. The optimisation parameters $\vec{\alpha}$ move the raceline points along the corresponding normal vectors $\vec{n}$.

The location of a third order spline and its first and second derivative with respect to t are as follows:

$$x_i(t) = a_i + b_i t + c_i t^2 + d_i t^3, \tag{4}$$

$$x_i'(t) = b_i + 2c_i t + 3d_i t^2, \tag{5}$$

$$x_i''(t) = 2c_i + 6d_i t \quad \text{and} \tag{6}$$

$$t_i(s) = \frac{s - s_{i0}}{\Delta s_i} \tag{7}$$

where t is the normalised curvilinear parameter along one spline segment starting at the distance s_{i0}. Therefore, $0 \leq t_i \leq 1$ holds. The spline interpolation requires that consecutive splines share their respective beginning and endpoint as well as their first and second derivative at these points. The latter fact ensures smooth curvature along the interpolation. Complying with these constraints, the spline parameters a_i, b_i, c_i and d_i are obtained from the solution of a linear equation system of the form $Az = b$. The matrix A and the vector b formalise the above constraints. Since A is full rank by construction of the problem, it can be inverted and leads to a unique solution for the spline coefficients contained in z.

For the minimum curvature optimisation, we want to minimise the discrete squared curvature κ_i^2 of the splines summed along the raceline with N points/splines as follows:

$$\begin{aligned} &\underset{[\alpha_1 \ldots \alpha_N]}{\text{minimize}} && \sum_{i=1}^{N} \kappa_i^2(t) \\ &\text{subject to} && \alpha_i \in \left[\alpha_{i,\min}, \alpha_{i,\max}\right] \quad \forall\, 1 \leq i \leq N. \end{aligned} \tag{8}$$

The L2 norm is chosen because of its desirable properties in terms of numerical optimisation and elegant formulation of the resulting optimisation problem. From the spline representation it follows that it is most convenient and efficient to evaluate the splines at $t = 0$. These are chosen as discretisation points evaluated in the problem above. Fixing $t = 0$ and dropping the spline parameter t for notational convenience, the curvature at

the discrete evaluation points can be expressed as [70, p.373]:

$$\kappa_i = \frac{x_i' y_i'' - y_i' x_i''}{\left(x_i'^2 + y_i'^2\right)^{\frac{3}{2}}} \quad \text{and} \tag{9}$$

$$\kappa_i^2 = \frac{x_i'^2 y_i''^2 - 2x_i' x_i'' y_i' y_i'' + y_i'^2 x_i''^2}{\left(x_i'^2 + y_i'^2\right)^3}. \tag{10}$$

Braghin et al. [30] uses a simplified curvature definition omitting the central part $2x_i' x_i'' y_i' y_i''$. This leads to a suboptimal solution in terms of the exact minimum curvature problem. In contrast, our formulation of the optimisation problem is based on the exact curvature definition, which results in improved optimisation results and allows us to introduce curvature constraints later.

Inserting (10) into the optimisation problem stated in (8), we obtain

$$\begin{aligned} \underset{[\alpha_1 \dots \alpha_N]}{\text{minimize}} \quad & \vec{x}''^T P_{xx} \vec{x}'' + \vec{y}''^T P_{xy} \vec{x}'' + \vec{y}''^T P_{yy} \vec{y}'' \\ \text{subject to} \quad & \alpha_i \in \left[\alpha_{i,\min}, \alpha_{i,\max}\right] \quad \forall\, 1 \leq i \leq N \end{aligned} \tag{11}$$

by defining $\vec{x}'' = [x_1'' \quad \dots \quad x_N'']^{\mathrm{T}}$ and $\vec{y}'' = [y_1'' \quad \dots \quad y_N'']^{\mathrm{T}}$ as the vector notation for the second derivative of the coordinates at each discretisation point.

The matrices P_{xx}, P_{xy} and P_{yy} can be written as:

$$P_{xx} = \begin{bmatrix} \frac{y_1'^2}{\left(x_1'^2 + y_1'^2\right)^3} & 0 & \dots & 0 \\ 0 & \frac{y_2'^2}{\left(x_2'^2 + y_2'^2\right)^3} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{y_N'^2}{\left(x_N'^2 + y_N'^2\right)^3} \end{bmatrix}, \tag{12}$$

$$P_{xy} = \begin{bmatrix} \frac{-2x_1' y_1'}{\left(x_1'^2 + y_1'^2\right)^3} & 0 & \dots & 0 \\ 0 & \frac{-2x_2' y_2'}{\left(x_2'^2 + y_2'^2\right)^3} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{-2x_N' y_N'}{\left(x_N'^2 + y_N'^2\right)^3} \end{bmatrix} \quad \text{and} \tag{13}$$

$$P_{yy} = \begin{bmatrix} \frac{x_1'^2}{\left(x_1'^2 + y_1'^2\right)^3} & 0 & \dots & 0 \\ 0 & \frac{x_2'^2}{\left(x_2'^2 + y_2'^2\right)^3} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{x_N'^2}{\left(x_N'^2 + y_N'^2\right)^3} \end{bmatrix}. \tag{14}$$

The above structure already suggests to treat the problem as a QP, under the additional assumption that the matrices P_{xx}, P_{xy} and P_{yy} are constant. This can be seen as approximately true because x' and y' are approximately constant since path heading only changes slightly compared to the reference line as long as it is tightly constrained by the inner and outer track boundaries. Therefore, we obtain x' and y' from the case $\alpha = 0$ which corresponds to the reference line. The process can be viewed as linearisation of the optimisation problem along the reference line.

We now need to express the second derivatives x'' and y'' in terms of the optimisation parameters $\vec{\alpha}$. For $t = 0$ x'' evaluates to:

$$x_i''(t = 0) = 2c_i. \tag{15}$$

The spline coefficients that resemble the second derivative at $t = 0$ can be extracted from the solution vector z, which can be expressed in terms of the inverse of the linear equation system matrix A and the constant vector b, via the extraction matrix $A_{\mathrm{ex,c}}$:

$$\begin{bmatrix} x_1'' \\ x_2'' \\ \vdots \\ x_N'' \end{bmatrix} = 2A_{\mathrm{ex,c}}A^{-1} \left(\underbrace{\begin{bmatrix} p_{1,x} \\ p_{2,x} \\ 0 \\ 0 \\ p_{2,x} \\ p_{3,x} \\ 0 \\ 0 \\ \vdots \\ 0 \\ p_{N,x} \\ p_{1,x} \\ 0 \\ 0 \end{bmatrix}}_{\vec{q}_x} + \underbrace{\begin{bmatrix} n_{1,x} & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & n_{2,x} & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & n_{2,x} & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & n_{3,x} & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & n_{N,x} \\ n_{1,x} & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix}}_{M_x} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{bmatrix} \right). \tag{16}$$

The first derivatives x_i' and y_i' are calculated in a similar manner. Reformulating (16) in vector notation gives us

$$\vec{x}'' = T_c \vec{q}_x + T_{n,x} \vec{\alpha}, \tag{17}$$

where $T_c = 2A_{\mathrm{ex,c}} A^{-1}$ and $T_{n,x} = 2A_{\mathrm{ex,c}} A^{-1} M_x$. T_c is equal for the x- and y-coordinate splines and therefore not distinguished. Inserting (17) and the corresponding version for y in (11) we finally obtain

$$\begin{aligned} &\underset{[\alpha_1 \ldots \alpha_N]}{\text{minimize}} \quad \vec{\alpha}^T (H_x + H_{xy} + H_y) \vec{\alpha} + \left(f_x + f_{xy} + f_y\right)^T \vec{\alpha} + \text{const}, \\ &\text{subject to} \quad \alpha_i \in \left[\alpha_{i,\min}, \alpha_{i,\max}\right] \quad \forall\, 1 \leq i \leq N \end{aligned} \tag{18}$$

where

$$\begin{aligned} H_x &= T_{n,x}^T P_{xx} T_{n,x}, \\ H_{xy} &= T_{n,y}^T P_{xy} T_{n,x}, \\ H_y &= T_{n,y}^T P_{yy} T_{n,y}, \\ f_x &= 2\, T_{n,x}^T P_{xx}^T T_c \vec{q}_x, \\ f_{xy} &= T_{n,y}^T P_{xy}^T T_c \vec{q}_x + T_{n,x}^T P_{xy}^T T_c \vec{q}_y, \\ f_y &= 2\, T_{n,y}^T P_{yy}^T T_c \vec{q}_y \quad \text{and} \\ \text{const} &= \vec{q}_x^T T_c^T P_{xx} T_c \vec{q}_x + \vec{q}_y^T T_c^T P_{xy} T_c \vec{q}_x + \vec{q}_y^T T_c^T P_{yy} T_c \vec{q}_y. \end{aligned}$$

Neglecting the constant term const, this can be reformulated in terms of a standard QP problem:

$$\begin{aligned} &\underset{[\alpha_1 \ldots \alpha_N]}{\text{minimize}} \quad \frac{1}{2} \vec{\alpha}^T H \vec{\alpha} + f^T \vec{\alpha} \\ &\text{subject to} \quad E \vec{\alpha} \leq k. \end{aligned} \tag{19}$$

3.4.2. Adaption of the optimisation problem

The presented implementation of the optimisation problem is quite sensitive to a noisy reference line, for example due to small jumps along a straight originating in map discretisation. Therefore, the reference line is preprocessed in four steps before it is handed over to the optimisation problem, see Figure 6. Firstly, the original reference line is linearly interpolated to a small step size. Based on this, a spline approximation is calculated to remove the noise. Afterwards, the track widths must be corrected due to the slight displacement of the spline approximation compared to the reference line. Finally, a spline interpolation is performed to obtain the desired step size for the optimisation problem, e.g. 3.0 m. Figure 7 compares the curvature profiles for the

original and the preprocessed reference lines. It can be seen, that the latter is much smoother.

To be able to consider the maximum drivable curvature constrained by the car's steering design $\kappa_{\text{bound}} = 0.12\,\text{rad/m}$, we introduced curvature constraints into the optimisation problem (19). Therefore, curvature is divided into a static curvature part κ_{ref} originating in the reference line and a variable curvature part κ_{var} originating in the shift along the normal vectors:

$$|\kappa_{\text{ref}} + \kappa_{\text{var}}| \leq \kappa_{\text{bound}}. \tag{20}$$

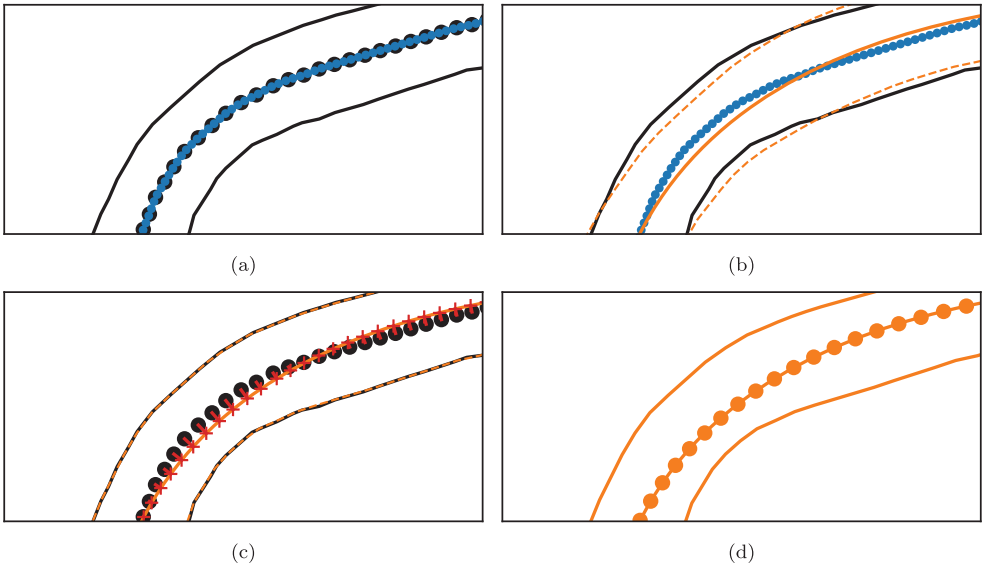

Figure 6. Four steps of reference line preprocessing (top left to bottom right). (a) Linear interpolation (b) Spline approximation (c) Correction of track widths (d) Spline interpolation

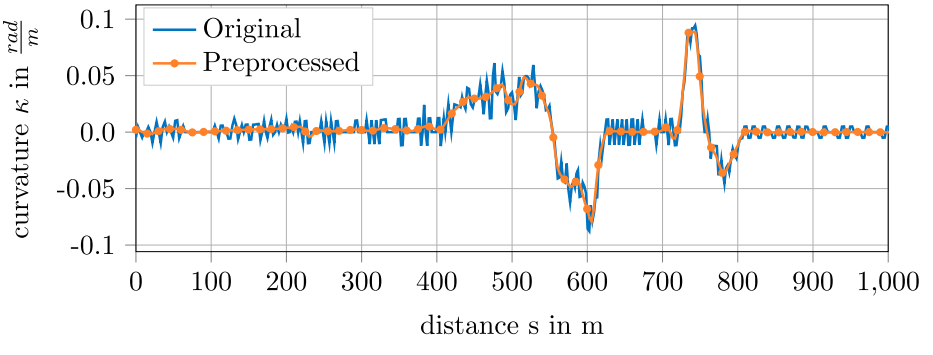

Figure 7. Curvature profiles of the original and the preprocessed reference lines (first 1000 m of the Berlin Formula E track).

For the sake of brevity, only the derivation of the upper boundary is described. The lower boundary is set up accordingly. Defining

$$Q_x = \begin{bmatrix} \frac{y'_1}{\left(x'^2_1 + y'^2_1\right)^{\frac{3}{2}}} & 0 & \dots & 0 \\ 0 & \frac{y'_2}{\left(x'^2_2 + y'^2_2\right)^{\frac{3}{2}}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{y'_N}{\left(x'^2_N + y'^2_N\right)^{\frac{3}{2}}} \end{bmatrix} \quad \text{and} \tag{21}$$

$$Q_y = \begin{bmatrix} \frac{x'_1}{\left(x'^2_1 + y'^2_1\right)^{\frac{3}{2}}} & 0 & \dots & 0 \\ 0 & \frac{x'_2}{\left(x'^2_2 + y'^2_2\right)^{\frac{3}{2}}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{x'_N}{\left(x'^2_N + y'^2_N\right)^{\frac{3}{2}}} \end{bmatrix}, \tag{22}$$

we can calculate the curvature of the reference line by

$$\kappa_{\text{ref}} = Q_y \, T_c \, \vec{q}_y - Q_x \, T_c \, \vec{q}_x. \tag{23}$$

The variable part of the curvature can be stated as

$$\kappa_{\text{var}} = \left(Q_y \, T_{n,y} - Q_y \, T_{n,x}\right) \, \vec{\alpha}. \tag{24}$$

Bringing (20) into the standard form as shown in (19), κ_{ref} is considered within the right side of the inequality k_κ and κ_{var} within the left side $E_\kappa \, \vec{\alpha}$. We finally obtain

$$E_\kappa = Q_y \, T_{n,y} - Q_y \, T_{n,x} \quad \text{and} \tag{25}$$

$$k_{\kappa,\text{upper}} = \kappa_{\text{bound}} - \kappa_{\text{ref}}. \tag{26}$$

E_κ is the same for the upper and lower boundary condition and must therefore not be distinguished.

Setting up the original optimisation problem before, we assumed the matrices P_{xx}, P_{xy} and P_{yy} being constant. This is approximately true as long as the heading of the optimised path differs only slightly from the reference line. However, in corner entries and exits the validity of this assumption reduces as can be seen in Figure 4. There, the curvature calculated based on the linearisation along the reference line differs significantly from the real curvature. This leads to suboptimal solutions and non-compliance with the given curvature constraints. To overcome this, we implemented a loop around the optimisation problem

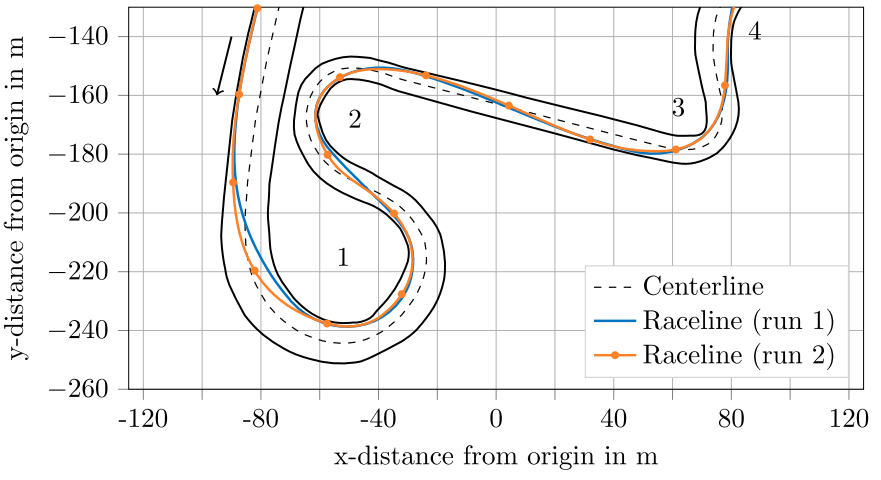

Figure 8. Comparison between the racelines after the first and second optimisation runs, where the second problem was solved based on the solution of the first run.

taking advantage of the reference line definition. In the first iteration, the optimisation problem is solved as described above. Starting from the second iteration, the reference line is replaced by the previous solution. As the solution is of limited validity during the first iterations, the solution vector $\vec{\alpha}$ is multiplied by a factor of $\frac{1}{3}$ respectively $\frac{2}{3}$ in iterations one and two to limit the displacement from the reference line and therefore the inaccuracies of the curvature calculation. When replacing the reference line by the optimisation result the track widths must of course be adapted for the further iterations as it was already done after the spline approximation. In addition, the resulting path must be (spline) interpolated in order to keep equal step sizes that are required by the optimisation problem formulation. The termination criterion for the loop is based on the maximum difference between the curvature profiles calculated based on linearisations along the original reference line and along the result itself. We set it to $\Delta\kappa_{\max} = 0.005\,\mathrm{rad/m}$.

The result for the first and second optimisation run (without reduction of $\vec{\alpha}$ for illustration purposes) is displayed in Figure 8. As expected, the path differs strongly at the corner entries and exits since the validity of the linearisation decreases in those zones due to the deviating heading. Figure 9 gives an impression of how much the real curvature exceeds the curvature considered within the optimisation problem if the iterative invocation would not be executed. For the Berlin track, the termination criterion is fulfilled after the fourth iteration. The final optimisation result provides a very smooth curvature profile, which is essential for the subsequent velocity profile calculation as well as for the vehicle controller.

3.5. Generation of a velocity profile

Within velocity profile calculation, the car's acceleration limits must be considered as defined by tires, motors and brakes. Again, a fast calculation time is of interest for our application. Another requirement is to exploit the potential of the tires, especially in combined slip conditions. A forward-backward-solver was implemented for this purpose. It calculates two velocity profiles, one forward and one backward, that are then intersected.

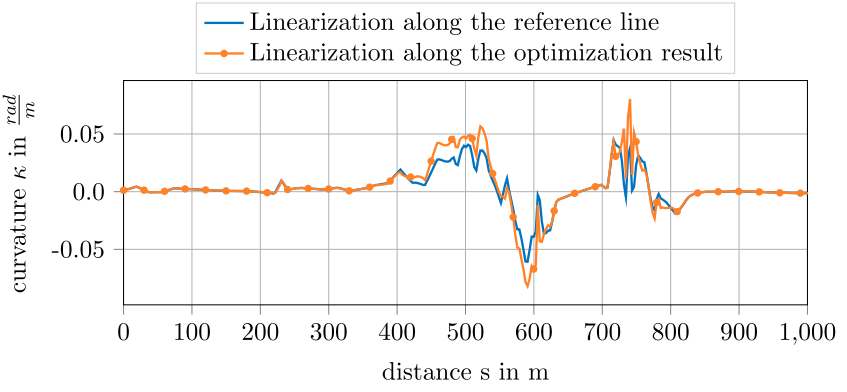

Figure 9. Comparison between the calculated curvature profiles after the first optimisation run based on linearisations around the original reference line and around the result itself (first 1000 m of the Berlin Formula E track).

The functional principle of intersecting different partial profiles is widely used in literature, e.g. [15,28–30], and is therefore only briefly summarised.

The basis of the approach is a ggv-diagram containing the longitudinal and lateral acceleration limits of the car at different velocities. The velocity dependency results from aerodynamic effects such as drag and lift. Thus, the diagram is a simplified substitute for the vehicle dynamics and the driving resistances. In a first step, the solver calculates an estimate of the velocity profile based on the smallest lateral acceleration potential of the car $a_{\mathrm{y,min}}$ at any velocity and the curvature profile of the raceline. The estimated velocity profile is then cut the top speed limit of the vehicle. Afterwards the forward calculation procedure is started. It modifies the velocity profile in such a way that it keeps the positive longitudinal as well as the lateral acceleration limits of the car. This is repeated in the backward procedure with the negative longitudinal acceleration limits.

The acceleration profile can then be calculated by

$$a_{\mathrm{x},i} = \frac{v_{\mathrm{x},i+1}^2 - v_{\mathrm{x},i}^2}{2\, l_i}. \tag{27}$$

Figure 10 shows the velocity and acceleration profiles used for the Berlin track. The maximum velocity was limited to 150 km/h.

3.6. Preparation for control

While driving, a path-matching algorithm is required to match the car's position to the global race trajectory because the car cannot follow the planned trajectory exactly due to control errors and external disturbances. For this purpose, the trajectory point with the minimum distance to the car's centre of gravity is used in the first step after starting the car. For all subsequent steps, we work with the minimum distance point that lies within a search window around the expected vehicle position $\hat{s}_{i+1}$ calculated by

$$\hat{s}_{i+1} = s_i + v_i\,(t_{i+1} - t_i)\,. \tag{28}$$

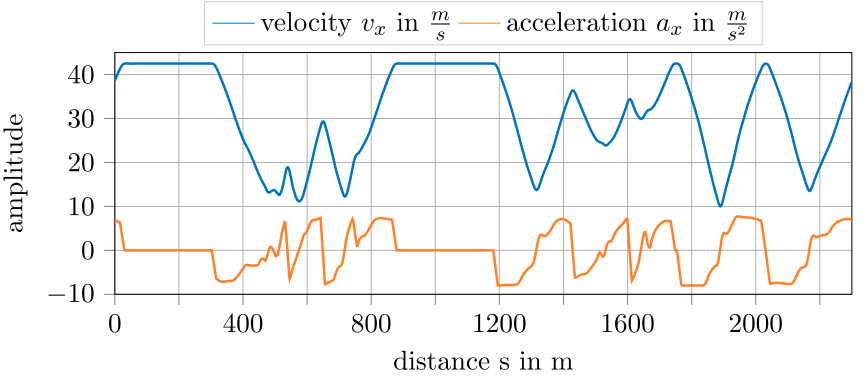

Figure 10. Longitudinal velocity and acceleration profile for a flying lap on the Berlin Formula E track. The maximum velocity was limited to 150 km/h.

This guarantees that the correct trajectory part is found, even if two parts lie close together, e.g. straight and back straight as in Figure 13.

As already stated, in every cycle we send an emergency trajectory to the controller. The path of this trajectory is equal to the normal one whereas the velocity and acceleration profile are modified such that the car comes to a standstill as quickly as possible, i.e. maximum longitudinal deceleration is used. Since the same velocity profile planner is used, the handling limits of the vehicle are taken into account as for the normal trajectory. However, the look-ahead time of this trajectory is set much higher to have enough space to come to a standstill even in difficult situations. Furthermore, the ggv-diagram for the emergency case allows slightly higher accelerations than the normal one.

3.7. Controller design

The overall control structure is depicted in Figure 11. The concept encapsulates all dynamics that are closely related to a specific vehicle using a low-level controller. The high-level controller is responsible for providing a suitable path-tracking functionality. The separation increases the system's robustness and eases the transfer of the trajectory-tracking controller to other vehicles. The difficulties of path tracking control for a race car arise from the high nonlinearity of the dynamics and the complex control allocation (split between feed forward and feedback parts and lateral and longitudinal couplings) at the limits of handling.

Resulting from the low-level dynamics abstraction layer, the basic concept of the trajectory controller relies on the description of a point-mass moving along a trajectory. Lateral and longitudinal control are designed separately in the following based on the assumption that the vehicle speed is both constant and known. This enables a gain-scheduling design for the lateral controller based on the vehicle velocity. Defining the lateral path deviation d as the control error, the control design system can be approximated by a double integrator

$$\ddot{d} = a_{y,c} = \kappa_c v^2, \tag{29}$$

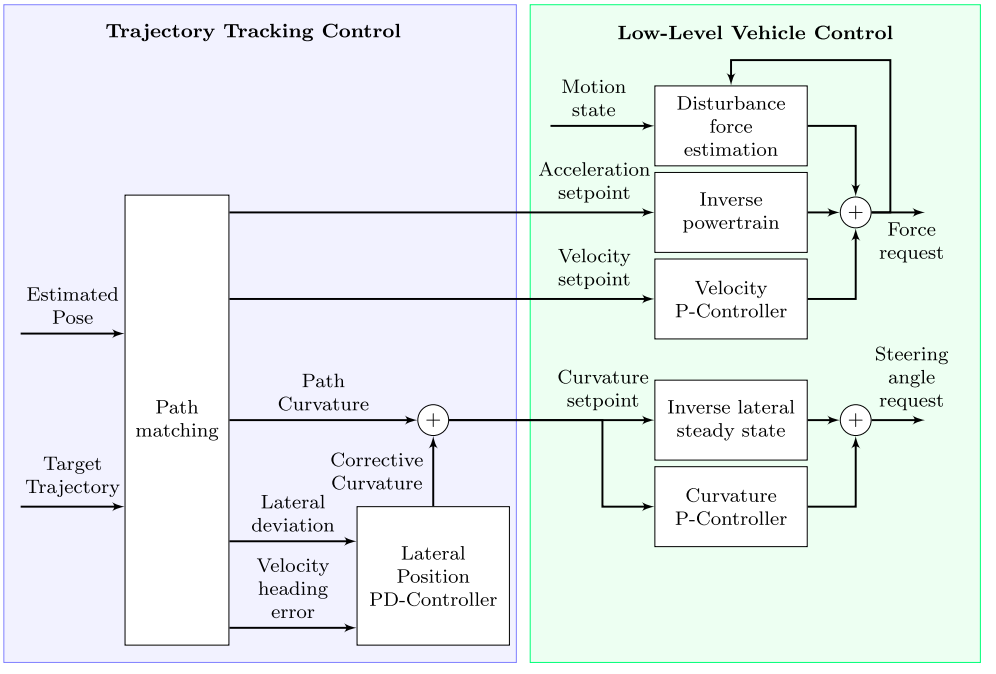

Figure 11. Control concept.

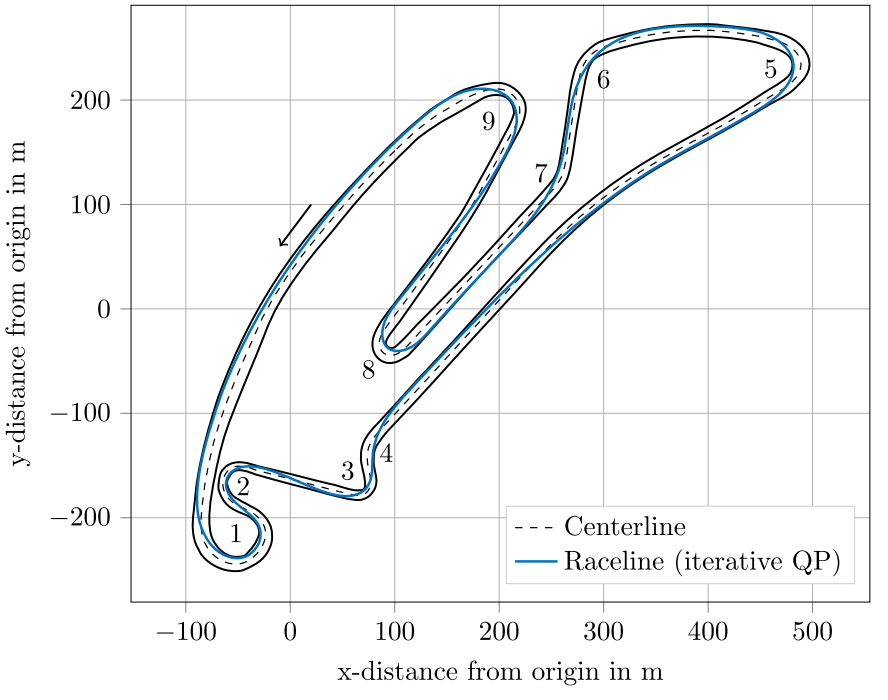

Figure 12. Optimised raceline for the Berlin Formula E track.

where $a_{y,c}$ is the lateral acceleration of the vehicle in path coordinates [55]. Assuming steady-state cornering, we can derive the second equality and use the corrective curvature κ_c as a virtual control input. The feedback law with the undamped eigenfrequency ω_0 and the damping constant D is given by

$$\kappa_{\mathrm{c}} = -\frac{1}{v^2}(\omega_0^2 d + 2D\omega_0 \dot{d}). \tag{30}$$

The resulting closed loop dynamics are independent of the velocity v. Stability of the controller is guaranteed by construction, since quadratic polynomials fulfil the Hurwitz stability criterion if all coefficients are positive. To prevent numerical derivation during implementation of (30), the lateral deviation derivative can be calculated from the difference between the trajectory heading ψ and the vehicle velocity heading, calculated from the vehicle heading ψ_{V} and the side slip angle β,

$$\dot{d} = v \sin(\psi - (\psi_{\mathrm{V}} + \beta)). \tag{31}$$

The low-level vehicle control consists of a curvature controller and a velocity controller. Since the sensors cannot measure curvature directly and the side slip angle derivatives are hard to estimate reliably, it is approximated using a steady state assumption

$$\kappa = \frac{\dot{\beta} + \dot{\psi}_{\mathrm{V}}}{v} \approx \frac{\dot{\psi}_{\mathrm{V}}}{v}. \tag{32}$$

The controller itself is designed to be a proportional feedback controller with gain K_κ. Its main purpose is to add counter-steering in case of instabilities. This becomes clear due to the simplified curvature approximation, which actually would allow a reformulation as a yaw rate controller with velocity dependent setpoints. Due to the near neutral setup of the DevBot, the inverse steady-state model could be implemented via a linear relationship between the target curvature κ_T and the steering angle δ using the wheelbase l_{V}. The overall control law is then given by

$$\delta = K_\kappa \, (\kappa_{\mathrm{T}} - \kappa) + \kappa_{\mathrm{T}} \, l_{\mathrm{V}}. \tag{33}$$

The general structure of the velocity controller is similar. It uses proportional feedback to achieve stability and tracking of the velocity set point and combines this with an inverse powertrain model that captures an estimate of the driving resistances and the forces required to overcome inertia. In contrast to the curvature controller, the high level of uncertainty of the driving resistances requires the longitudinal controller to apply additional integral action. It is incorporated by the use of a disturbance observer [71], which outputs an estimate $\hat{F}_{\mathrm{d}}$ for the unmodelled forces acting upon the system. This mechanism can also account for mismatch between the setpoint and the applied force (control variable uncertainty). It was implemented using a steady-state Kalman Filter under the assumption that a constant disturbance acts upon the system. The resulting control law can be written as

$$F = K_v(v_{\mathrm{T}} - v) + F_{\mathrm{PT}}(a_{x,\mathrm{T}}) + \hat{F}_{\mathrm{d}}, \tag{34}$$

with the powertrain model depending on the feed forward trajectory acceleration $a_{x,\mathrm{T}}$

$$F_{\mathrm{PT}}(a_{x,\mathrm{T}}) = \left(m + \frac{I_{\mathrm{F}}}{r_{\mathrm{F}}^2} + \frac{I_{\mathrm{R}}}{r_{\mathrm{R}}^2}\right) a_{x,\mathrm{T}} + 0.5\rho c_{\mathrm{w}} v^2, \tag{35}$$

where I_{F} and I_{R} depict the front and rear powertrain inertia and r_{F} and r_{R} the corresponding tire radii. ρ is the air density and c_{w} the effective drag coefficient with the reference area already included.

4. Results and discussion

In this section we present the results obtained by the suggested methodology and discuss the approach critically.

4.1. Minimum curvature trajectory results

As already stated in Section 3.4, the computation time from centreline import to trajectory output for the Berlin track is 18 s with a discretisation step size of 3.0 m and a raceline length of approximately 2300 m. Each of the four QP iterations is solved in about 0.85 s. The velocity profile calculation requires 65 ms. The rest of the computation time is primarily spent in spline calculations and interpolations. The programme is implemented in Python 3 using the numerical math library NumPy. The entire raceline for the Berlin track is shown in Figure 12.

During the Berlin event in May 2018 an older version of the QP was used, which had no iteration loops yet. The lap time calculated back then was within half a second of the 91.59 s we achieved in the flying lap in the real event. With the improved optimisation problem, we calculate a lap time of 88.41 s taking into account 2.5 m safety distance to the track boundaries as we did in 2018. 86.13 s is the lap time calculated without safety margin. We expect to get close to it in the next race. This should be achievable as we will be

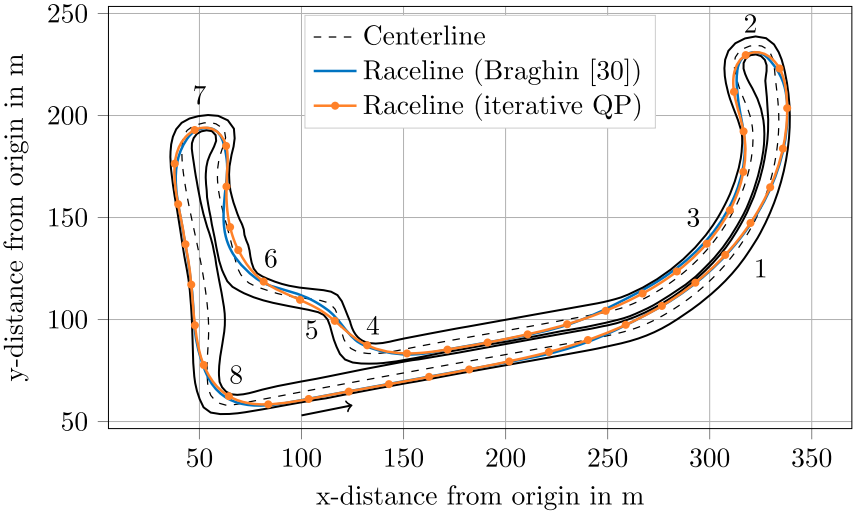

Figure 13. Comparison of the racelines for the Upper Heyford test track between the original formulation in [30] and the iterative QP formulation.

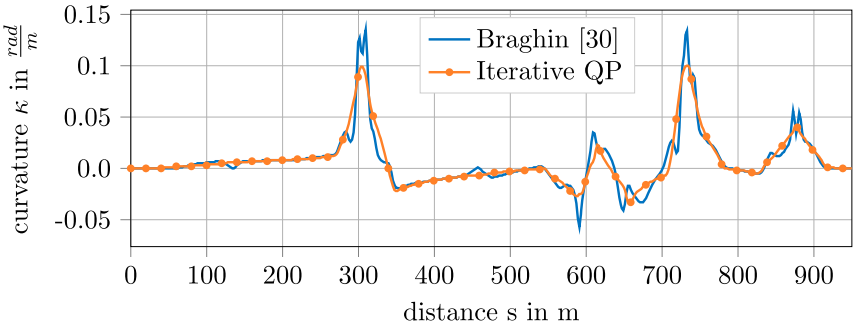

Figure 14. Comparison of the curvature profiles for the Upper Heyford test track between the original formulation in [30] and the iterative QP formulation.

able to reduce the safety margin and as all the lap times are calculated on the basis of the same ggv-diagram, which exploits about 80% of the acceleration limits of the car. A further increase of the acceleration limits in the ggv-diagram will probably not be reached in reality due to the simplified vehicle dynamics of the point-mass model in the velocity profile generation.

Figures 13 and 14 visualise the differences of the racelines and curvature profiles between the original problem stated by [30] and the iterative QP formulation for the Upper Heyford test track. It includes a few tight corners and is therefore better suited for the comparison than the Berlin track. Both algorithms are fed with the same smoothed reference line input. The main differences between the racelines can be seen in the corners 2, 5, 6, 7 and 8. Here, the iterative QP makes better use of the track width and therefore avoids unnecessary deflections of the raceline. The differences are even more obvious in the curvature profiles. The iterative QP result shows both a smaller peak curvature and a smoother curve. This results in a calculated lap time of 43.54 s compared to 45.55 s for the original implementation.

4.2. Control parameter effects

In general, the controller must balance good disturbance rejection against oscillatory behaviour and noise amplification introduced by high gains. The path tracking parameters have been set to $\omega_0 = 2$ and $D = 0.53$, while the velocity feedback gain was set to $K_v = 1000$ and the curvature feedback gain to $K_\kappa = 0.3$. The tracking results for the Berlin Formula E track are depicted in Figure 15. The achieved accelerations are depicted in the measured gg-Diagram in Figure 16. It can be seen that the car does not only achieve the claimed accelerations with respect to the longitudinal or lateral direction, but also in the combined setting.

The main reason for the large path tracking errors at some areas of the track are the unmodelled nonlinearities in the vehicle steady-state response to the steering angle input. This could be improved by incorporation of an integral action into the lateral tracking controller. However, this might pose difficulties in case of heavy understeering which might lead to an effect similar to wind-up of the controller. This could not be prevented by standard anti-wind up mechanisms as the maximum steering angle which adds an effective

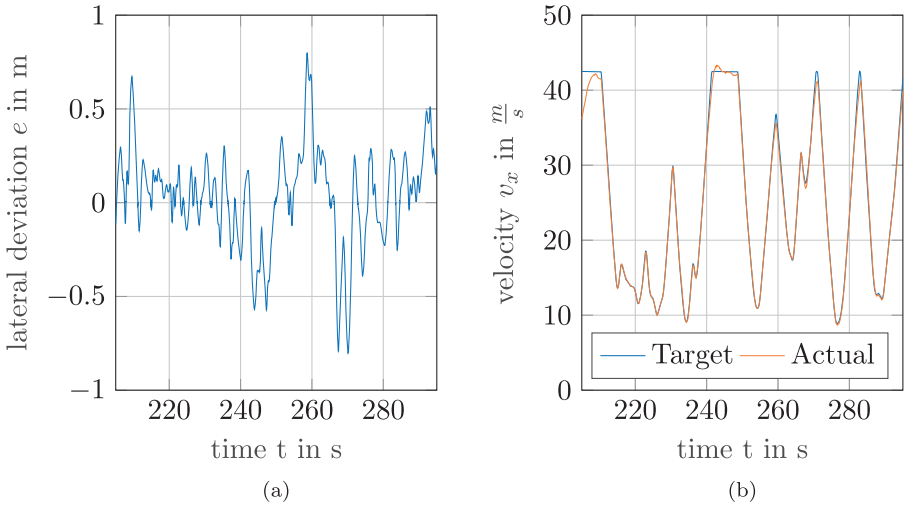

Figure 15. Performance of the overall system at the Formula E track in Berlin 2018. (a) Path tracking error (b) Velocity tracking

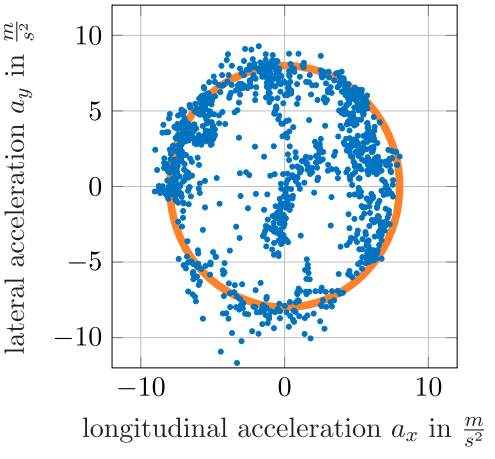

Figure 16. gg-Diagram of the DevBot during its flying lap at the Formula E circuit in Berlin 2018. The raw sensor data was filtered by a moving average filter with a window length of 100 ms before plotting. The red circle specifies the gg-Diagram assumed for planning the trajectory.

lateral acceleration is not known beforehand. Another difficulty during the tuning of the path tracking controllers is that the vehicle response becomes underdamped at high velocities. This poses an upper limit on the maximum achievable speed for a certain closed loop frequency ω_0. A method to circumvent this would be the application of a fine tuned yaw rate controller, which ensures comparable dynamic behaviour over the whole velocity and lateral acceleration range. The velocity tracking error stays below 1 m/s most of the time. The deviations at high speeds result from dynamic torque limitations imposed by the powertrain but not modelled within the controller design.

During the development it was of great interest, how accurate the feed forward law must be to be able to achieve the required control quality. Following [72] one can calculate an

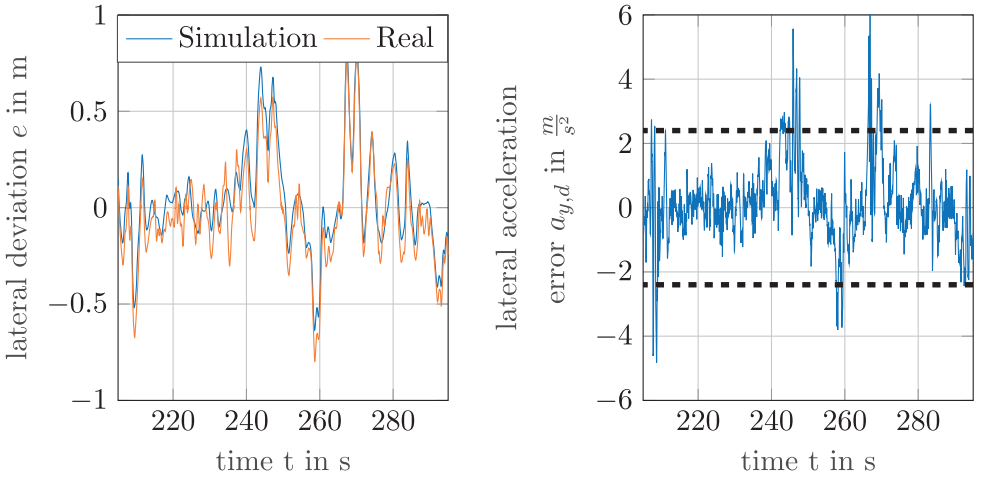

Figure 17. Results of controller disturbance rejection capabilities. (a) Validation of the linear closed loop model used for controller design using the dataset and control parameters from the Berlin Formula E track. (b) Lateral acceleration disturbance calculated based on the trajectory, control request and measured lateral acceleration. The dashed lines depict the maximum allowed disturbance level calculated using the L1-norm of the closed-loop system and a maximum control error of 0.8 m.

upper bound on the allowed disturbance input Δa_y based on the L1-norm of the linearised error system

$$\|G_c\|_1 = \frac{\|e\|_\infty}{\|\Delta a_y\|_\infty}, \tag{36}$$

where $\|\cdot\|_\infty$ represents the signal's peak value. To be meaningful, it is required that the assumptions made earlier about the system dynamics hold at least approximately. This can be checked by calculating the disturbance acting upon the closed-loop system from the sensor data and forward simulation of the linear closed loop system. The disturbance acceleration $a_{y,\mathrm{d}}$ is calculated from the target trajectory acceleration $a_{y,\mathrm{T}}$, the corrective control acceleration $a_{y,\mathrm{C}}$ and the measured acceleration a_y as follows:

$$a_{y,\mathrm{d}} = a_y - a_{y,\mathrm{T}} - a_{y,\mathrm{C}} = a_y - (\kappa_\mathrm{T} + \kappa_\mathrm{C})\, v^2 \tag{37}$$

The resulting disturbance acceleration is used as a system input to the closed loop system resulting from (29) and (30). The predicted lateral control error is compared to the measured control error in Figure 17(a). Using the maximum error of 0.8 m and the L1 system norm of the closed loop system of 0.33 (calculated based on the approach of [72]), it follows that the maximum peak on the acceleration disturbance should stay below 2.4 m/s for all time. This requirement holds nearly everywhere on the track. However, it is violated with a few peaks. This is not harmful, as the derived result is a worst-case bound. It would be reached for example by a step-like signal.

4.3. Discussion of the approach

With hindsight, the chosen methodology proved to be a good and reliable way for the planning and control software of an autonomous race car. The workflow from centreline

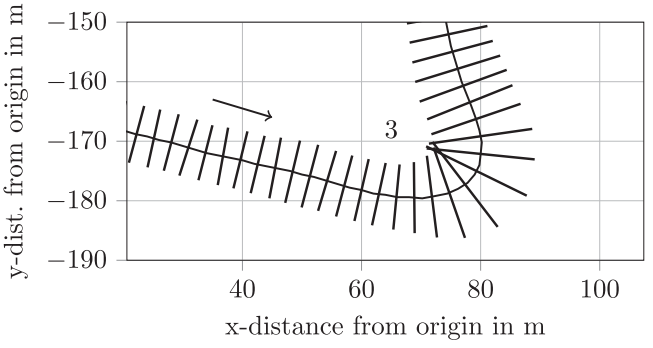

Figure 18. Reference line normals crossing at $x = 72$ m, $y = -171$ m due to a small corner radius in combination with a large track width.

to control is easy to adapt to different race environments and showed robust performance on various tracks. The computation times remain low for small testing tracks as well as for a whole Formula E track. Furthermore, reasonably fast lap times are achieved.

For the Upper Heyford test track displayed in Figure 13, a lap time improvement of 4.4% was reached compared to the original formulation of the optimisation problem, which is a lot in motorsports. This was achieved by the extension of the optimisation problem to approximate the real curvature as well as by the introduction of the iterative invocation of the QP. Together with the spline approximation for the reference line input the latter significantly improves robustness when using real world tracks with imperfect reference lines. It also allowed us to introduce curvature constraints into the optimisation problem such that the result is drivable with a real car. This is especially important on tighter tracks containing 180° hairpin corners, e.g. Formula E tracks.

A disadvantage arose in the context of the curvilinear coordinate system. In tight corners it can happen that several normals are crossed at the track boundary as shown in Figure 18. This happens if a small corner radius coincides with a large track width and if the discretisation step size between two points is small enough. This ambiguity can appear for all approaches based on curvilinear coordinate systems. We solve it by raising the smoothing factor of the spline approximation. This results in the reference line moving towards the inside of the corner, thus avoiding the problem, cp. Figure 6(b).

5. Conclusion and outlook

In this paper, we presented our approach to planning a minimum curvature trajectory for an autonomous race car and introduced a controller design that allows it to follow the trajectory at the handling limits. Improvements to the optimisation problem were introduced that resulted in a significant lap time improvement and made it a lot more robust to real world application. The software design proved to work at a velocity of 150 km/h and 80% of the vehicle's acceleration potential. The whole software pipeline can easily be adapted to various racetracks. The results illustrate the capabilities of the pipeline and its suitability for autonomous motorsports.

In the future, we plan to retain the general approach. The global planner will be complemented by a local trajectory planner to allow overtaking and evasion manoeuvres taking

static and dynamic objects on the racetrack into account. This is where we will be able to take advantage of the modular system structure. Furthermore, some effort will be put into the software part that is currently executed offline so that it can be executed online on the car, e.g. to re-plan the raceline due to a new static object appearing during the race. Due to its fast calculation time, the minimum curvature optimisation can be performed within reasonable time also on weak CPUs and is therefore well suited for this purpose. In parallel with this, we will further evaluate other optimisation approaches that consider a more detailed vehicle dynamics model and directly minimise the lap time.

The entire Python code used in the TUM Roborace team for global trajectory optimisation will be published under an open source license on GitHub after publication of this paper.[1]

Note

1. https://github.com/TUMFTM/global_racetrajectory_optimization.

Acknowledgments

Alexander Heilmeier as the first author is responsible for the trajectory planning software. He initiated the idea of the paper and contributed to the analysis and modification of the optimisation problem, to the velocity profile generation as well as to the behaviour state machine. Furthermore, he contributed essentially to the overall system design including the distributed calculation architecture between offline optimisation and online processing. Alexander Wischnewski is responsible for the control software. He is the main contributor to the control system design and contributed essentially to the reformulation of the optimisation problem. Leonhard Hermansdorfer is responsible for processing and preparing the map and centerline extrication for the trajectory optimisation problem. Johannes Betz contributed to the overall system design and the mapping software. Markus Lienkamp and Boris Lohmann contributed equally to the conception of the research project and revised the paper critically for important intellectual content. They gave final approval of the version to be published and agree to all aspects of the work. As guarantors, they accept responsibility for the overall integrity of the paper. Research was supported by the basic research fund of the Chair of Automotive Technology of the Technical University of Munich.

We would like to thank Roborace for the possibility of demonstrating our software on their car as well as for their support during the testing sessions and the Berlin event. We would also like to thank Fabian Christ, who implemented the minimum time optimisation as well as the spline approximation in his master thesis within the project.

Disclosure statement

No potential conflict of interest was reported by the authors.

ORCID

Alexander Heilmeier http://orcid.org/0000-0002-2442-3652

References

[1] Betz J, Wischnewski A, Heilmeier A, et al. What can we learn from autonomous level-5 motorsport?. In: Pfeffer P, editor. 9th International Munich Chassis Symposium 2018. Wiesbaden: Springer Fachmedien Wiesbaden; 2019. p. 123–146.

[2] The Fast and the Driverless: Munich Team Takes Home Roborace Victory [Internet]. [cited 2018 June 29]; 2018. Available from: https://blogs.nvidia.com/blog/2018/06/29/fast-and-driverless-munich-rob orace-victory.
[3] Grisetti G, Stachniss C, Burgard W. Improved techniques for grid mapping with Rao-Blackwellized particle filters. IEEE Trans Robot. 2007;23(1):34–46.
[4] Hess W, Kohler D, Rapp H, et al. Real-Time loop closure in 2D LIDAR SLAM. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). 2016. p. 1271–1278.
[5] Thrun S, Montemerlo M, Dahlkamp H, et al. Stanley: the robot that won the DARPA grand challenge. J Field Robot. 2006;23(9):661–692.
[6] Urmson C, Anhalt J, Bae H, et al. Autonomous driving in urban environments: boss and the urban challenge. J Field Robot Special Issue on the 2007 DARPA Urban Challenge, Part I. 2008 June;25(8):425–466.
[7] Dolgov D, Thrun S, Montemerlo M, et al. Practical search techniques in path planning for autonomous driving. AAAI Workshop - Technical Report. 2008 01.
[8] Bacha A, Bauman C, Faruque R, et al. Odin: team victortango's entry in the DARPA urban challenge. J Field Robot. 2008 Jan;25:467–492.
[9] John L, Jonathan H, Seth T, et al. A perception-driven autonomous urban vehicle. J Field Robot. 2008;25(10):727–774.
[10] Urmson C, Anhalt J, Bagnell D, et al. Autonomous driving in urban environments: boss and the urban challenge. J Field Robot. 2008 Aug;25(8):425–466.
[11] Kammel S, Ziegler J, Pitzer B, et al. Team AnnieWAY's autonomous system for the 2007 DARPA urban challenge. J Field Robot. 2008;25(9):615–639.
[12] Montemerlo M, Becker J, Bhat S, et al. Junior: the stanford entry in the urban challenge. Berlin, Heidelberg: Springer Berlin Heidelberg; 2009. Chapter 3; p. 91–123.
[13] Kapania NR, Gerdes JC. Design of a feedback-feedforward steering controller for accurate path tracking and stability at the limits of handling. Vehicle Syst Dyn. 2015;53(12):1687–1704.
[14] Laurense V, Goh J, Gerdes J. Path-tracking for autonomous vehicles at the limit of friction. In: 2017 American Control Conference (ACC); 2017 May. p. 5586–5591.
[15] Kapania N, Subosits J, Gerdes J. A sequential two-step algorithm for fast generation of vehicle racing trajectories. J Dyn Syst Meas Control. 2016;138(9):091005.
[16] Liniger A. Path Planning and Control for Autonomous Racing [dissertation]. ETH Zurich; 2018.
[17] Liniger A, Domahidi A, Morari M. Optimization-based autonomous racing of 1:43 scale RC cars. Optim Control Appl Methods. 2014;36(5):628–647.
[18] Serban AC, Poll E, Visser J. A standard driven software architecture for fully autonomous vehicles. In: 2018 IEEE International Conference on Software Architecture Companion (ICSA-C); 2018. p. 120–127.
[19] Paden B, Čáp M, Yong SZ, et al. A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Trans Intell Vehicles. 2016 March;1(1):33–55.
[20] Pendleton SD, Andersen H, Du X, et al. Perception, planning, control, and coordination for autonomous vehicles. Machines. 2017;5(1):6.
[21] Matkan AA, Hajeb M, Sadeghian S. Road extraction from lidar data using support vector machine classification. Photogramm Eng Remote Sensing. 2014;80(5):409–422.
[22] Miao Z, Wang B, Shi W, et al. A semi-automatic method for road centerline extraction from VHR images. IEEE Geosci Remote Sensing Lett. 2014;11(11):1856–1860.
[23] Guan J, Wang Z, Yao X. A new approach for road centerlines extraction and width estimation. In: IEEE 10th International Conference on Signal Processing Proceedings. IEEE; 24.10.2010 - 28.10.2010. p. 924–927.
[24] Cheng G, Wang Y, Xu S, et al. Automatic road detection and centerline extraction via cascaded end-to-end convolutional neural network. IEEE Trans Geosci Remote Sens. 2017;55(6):3322–3337.
[25] Guedri H, Abdallah MB, Nasri F, et al. Computer method for tracking the centerline curve of the human retinal blood vessel. In: 2017 International Conference on Engineering and MIS (ICEMIS). IEEE; 08 May 2017–10 May 2017. p. 1–6.

[26] Sironi A, Lepetit V, Fua P. Multiscale centerline detection by learning a scale-space distance transform. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. IEEE; 23 June 2014–28 June 2014. p. 2697–2704.
[27] Sironi A, Turetken E, Lepetit V, et al. Multiscale centerline detection. IEEE Trans Pattern Anal Mach Intell. 2016;38(7):1327–1341.
[28] Velenis E, Tsiotras P. Optimal velocity profile generation for given acceleration limits: receding horizon implementation. In: Proceedings of the 2005 American Control Conference; Vol. 3; June; 2005. p. 2147–2152.
[29] Brayshaw DL, Harrison MF. A quasi steady state approach to race car lap simulation in order to understand the effects of racing line and centre of gravity location. Proc Inst Mech Eng Part D: J Automob Eng. 2005;219(6):725–739.
[30] Braghin F, Cheli F, Melzi S, et al. Race driver model. Comput Struct. 2008 Jul;86(13–14): 1503–1516.
[31] Katrakazas C, Quddus M, Chen WH, et al. Real-time motion planning methods for autonomous on-road driving: state-of-the-art and future research directions. Transport Res Part C: Emerging Technol. 2015;60:416–442.
[32] Betts JT. Survey of numerical methods for trajectory optimization. J Guid Control Dyn. 1998;21(2):193–207.
[33] Polak E. An historical survey of computational methods in optimal control. SIAM Rev. 1973;15(2):553–584.
[34] Perantoni G, Limebeer DJ. Optimal control for a formula one car with variable parameters. Vehicle Syst Dyn. 2014;52(5):653–678.
[35] Dal Bianco N, Lot R, Gadola M. Minimum time optimal control simulation of a gp2 race car. Proc Inst Mech Eng Part D: J Automob Eng. 2018;232(9):1180–1195.
[36] Dal Bianco N, Bertolazzi E, Biral F, et al. Comparison of direct and indirect methods for minimum lap time optimal control problems. Vehicle Syst Dyn. 2018;57(5):1–32.
[37] Dijkstra EW. A note on two problems in connexion with graphs. Numer Math. Dec 1959;1(1):269–271.
[38] Hart PE, Nilsson NJ, Raphael B. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans Syst Sci Cybernet. 1968 July;4(2):100–107.
[39] Pohl I. First results on the effect of error in heuristic search. Edinburgh University, Department of machine intelligence and perception; 1969. MIP-R.
[40] Stentz A. Optimal and efficient path planning for partially-known environments. In: Proceedings of the 1994 IEEE International Conference on Robotics and Automation. Vol. 4, 1994 May. p. 3310–3317.
[41] Stentz A. The focussed D* algorithm for real-time replanning. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2. Morgan Kaufmann Publishers Inc.; 1995. p. 1652–1659. IJCAI'95.
[42] Koenig S, Likhachev M. Fast replanning for navigation in unknown terrain. IEEE Trans Robot. 2005 June;21(3):354–363.
[43] Lavalle SM. Rapidly-Exploring random trees: a new tool for path planning. Ames: Iowa State University; 1998.
[44] Karaman S, Frazzoli E. Optimal kinodynamic motion planning using incremental sampling-based methods. In: 49th IEEE Conference on Decision and Control (CDC). 2010 Dec. p. 7681–7687.
[45] Otte M, Frazzoli E. RRT-X: real-time motion planning/replanning for environments with unpredictable obstacles. Springer International Publishing; 2015. Chapter 27; p. 461–478.
[46] Hsu D, Kindel R, Latombe JC, et al. Randomized kinodynamic motion planning with moving obstacles. Int J Rob Res. 2002;21(3):233–255.
[47] Katriniok A, Abel D. LTV-MPC approach for lateral vehicle guidance by front steering at the limits of vehicle dynamics. In: 2011 50th IEEE Conference on Decision and Control and European Control Conference. 2011 Dec. p. 6828–6833.

[48] Gerdts M, Karrenberg S, Müller-Beßler B, et al. Generating locally optimal trajectories for an automatically driven car. Optim Eng. 2008 Apr;10(4):439.
[49] Glaser S, Vanholme B, Mammar S, et al. Maneuver-based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction. IEEE Trans Intell Transp Syst. 2010 Sept;11(3):589–606.
[50] Jeon Jh, Cowlagi RV, Peters SC, et al. Optimal motion planning with the half-car dynamical model for autonomous high-speed driving. In: 2013 American Control Conference. 2013 June. p. 188–193.
[51] Arab A, Yu K, Yi J, et al. Motion planning for aggressive autonomous vehicle maneuvers. In: 2016 IEEE International Conference on Automation Science and Engineering (CASE). 2016 Aug. p. 221–226.
[52] Rizano T, Fontanelli D, Palopoli L, et al. Global path planning for competitive robotic cars. In: 52nd IEEE Conference on Decision and Control. 2013 Dec. p. 4510–4516.
[53] Rizano T, Fontanelli D, Palopoli L, et al. Local motion planning for robotic race cars. 52nd IEEE Conference on Decision and Control. 2013. p. 4510–4516.
[54] Funke J, Theodosis P, Hindiyeh R, et al. Up to the limits: autonomous audi TTS. In: IEEE Intelligent Vehicles Symposium. 2012 June. p. 541–547.
[55] Guldner J, Tan HS, Patwardhan S. Study of design directions for lateral vehicle control. In: Proceedings of the 36th IEEE Conference on Decision and Control; Vol. 5, 1997. p. 4732–4737.
[56] Roselli F, Corno M, Savaresi SM, et al. H-Infinity control with look-ahead for lane keeping in autonomous vehicles. 2017 IEEE Conference on Control Technology and Applications (CCTA). 2017. p. 2220–2225.
[57] Werling M, Gröll L, Bretthauer G. Invariant trajectory tracking with a full-size autonomous road vehicle. IEEE Trans Robot. 2010;26(4):758–765.
[58] Werling M. Ein neues Konzept für die Trajektoriengenerierung und -stabilisierung in zeitkritischen Verkehrsszenarien. At-Automatisierungstechnik. 2012;60(1):53–54.
[59] Fuchshumer S, Schlacher K, Rittenschober T. Nonlinear vehicle dynamics control - a flatness based approach. In: Proceedings of the 44th IEEE Conference on Decision and Control. 2005 Dec. p. 6492–6497.
[60] Aguilar LE, Hamel T, Soueres P. Robust path following control for wheeled robots via sliding mode techniques. In: Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems. Innovative Robotics for Real-World Applications. IROS '97; Vol. 3, 1997 Sept. p. 1389–1395.
[61] Hamerlain F, Achour K, Floquet T, Trajectory tracking of a car-like robot using second order sliding mode control. In: 2007 European Control Conference (ECC). 2007 July. p. 4932–4936.
[62] Katriniok A, Maschuw JP, Eckstein L, et al. Optimal vehicle dynamics control for combined longitudinal and lateral autonomous vehicle guidance. In: 2013 European Control Conference (ECC). 2013. p. 974–979.
[63] Calzolari D, Schürmann B, Althoff M. Comparison of trajectory tracking controllers for autonomous vehicles. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). 2017 Oct. p. 1–8.
[64] Roborace [Internet]. [cited 2018 June 01]; 2018. Available from: http://roborace.com.
[65] Kong TY, Rosenfeld A. Topological algorithms for digital image processing. Amsterdam: Elsevier Science Inc.; 1996.
[66] Maurer CR, Qi R, Raghavan V. A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. IEEE Trans Pattern Anal Mach Intell. 2003 Feb;25(2):265–270.
[67] Meyer F. Topographic distance and watershed lines. Signal Processing. 1994;38(1):113–125. Mathematical Morphology and its Applications to Signal Processing.
[68] Orfanidis S. Introduction to signal processing. Upper Saddle River: Pearson Education, Inc.; 1996.
[69] Gundlach I, Konigorski U, Hoedt J. Zeitoptimale Trajektorienplanung für automatisiertes Fahren im fahrdynamischen Grenzbereich. VDI Ber. 2017;2292:223–234.

[70] Papula L. Mathematische formelsammlung: für ingenieure und naturwissenschaftler. Braunschweig/Wiesbaden: Springer Fachmedien Wiesbaden; 2017.
[71] Chen W, Yang J, Guo L, et al. Disturbance-Observer-Based control and related methods—an overview. IEEE Trans Ind Electron. 2016 Feb;63(2):1083–1095.
[72] Boyd S, Doyle J. Comparison of peak and RMS gains for discrete-time systems. Syst Control Lett. 1987;9(1):1–6.