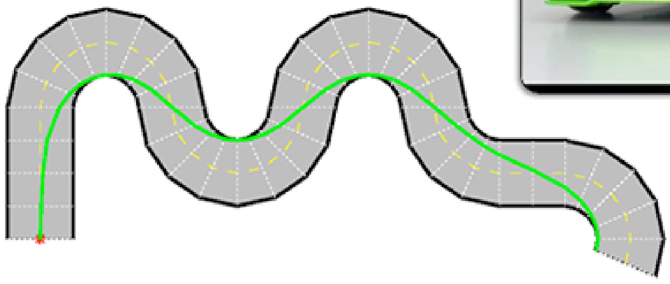
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Kevin Meier

Sub-optimal path planning for 1:43 scale race cars



Semester project FS10

Automatic Control Laboratory (Institut für Automatik, IfA)
Swiss Federal Institute of Technology (ETH) Zurich

Supervisors

Dr. Colin Jones
Melanie Zeilinger
Prof. Dr. Manfred Morari

June 2010

Abstract

A race driver model determines mathematically the shortest lap time for a known track. Human competitors as well as different race cars could therefore be rated according to this lower bound on lap time.
A two step approach solves first the optimal trajectory and as second step determines the speed profile for the optimal trajectory. Race car controllers can use the optimal trajectory as well as the speed profile as reference values. Data are updated at high sampling rates which requires the algorithm to run online and the optimization to be solved fast and efficient.
Such a virtual race driver environment is developed, improved and coded in MATLAB and C++.

Acknowlegement

Optimization is a greatfull tool for a lot of problems. The "Institut für Automatik" (IfA) offered interesting semester projects in this area. I am glad I took the opportunity to get insights into optimization. It was a wonderful time and I gained a lot of experience. I would like to thank my supervisors Dr. Colin Jones, Melanie Zeilinger and the IfA. They all made this possible and supported me the entire time!

June 2010,
Kevin Meier

Contents

1 Introduction

1.1 Motivation

A race driver model determines mathematically the best performance for a known track, which is obviously the shortest lap time. Human competitors as well as different race cars could therefore be rated according to this lower bound on lap time. Moreover cost benefits at a design state of such racing cars could be achieved. Expensive activities like aerodynamic devices and other tuning could be carried out in a virtual environment. This holds not only for the design stage but also for the identification of the optimal vehicle set-up to obtain the best lap time [FB07].

1.2 Goal and content

In a previous semester project at the IfA about convex optimization and time optimum path planning [Col09], as well as in [FB07], a race driver model which includes the geometry of the track and the car dynamics is introduced.
The optimization is split into two parts. The first step is represented by finding the optimal trajectory which is the best compromise between the shortest track and the track that allows to achieve the highest speeds (least curvature track). The second step is then to optimize the speed profile for the determined optimal trajectory.
For both steps assumptions and simplifications are made to derive simple and fast solvable optimization structures like convex linear/quadratic programs.

With each sampling one gets new feedback in form of position, speed and orientation. The higher the sampling rate, the sooner the controller can counteract deviations from the desired value and disturbances. Since fast sampling in the range of 20-50 Hz is desired, such simplifications to achieve fast solvable problems are beneficial. One goal is to justify these assumptions and look for alternative sparse problem structure. How suboptimal are they and what effect on the lap time do they have? Which structure can be solved faster?
An other goal and part is to develop and implement a continuously repeating optimization algorithm in C++ and MATLAB.

In section 2 a closer look is taken at the above mentioned assumptions, simplifications and problem structures.
Section 3 then deals with the implementation in C++ and MATLAB.
At the end a summary is followed by an outlook to some further research topics.

2 Justification of the simplifications

Table 2.1 shows an overview of the approximations and their impact, which will be investigated and discussed in this chapter.

- The shortest track method is a Second Order Cone Program (SOCP) which is converted into a Quadratic Program (QP) using squared distances rather than only distances.
- The arc-length parameterization simplifies the curvature calculation but is only valid exactly on the trajectory. Repeating the optimization and updating the parameterization lets the solution converge to the least curvature trajectory and therefore does not introduce loss of optimality.
- The speed profile is a Semi-definite Program (SDP) and is converted into a Linear Program (LP) using squared speed samples. There is no loss of optimality.

Method	Problem structure	Approximation	Simplified structure	Loss of optimality
Shortest track	SOCP	Squared distance	QP	$<$ 3.5% slower
Least curvature	QP	Arc-length parameterization Squared curvature	QP	Iteration, no loss
Speed profile	SDP	Squared speed	LP	No loss

Table 2.1: Overview and impact of the approximations.

Before any optimization problem can be derived, a suitable mathematical model and parameterization of a track and possible trajectories are required. The first part of this sections therefore deals with the modeling and parameterization of a track and and how a possible trajectory is mathematically described.

The entire algorithm is divided into finding the optimal trajectory and then solving the speed profile for this trajectory.
The optimal trajectory depends on the car dynamics and is a combination of shortest track and least curvature. Three of the following subsections investigate the approximations and possible extensions to this optimal trajectory problem. The last subsection deals with the speed profile.

2.1 Track Model and Parameterization

One common way to model a track is described in [Col09], [FB07] and [QD].
The track consists of inner (r_{inner}) and outer (r_{outer}) boarders which are described in Carthesian coordinates ($r = (x, y)^T \in \mathbb{R}^2$). Any trajectory inside these boarders can then be represented with a relative distance ($\alpha \cdot \Delta$) from the inner boarder, whereas $\Delta = r_{outer} - r_{inner}$ describes the differences between the boarder lines.
Equation (2.1) summaries the math and Figure 2.1 illustrates the continuous case.
Discretizing the track into several segments, used for numerical calculations, is obtained by sampling θ at N sampling points $\theta_i, i \in \{0, 1, \ldots, N-1\}$ and shown in Figure 2.2.
The track is arc-length parameterized, so that uniform sampling of the parameterization variable θ leads to uniform sampling in space.

$$\begin{aligned} r_{inner}(\theta) &= (x_{inner}(\theta), y_{inner}(\theta))^T \\ r_{outer}(\theta) &= (x_{outer}(\theta), y_{outer}(\theta))^T \\ \Delta(\theta) &= r_{outer}(\theta) - r_{inner}(\theta) \\ r(\theta) &= r_{inner}(\theta) + \alpha(\theta) \cdot \Delta(\theta) \end{aligned} \tag{2.1}$$

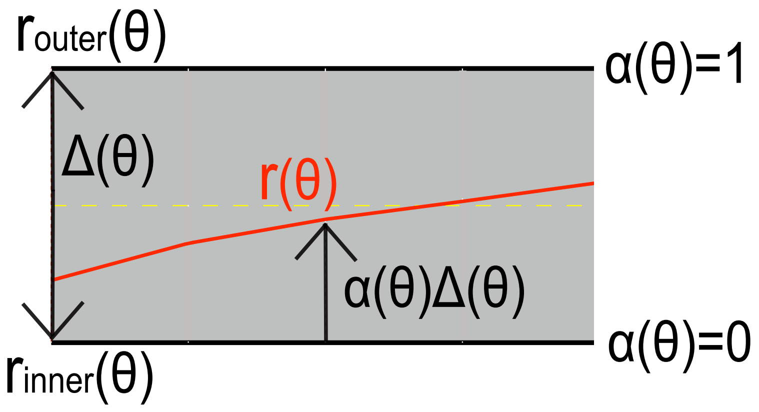


Figure 2.1: Continous-space track and trajectory model.

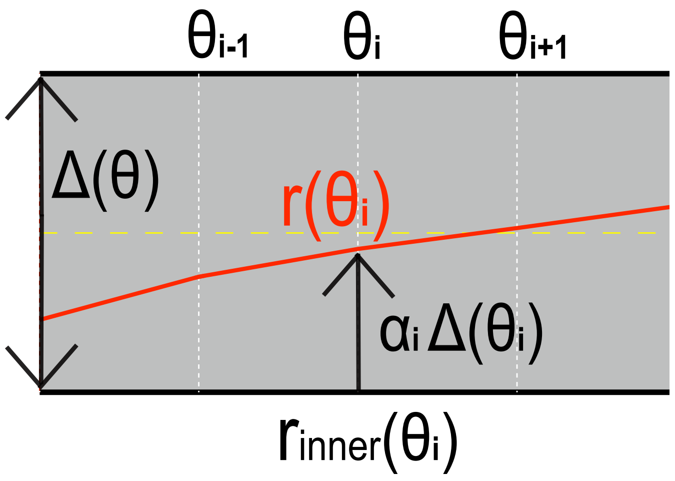


Figure 2.2: Discretized-space track and trajectory model.

2.2 Shortest Track

Since time is measured as distance divided by speed (time $= \frac{\text{distance}}{\text{speed}}$), minimizing the total distance seems to be an intuitive way of minimizing the total travel time.

2.2.1 Problem Formulation

In [Col09] the continuous (equation 2.2) and the discrete (equation 2.3) problem formulations are derived. The discrete form shown in equation (2.3) is convex and a second order cone program (SOCP). Summing over the *squared* euclidean distance and therefore omitting the square root on the right hand side reduces the computational complexity and simplifies the problem into a quadratic program (QP) (equation 2.4).

$$\begin{aligned} \text{min.} \quad & \int_0^L dl = \int_0^1 \sqrt{x'(\theta)^2 + y'(\theta)^2} d\theta \qquad \theta \in [0,1] \\ \text{s.t.} \quad & 0 \leq \alpha(\theta) \leq 1 \\ & r(\theta) = r_{inner}(\theta) + \alpha(\theta)\Delta(\theta) \end{aligned} \tag{2.2}$$

$$\begin{aligned} \text{min.} \quad & \sum_{i=0}^{N-2} li = \sum_{i=0}^{N-2} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq 1 \quad \forall\, i \in \{0, 1, \dots, N-1\} \\ & r_i = r_{inner,i} + \alpha_i \Delta_i \end{aligned} \tag{2.3}$$

$$\begin{aligned} \text{min.} \quad & \sum_{i=0}^{N-2} li^2 = \sum_{i=0}^{N-2} (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq 1 \quad \forall i \in \{0, 1, \dots, N-1\} \\ & r_i = r_{inner,i} + \alpha_i \Delta_i \end{aligned} \tag{2.4}$$

Figure 2.3 and 2.4 show the shortest track trajectory for two different tracks, each with track width $\Delta = 1$. The green trajectory describes the original SOCP problem and red the simplified QP. They qualitatively coincide almost everywhere.
Table 2.2 quantifies the difference in terms of total lap time. The calculated trajectory is given to the speed profile and with the optimized speed samples the lap time can be calculated. It is assumed that the speed changes linearly between two speed samples. The time to travel from one sample to an other is then the averaged speed divided by the distance given from the optimization problem.

$$T = \sum_{i=0}^{N-1} \frac{\frac{1}{2} \cdot (v_{i+1} + v_i)}{\text{distance}} \tag{2.5}$$

Wider tracks have higher deviations, but even with width Δ of 1.5 the lap time of the QP approximation was at most 3.5% slower. Therefore, the simplification is valid and close to the optimum.

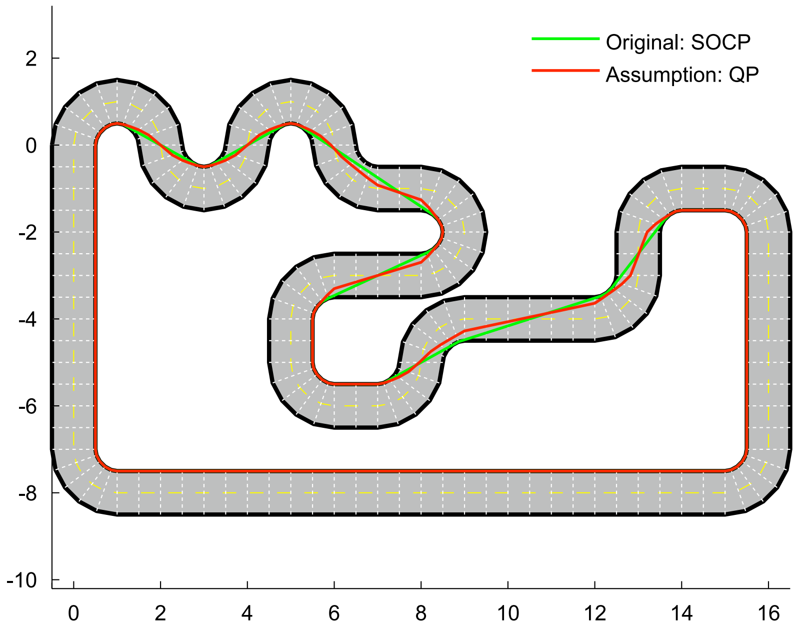


Figure 2.3: Track 1 with track width $\Delta = 1$.

Track 1	SOCP Lap time	QP Lap time
Width $\Delta = 0.5$	58.2283 s	+0.1033%
Width $\Delta = 1$	53.8070 s	+0.7512%
Width $\Delta = 1.5$	50.1494 s	+2.6001%
Track 2	**SOCP Lap time**	**QP Lap time**
Width $\Delta = 0.5$	44.7654 s	+0.0888%
Width $\Delta = 1$	40.4514 s	+0.8809%
Width $\Delta = 1.5$	36.6189 s	+3.4196%

Table 2.2: SOCP versus QP for different track widths (Δ).

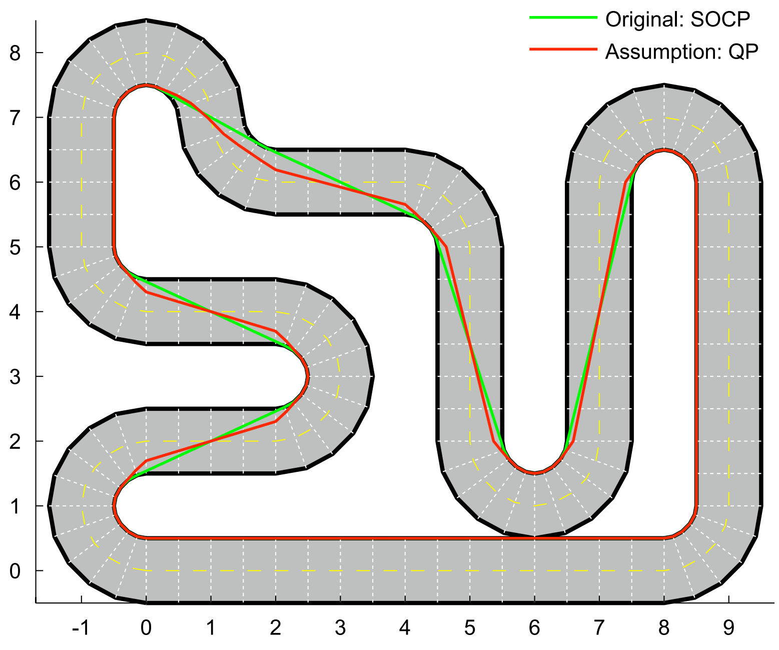


Figure 2.4: Track 2 with track width $\Delta = 1$.

2.2.2 Problem Structure

There are two ways of formulating the discrete quadratic program. The equality constraint of equation (2.4) can be separated in x and y coordinates and either inserted into the cost function (done in [Col09],[FB07]) or left as equality constraints.

Shortest Track - depended on α

The equality constraint of (2.4) is written in x and y separately.

$$\begin{aligned} r_i = r_{inner,i} + \alpha_i \Delta_i \quad &\rightarrow \quad x_i = x_{inner,i} + \alpha_i \Delta x_i \\ &\rightarrow \quad y_i = y_{inner,i} + \alpha_i \Delta y_i \end{aligned} \tag{2.6}$$

Inserting these separated equalities and writing the difference with aid of the auxiliary variable ΔP leads to equation (2.7).

$$\begin{aligned}
\Delta P_{x,i} = x_{i+1} - x_i &= \underbrace{x_{inner,i+1} - x_{inner,i}}_{\Delta x_{i,0}} + \alpha_{i+1}\Delta x_{i+1} - \alpha_i \Delta x_i \qquad (2.7)\\
&= \Delta x_{i,0} + [\Delta x_{i+1}, -\Delta x_i] \cdot \underbrace{\begin{bmatrix}\alpha_{i+1}\\ \alpha_i\end{bmatrix}}_{\bar{\alpha}_i}\\
&= \Delta x_{i,0} + [\Delta x_{i+1}, -\Delta x_i] \cdot \underbrace{\begin{bmatrix}0 & \cdots & 0 & 1_i & 0 & 0 & \cdots & 0\\ 0 & \cdots & 0 & 0 & 1_{i+1} & 0 & \cdots & 0\end{bmatrix}}_{E_i} \cdot \underbrace{\begin{bmatrix}\alpha_0\\ \vdots \\ \alpha_{N-1}\end{bmatrix}}_{\bar{\alpha}}
\end{aligned}$$

$\Delta P_{y,i}$ is derived analog. The cost function of equation (2.4) can then be formulated like equation (2.8).

$$\begin{aligned}
L &= \sum_{i=0}^{N-2} \Delta P_{x,i}^T \cdot \Delta P_{x,i} + \Delta P_{y,i}^T \cdot \Delta P_{y,i} \qquad (2.8)\\
&= \sum_{i=0}^{N-2} (\Delta x_{i,0} + [\Delta x_{i+1}, -\Delta x_i] \cdot E_i \cdot \bar{\alpha}) \cdot (\Delta x_{i,0} + [\Delta x_{i+1}, -\Delta x_i] \cdot E_i \cdot \bar{\alpha})^T + \Delta P_{y,i}^T \cdot \Delta P_{y,i}\\
&= \sum_{i=0}^{N-2} \bar{\alpha}_i^T \cdot H_i \cdot \bar{\alpha}_i + \sum_{i=0}^{N-2} B_i \cdot \bar{\alpha}_i + const.\\
&= \bar{\alpha}^T \cdot \underbrace{\sum_{i=0}^{N-2} E_i^T \cdot H_i \cdot E_i}_{H_{st}} \cdot \bar{\alpha} + \underbrace{\sum_{i=0}^{N-2} B_i \cdot E_i}_{B_{st}} \cdot \bar{\alpha} + const.
\end{aligned}$$

with $$H_i = \begin{bmatrix} \Delta x_{i+1}^2 + \Delta y_{i+1}^2 & -\Delta x_{i+1}\Delta x_i - \Delta y_{i+1}\Delta y_i \\ -\Delta x_{i+1}\Delta x_i - \Delta y_{i+1}\Delta y_i & \Delta x_i^2 + \Delta y_i^2 \end{bmatrix}$$

$$B_i = 2 \cdot \begin{bmatrix} \Delta x_{i+1}\Delta x_{i,0} + \Delta y_{i+1}\Delta y_{i,0} & -\Delta x_i \Delta x_{i,0} - \Delta y_i \Delta y_{i,0} \end{bmatrix}$$

Minimizing the length L yields to

$$\begin{aligned}
\text{min.} \quad & \bar{\alpha}^T \cdot H_{st} \cdot \bar{\alpha} + B_{st} \cdot \bar{\alpha} + const.\\
\text{s.t.} \quad & 0 \leq \alpha_i \leq 1 \quad \forall i \in \{0, 1, \ldots, N-1\} \qquad (2.9)
\end{aligned}$$

The optimizer variable α has dimension $N \times 1$. H_{st} is 2×2 block diagonal and sparse with dimension N×N.

Shortest Track - depended on α, x and y

The optimizer variable is extended with the variables x and y and has dimension 3·N × 1. The equality constraints remain unchanged and are taken into the problem formulation.

$$z = \begin{bmatrix} \alpha \\ x \\ y \end{bmatrix} \tag{2.10}$$

The H_{st} is build according to the evaluated cost function of equation (2.4).

$$\begin{aligned} (x_{i+1} - x_i)^2 &= x_{i+1}^2 - 2x_{i+1}x_i + x_i^2 \\ \sum_{i=0}^{N-2} (x_{i+1} - x_i)^2 &= x^T \cdot \tilde{H} \cdot x \end{aligned} \tag{2.11}$$

$$\text{with} \qquad \tilde{H} = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ -1 & 2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 2 & -1 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix}$$

The $\tilde{H}$ is formed analog for y and is obviously the same as for x. Minimizing again yields to the following problem formulation.

$$\begin{aligned} \text{min.} \quad & z^T \cdot H_{st} \cdot z \\ \text{s.t.} \quad & 0 \le \alpha_i \le 1 \quad \forall i \in \{0, 1, \dots, N-1\} \\ & x_i = x_{inner,i} + \alpha \cdot (x_{outer,i} - x_{inner,i}) \\ & y_i = y_{inner,i} + \alpha \cdot (y_{outer,i} - y_{inner,i}) \\ \text{with} \quad & H_{st} = \begin{bmatrix} 0_{N\times N} & 0_{N\times N} & 0_{N\times N} \\ 0_{N\times N} & \tilde{H} & 0_{N\times N} \\ 0_{N\times N} & 0_{N\times N} & \tilde{H} \end{bmatrix} \end{aligned} \tag{2.12}$$

The optimizer variable z has dimension $3 \cdot N \times 1$. H_{st} is again block diagonal and sparse but with dimension $3 \cdot N \times 3 \cdot N$.

The second method, using all three variables α, x and y and having therefore a factor of three times bigger dimension could only perform better if the appropriate QP-solver takes into account the sparsity of the H_{st} matrix. Since the H_{st} matrix of the first method depending only on α is already sparse, transforming the problem to higher dimension does not bring any benefit, it is even worse since the dimension has increased by a factor of three.
To illustrated this, both problems were solved with the MATLAB-CPLEX-solver and the MPT-Toolbox. Table 2.3 shows the differences. The second method is indeed slower.

Method	CPLEX-solver time for a horizon of 30 steps
Depended on α	0.72 ms
Depended on α, x, y	2.29 ms

Table 2.3: Shortest track: CPLEX-solver time to solve the two different methods for a horizon of 30 steps.

2.3 Least Curvature Track

Any car driving to fast in a curve will eventually loose grip and begin to slide. The car begins to slip if its lateral acceleration exceeds a certain limit. For simplicity the maximal lateral acceleration is assumed to be known and constant. According to [Col09] the lateral acceleration and the speed of the car have the following relationship.

$$\begin{aligned} a_{lat} &= \kappa \cdot s^2 = \frac{s^2}{R} \\ & \kappa : \text{curvature} \\ & s : \text{speed} \\ & R : \text{curvature radius} \end{aligned} \tag{2.13}$$

Equation (2.13) shows if the the lateral acceleration is constrained to avoid slippage, higher speed s can be achieved if the curvature κ is as low as possible, hence minimized.

2.3.1 Problem Formulation

In [Col09] the continuous (equation 2.15) and the discrete (equation 2.16) problem formulations are derived. Minimizing the *squared* curvature convert the discrete problem into a quadratic program (equation 2.17). This is a reasonable way of minimizing curvature since it provides a smooth curvature along the entire track.
[FB07] suggests to use *natural cubic spline* to interpolate the trajectory between the sampled data points. Cubic splines are smooth and continuous in first and second derivatives and therefore continuous in curvature.
Arc-length parameterization is assumed and the curvature is therefore equal to the second derivative of the trajectory (equation 2.14).

$$\text{Curvature} \quad \kappa(\theta) = ||r''(\theta)|| \tag{2.14}$$

$$\begin{aligned} \text{min.} \quad & \int_0^1 \kappa(\theta) d\theta \qquad \theta \in [0,1] \\ \text{s.t.} \quad & 0 \leq \alpha(\theta) \leq 1 \\ & r(\theta) = r_{inner}(\theta) + \alpha(\theta)\Delta(\theta) \end{aligned} \tag{2.15}$$

$$\begin{aligned} \text{min.} \quad & \sum_{i=0}^{N-2} \kappa(\theta_i) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq 1 \quad \forall i \in \{0,1,\ldots,N-1\} \\ & r_i = r_{inner,i} + \alpha_i \Delta_i \end{aligned} \tag{2.16}$$

$$\begin{aligned} \text{min.} \quad & \sum_{i=0}^{N-2} \kappa(\theta_i)^2 = \sum_{i=0}^{N-2} ||r''(\theta_i)||^2 = \sum_{i=0}^{N-2} [x''(\theta_i)]^2 + [y''(\theta_i)]^2 \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq 1 \quad \forall i \in \{0, 1, \ldots, N-1\} \\ & r_i = r_{inner,i} + \alpha_i \Delta_i \end{aligned} \tag{2.17}$$

Spline Interpolation

Equation (2.17) uses the second derivatives. As mentioned above, they are calculated with natural cubic splines, as in [Col09]. The spline problem can be solve according to [Wik10].
With the following notations, the spline problem can be formulated as in equation (2.18).

- Second order derivatives in x and y: $z_{x,i} := x''(\theta_i)$ and $z_{y,i} := y''(\theta_i)$
- Distance between two sample points: $h_i := \theta_{i+1} - \theta_i = \Delta\theta$
  Where θ is arc-length parametrized as mentioned in chapter 2.1.
- $b_{x,i} := \frac{1}{h_i}(x_{i+1} - x_i)$ and $b_{y,i} := \frac{1}{h_i}(y_{i+1} - y_i)$

$$\tilde{H}_s \tilde{z}_x = \tilde{b}_x \qquad \tilde{H}_s \tilde{z}_y = \tilde{b}_y \tag{2.18}$$

whereas $\tilde{H}_s, \tilde{z}_x, \tilde{z}_y, \tilde{b}_x$ and $\tilde{b}_y$ are defined in equation (2.19).

$$\begin{aligned} \tilde{H}_s &:= \begin{bmatrix} 2(h_0+h_1) & h_1 & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & h_2 & 2(h_2+h_3) & h_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{N-4} & 2(h_{N-4}+h_{N-3}) & h_{N-3} \\ & & & & h_{N-3} & 2(h_{N-3}+h_{N-2}) \end{bmatrix} \\ \tilde{z}_x &:= \begin{bmatrix} z_{x,1} \\ \vdots \\ z_{x,N-2} \end{bmatrix} \qquad \tilde{z}_y := \begin{bmatrix} z_{y,1} \\ \vdots \\ z_{y,N-2} \end{bmatrix} \\ \tilde{b}_x &:= 6 \cdot \begin{bmatrix} b_{x,1} - b_{x,0} \\ \vdots \\ b_{x,N-2} - b_{x,N-3} \end{bmatrix} \qquad \tilde{b}_y := 6 \cdot \begin{bmatrix} b_{y,1} - b_{y,0} \\ \vdots \\ b_{y,N-2} - b_{y,N-3} \end{bmatrix} \end{aligned} \tag{2.19}$$

The linear system of equations in this form is under-determined. Using natural cubic spline provides two additional conditions. The first and last second derivative has to equal be equal to zero. The system converts then into a determined linear system of equations.

$$z_{x,i} = z_{y,i} = 0, \quad i = \{0, N-1\} \tag{2.20}$$

The $\tilde{H}_s$ is then extended to H_s which has dimension N $\times$ N.

$$H_s \quad := \quad \begin{bmatrix} 1 & 0_{1\times N-2} & 0 \\ h_0 & & 0_{N-3\times 1} \\ 0_{N-3\times 1} & \tilde{H}_s & h_{N-2} \\ 0 & 0_{1\times N-2} & 1 \end{bmatrix} \qquad z_x := \begin{bmatrix} z_{x,0} \\ \tilde{z}_x \\ z_{x,N-1} \end{bmatrix} \qquad b_x := \begin{bmatrix} 0 \\ \tilde{b}_x \\ 0 \end{bmatrix} \tag{2.21}$$

H_s is strictly diagonal dominant and thus invertible.

$$z_x \quad = \quad H_s^{-1} \cdot b_x \tag{2.22}$$

Since b_x depends on h_i and x, it can be written as

$$b_x \quad = \quad B_s \cdot x \tag{2.23}$$

$$\text{where } B_s \quad := \quad 6 \cdot \begin{bmatrix} 0 & 0 & 0 & & & & \\ \frac{1}{h_0} & -(\frac{1}{h_0} + \frac{1}{h_1}) & \frac{1}{h_1} & & & & \\ & \frac{1}{h_1} & -(\frac{1}{h_1} + \frac{1}{h_2}) & \frac{1}{h_2} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \frac{1}{h_{N-3}} & -(\frac{1}{h_{N-3}} + \frac{1}{h_{N-2}}) & \frac{1}{h_{N-2}} \\ & & & 0 & 0 & 0 \end{bmatrix}$$

This finally turns the entire spline problem into a linear system of equations (2.24) where $H_s^{-1} \cdot B_s$ is constant since the parameterization is constant and not part of the optimization. The problem for this approach is the constant parameterization. [Col09] suggests an iterative procedure. After each optimization, the new trajectory is calculated and the parameterization updated. The optimization is then run again with the updated parameterization and the new trajectory converges finally to the least curvature trajectory, which is illustrated in Figure 2.5.

$$\begin{aligned} z_x &= H_s^{-1} \cdot B_s \cdot x \\ z_y &= H_s^{-1} \cdot B_s \cdot y \end{aligned} \tag{2.24}$$

2.3.2 Problem Structure

There are again the same two ways of formulating the discrete quadratic program for the least curvature method as for the shortest track method. The equality constraint of equation (2.17) can be separated in x and y coordinates and either inserted into the cost function (done in [Col09],[FB07]) or left as equality constraints.

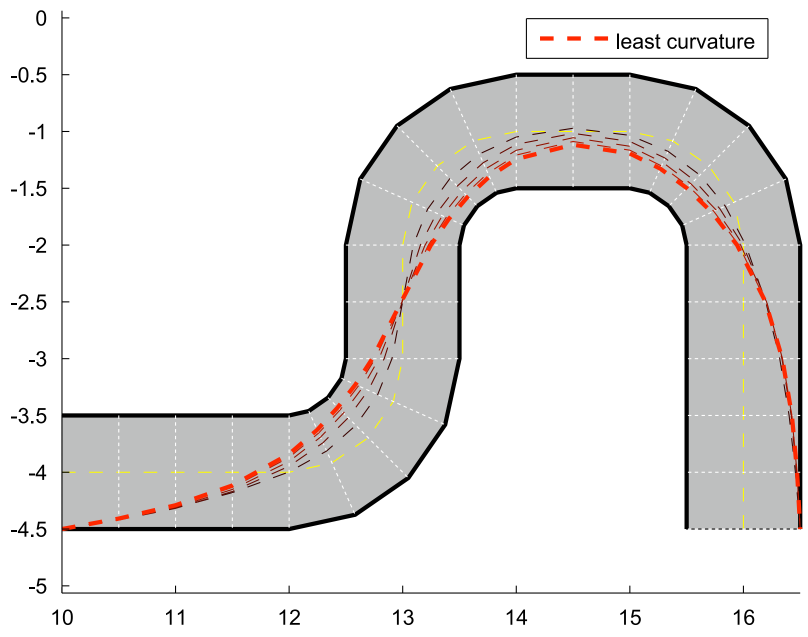


Figure 2.5: 5 least curvature iterations.

Least Curvature - depended on α

Since $z_x = x''(\theta)$ and equation (2.24) holds, the cost function of equation (2.17) can be written as equation (2.25).

$$\begin{aligned} & \text{min.} && z_x^T \cdot z_x + z_y^T \cdot z_y \\ \Rightarrow & \text{min.} && x^T (H_s^{-1} \cdot B_s)^T (H_s^{-1} \cdot B_s) x + y^T (H_s^{-1} \cdot B_s)^T (H_s^{-1} \cdot B_s) y \end{aligned} \tag{2.25}$$

Inserting the equality constraints $x_i = x_{inner,i} + \alpha_i \Delta x_i$ and defining $\Delta x_i = x_{outer,i} - x_{inner,i}$ yields to the following optimization problem.

$$\begin{aligned} \text{min.} \quad & \alpha^T \cdot H_{lc} \cdot \alpha + B_{lc} \cdot \alpha + const. \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq 1 \quad \forall i \in \{0, 1, \dots, N-1\} \end{aligned} \tag{2.26}$$

where

$$\begin{aligned} D &= H_s^{-1} \cdot B_s \\ H_{lc} &= \text{diag}(\Delta x)^T \cdot D^T \cdot D \cdot \text{diag}(\Delta x) + \text{diag}(\Delta y)^T \cdot D^T \cdot D \cdot \text{diag}(\Delta y) \\ B_{lc} &= 2 * (x_{inner}^T \cdot D^T \cdot D \cdot \text{diag}(\Delta x) + y_{inner}^T \cdot D^T \cdot D \cdot \text{diag}(\Delta y)) \\ \text{const.} &= x_{inner}^T \cdot D^T \cdot D \cdot x_{inner} + y_{inner}^T \cdot D^T \cdot D \cdot y_{inner} \end{aligned}$$

The optimizer variable α has dimension $N \times 1$. H_{lc} is a dense matrix with dimension N×N.

During tests with MATLAB, an other problem arose. The matrix $D = H_s^{-1} \cdot B_s$ was ill conditioned. Normalizing the distance vector h with its smallest value h_{min} worked out to avoid ill conditioned matrices. One has to make sure that h_{min} will never be zero, but since the discretized track does not contain any point more than once, h_{min} will always be greater than zero.
Table 2.4 shows the condition number and maximum absolute eigenvalue for different matrices before and after the scaling for track 1 (see Figure 2.3) and a horizon of 100 samples.

Matrix	Before scaling	After scaling
Max. element of H_s	$5.1 \cdot 10^{10}$	72
Condition number of H_s	81	9.7
Max. abs. eigenvalue of D	$3.1 \cdot 10^{5}$	11.9
Max. abs. eigenvalue of $D^T D$	$9.9 \cdot 10^{10}$	142

Table 2.4: Condition number and maximum absolute eigenvalue of different matrices before and after scaling for track 1 and a horizon of 100 samples.

Least Curvature - depended on α, z_x and z_y

The optimizer variable is extended with the second order derivatives z_x and z_y and has dimension 3·N × 1.

$$\tilde{x} = \begin{bmatrix} \alpha \\ z_x \\ z_y \end{bmatrix} \tag{2.27}$$

z_x and z_y remain as equality constraints.

$$\begin{aligned} z_x &= H_s^{-1} \cdot B_s \cdot (x_{inner,i} + \alpha_i \Delta x_i) \\ z_x - (H_s^{-1} \cdot B_s)\, \alpha_i \Delta x_i &= H_s^{-1} \cdot B_s \cdot x_{inner,i} \end{aligned} \tag{2.28}$$

The optimization problem can then be written as equation (2.29).

$$\begin{aligned} \text{min.} \quad & \tilde{x}^T \cdot H_{lc} \cdot \tilde{x} \\ \text{s.t.} \quad & 0 \le \alpha_i \le 1 \quad \forall i \in \{0, 1, \dots, N-1\} \\ & z_x - (H_s^{-1} \cdot B_s)\, \alpha_i \Delta x_i = H_s^{-1} \cdot B_s \cdot x_{inner,i} \\ & z_y - (H_s^{-1} \cdot B_s)\, \alpha_i \Delta y_i = H_s^{-1} \cdot B_s \cdot y_{inner,i} \\ \text{with} \quad & H_{lc} = \begin{bmatrix} 0_{N\times N} & 0_{N\times N} & 0_{N\times N} \\ 0_{N\times N} & \mathrm{I}_{N\times N} & 0_{N\times N} \\ 0_{N\times N} & 0_{N\times N} & \mathrm{I}_{N\times N} \end{bmatrix} \end{aligned} \tag{2.29}$$

Although the matrices of the second method were sparse, the MATLAB-CPLEX-solver of the MPT-Toolbox was much slower in this case. Table 2.5 shows the differences in time consumption for both methods.

Method	CPLEX-solver time for a horizon of 30 steps, including 5 intern iterations.
Depended on α	6.1 ms
Depended on α, z_x, z_y	25.4 ms

Table 2.5: Least curvature: CPLEX-solver time to solve the two different methods for a horizon of 30 steps and 5 intern iterations.

2.4 Combination of Shortest and Least Curvature Track

Shortest track and least curvature are both quadratic programs. Instead of solving them separately, the matrices can be combined and solved as one single QP, as done in [FB07].

L	:	Linear Combination
LC	:	Least Curvature
ST	:	Shortest Track

$$\begin{aligned} L &= (1-\varepsilon)\cdot LC + \varepsilon \cdot ST \qquad (2.30)\\ &= \alpha^T H_L \alpha + B_L \alpha^T \\ \text{where} & \\ H_L &= (1-\varepsilon)\cdot H_{LC} + \varepsilon \cdot H_{ST} \\ B_L &= (1-\varepsilon)\cdot B_{LC} + \varepsilon \cdot B_{ST} \end{aligned}$$

This leads to a linear change from the least curvature to the shortest track solution. For different tracks, the achieved lap time for the shortest track method was about 8-10% slower.
An alternative way of scaling is to have a higher and finer resolution around the (faster) least curvature method ($\varepsilon \to 0$). This is done by introducing different weights. The weight for the shortest track w_{ST} is the worst case shortest track, namely the length of the outer boarder line. The weight for the least curvature w_{LC} is the value of the cost function for the trajectory described by the centerline, which is lower than the worst case (inner boarder). The shortest track is therefore weighted more. This lets the solution change from least curvature to shortest track finer around small ε, but higher at high ε. The new weighted matrices are calculated then according to equation (2.31).

$$\begin{aligned} H_L &= \frac{(1-\varepsilon)}{w_{LC}}\cdot H_{LC} + \frac{\varepsilon}{w_{ST}} \cdot H_{ST} \qquad (2.31)\\ B_L &= \frac{(1-\varepsilon)}{w_{LC}}\cdot B_{LC} + \frac{\varepsilon}{w_{ST}} \cdot B_{ST} \end{aligned}$$

Indeed, the introduced weighting performed better in terms of lap time. To show this in an example, Table 2.6 shows some of the achieved lap time for different combination factors changing from 0 to 1 in 0.025 steps.
The lap time first decreases slightly and later increases again. With the old, linear scaling, this little time decrease might be jumped. Table 2.6 shows that the new scaling method detects a lap time which is faster.

Combination factor	Lap time - old method	Lap time - new method
0.0	14.88 s	14.88 s
0.05	14.93 s	14.85 s
0.1	14.98 s	14.84 s
0.3	15.34 s	14.82 s
0.5	15.67 s	14.90 s
0.9	16.33 s	15.21 s
1.0	16.46 s	16.46 s

Table 2.6: Achieved lap time for different combination factors and scaling methods.

2.5 Speed Profile

Once the optimal trajectory is known, it is time to make a start on the speed profile.

2.5.1 Problem Formulation

The problem formulations made in [Col09] are shown below.

$$\begin{aligned}
\text{min.} \quad & \int_0^T dt = \int_0^1 \frac{d\theta}{ds(\theta)} \qquad \theta \in [0,1] \\
\text{s.t.} \quad & a_{long,min}(\theta) \leq a_{long}(\theta) \leq a_{long,max}(\theta) \\
& 0 \leq a_{lat}(\theta) \leq a_{lat,max}(\theta) \\
& s_{min}(\theta) \leq s(\theta) \leq s_{max}(\theta) \\
& \\
& |a_{lat}| = \frac{s^2}{R} \\
& a_{long} = \dot{s} = \frac{ds}{dt} = \frac{ds}{d\theta}\frac{d\theta}{dt} = s's = \frac{1}{2}(s^2)'
\end{aligned} \tag{2.32}$$

$$\begin{aligned}
\text{min.} \quad & \sum_{i=0}^{N-2} \Delta t_i = \sum_{i=0}^{N-2} \frac{\Delta\theta_i}{\frac{s_{i+1}+s_i}{2}} \\
\text{s.t.} \quad & a_{long,min,i} \leq a_{long,i} \leq a_{long,max,i} \quad \forall i \in \{0,1,\cdots,N-2\} \\
& 0 \leq a_{lat,i} \leq a_{lat,max,i} \\
& s_{min,i} \leq s_i \leq s_{max,i} \quad \forall i \in \{0,1,\cdots,N-1\} \\
& \\
& |a_{lat,i}| = \frac{s_i^2}{R} \\
& a_{long,i} = \frac{\Delta s_i}{\Delta t_i} = \frac{\Delta s_i}{\Delta\theta_i}\frac{\Delta\theta_i}{\Delta t_i} = \frac{s_{i+1}-s_i}{\Delta\theta_i}\frac{s_{i+1}+s_i}{2} = \frac{s_{i+1}^2-s_i^2}{2\Delta\theta_i}
\end{aligned} \tag{2.33}$$

The continuous (equation 2.32) and the discrete (equation 2.33) problem have quadratic equality constraints. Convex optimization problems require affine equality constraints. Introducing a new optimization variable "squared speed" instead of speed,

$$\tilde{s_i} := s_i^2 \tag{2.34}$$

transforms the problem into a convex form (equation 2.35). The cost function formulation in the new variable $\tilde{s}$ has square roots in the denominator and is therefore a semi-definite program (SDP). [Col09] suggests to change the objective function in an intuitive, heuristic way. Maximizing the sum of *squared* speed samples should solve the same as minimizing the lap time, since higher speeds allow to travel faster. The cost function of equation (2.35) is therefore replaced by equation (2.36) and simplifies the SDP to a linear program (LP). Constraints on the upper speed limit does not make sense at the moment because the car should drive as fast as possible to achieve fast lap times. It is left in the optimization problem as an other degree of freedom, for example if one wants to drive behind an other car whose performance is weaker, or if car models are verified which are only valid up to certain speed limits, etc.

$$\begin{aligned}
\text{min.} \quad & \sum_{i=0}^{N-2} \frac{2\Delta\theta_i}{\sqrt{\tilde{s}_{i+1}} + \sqrt{\tilde{s}_i}} \\
\text{s.t.} \quad & a_{long,min,i} \leq a_{long,i} \leq a_{long,max,i} \quad \forall i \in \{0, 1, \cdots, N-2\} \\
& 0 \leq a_{lat,i} \leq a_{lat,max,i} \\
& s^2_{min,i} \leq \tilde{s}_i \leq s^2_{max,i} \quad \forall i \in \{0, 1, \cdots, N-1\} \\
& \\
& |a_{lat,i}| = \frac{\tilde{s}_i}{R} \\
& a_{long,i} = \frac{\tilde{s}_{i+1} - \tilde{s}_i}{2\Delta\theta_i}
\end{aligned} \tag{2.35}$$

$$\begin{aligned}
& \text{max.} \quad \sum_{i=0}^{N-2} \tilde{s}_i \\
\Leftrightarrow \quad & \text{min.} \quad -\sum_{i=0}^{N-2} \tilde{s}_i
\end{aligned} \tag{2.36}$$

For several tracks, the sum of the absolute difference of the two solutions (LP versus SDP) for horizon length of 100 was less than 10^{-5}. Although the cost function are different, they produce identical solutions. Therefore, the simplification is valid and does not introduces any loss of optimality.

2.5.2 Problem Structure

Speed Profile - depended on $\tilde{s}$

Inserting the equality constraints into the inequality constraints gives the following structure.

$$\begin{aligned}
\text{min.} \quad & c^T \cdot \tilde{s} \\
\text{s.t.} \quad & 2 \cdot \Delta\theta_i \cdot a_{long,min,i} \leq \tilde{s}_{i+1} - \tilde{s}_i \leq 2 \cdot \Delta\theta_i \cdot a_{long,max,i} \\
& 0 \leq \tilde{s}_i \leq a_{lat,max,i} \cdot R \\
& s^2_{min,i} \leq \tilde{s}_i \leq s^2_{max,i} \\
& \\
\text{where} \quad & c = [-\mathrm{I}_{N\times 1}]
\end{aligned} \tag{2.37}$$

Speed Profile - depended on $\tilde{s}$ and a_{long}

The optimizer variable contains the squared speed sample and the longitudinal acceleration.

$$\tilde{x} = \begin{bmatrix} \tilde{s} \\ a_{long} \end{bmatrix} \tag{2.38}$$

The problem is equal to equation (2.35) with the cost function of equation (2.36) and summarized in equation (2.39).

$$\begin{aligned} \text{min.} \quad & c^T \cdot \tilde{x} \\ \text{s.t.} \quad & a_{long,min,i} \leq a_{long,i} \leq a_{long,max,i} \\ & 0 \leq \tilde{s}_i \leq a_{lat,max,i} \cdot R \\ & s^2_{min,i} \leq \tilde{s}_i \leq s^2_{max,i} \\ & a_{long,i} = \frac{\tilde{s}_{i+1} - \tilde{s}_i}{2\Delta\theta_i} \\ \text{where} \quad & c = \begin{bmatrix} -\mathrm{I}_{N\times 1} \\ 0_{N\text{-}1\times 1} \end{bmatrix} \end{aligned} \tag{2.39}$$

Although the second method has more optimization variables, the MATLAB-CPLEX-solver of the MPT-Toolbox was almost as fast as for the first method with only $\tilde{s}$ as optimizer. The solver made use of the well structured problem but time loss due to higher dimension and gain due to nicer structure balanced each other. Table 2.7 shows the small differences in time consumption for both methods.

Method	CPLEX-solver time for a horizon of 30 steps
Depended on $\tilde{s}$	0.82 ms
Depended on $\tilde{s}$, a_{long}	0.85 ms

Table 2.7: Speed profile: CPLEX-solver time to solve the two different methods for a horizon of 30 steps.

3 Algorithm implementation

The above derived optimization problems are finally combined into an algorithm. The code is implemented in MATLAB and in C++.
MATLAB is simpler to code, easier to cope with and has nice properties to plot and therefore visually check the results. It is an optimal development environment. On the other hand, the algorithm needs to run as fast as possible and is therefore also implemented in C++. The NAG-C Library provides optimization and linear system equations functions for C++ and is therefore used in this project.

Figure 3.1 shows a sketch of the algorithm. It is divided into two parts. The "offline" method parameterizes, defines and calculates the track and all the parameters required to run the second part, the "online" method, continuously in a loop.

3.1 Offline calculation

The following functions are called once and serve for preparation for the "online" part.

3.1.1 Track generation

This functions reads all the Cartesian coordinates at each sample point for the inner and outer boarder, as well as for the centerline. It also calculates the distance between the actual and the previous centerline sample.
In this project, four different closed tracks are used, two them are Figure 2.3 and 2.4. Any other track build with blocks of straights and 90°-curves could be built with the MATLAB version.

3.1.2 Offline_factors

The optimal trajectory calculation as introduced above, requires scaling factors for the shortest track and least curvature method and a combination factor to derive this optimal trajectory.
The best combination factors is the one, that gives the shortest lap time. It is determined with a brute force method, which solves the trajectory, speed optimization and calculates the lap time for combination factors between 0 and 1 increasing in 0.025 steps
.
The trajectory with the best lap time is later also used to introduce terminal state constraints, to ensure the optimization at horizon N coincide with the best lap time

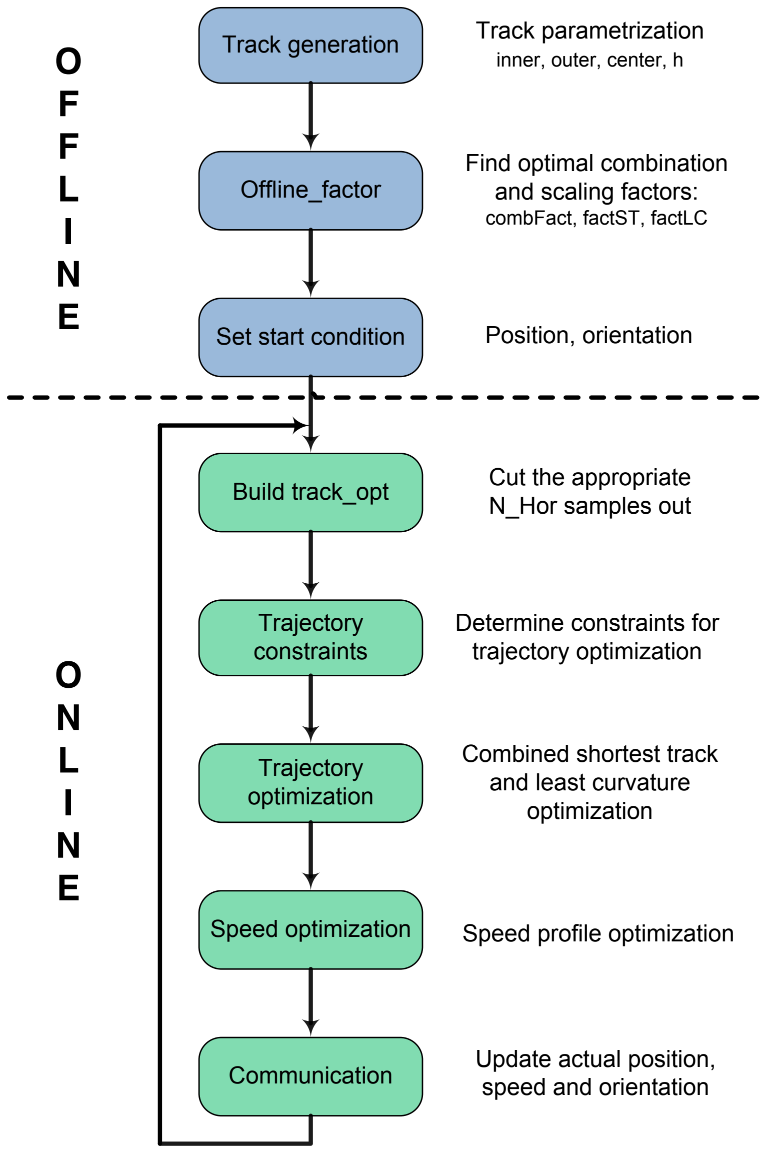

OFFLINE
Track generation
Track parametrization
inner, outer, center, h
Offline_factor
Find optimal combination and scaling factors:
combFact, factST, factLC
Set start condition
Position, orientation
ONLINE
Build track_opt
Cut the appropriate N_Hor samples out
Trajectory constraints
Determine constraints for trajectory optimization
Trajectory optimization
Combined shortest track and least curvature optimization
Speed optimization
Speed profile optimization
Communication
Update actual position, speed and orientation

Figure 3.1: Algorithm overview.

trajectory.

All the so far gathered information are stored into a struct. Table 3.1 shows the structure.

Struct Track	Information
track.N	Number of samples.
track.inner	Cartesian coordinates of inner track boarder.
track.outer	Cartesian coordinates of inner track boarder.
track.center	Cartesian coordinates of track centerline.
track.h	Distance between centerlines.
track.alpha_opt	α for the best lap time trajectory.
track.factST	Scaling factor for the shortest track method.
track.factLC	Scaling factor for the least curvature method.
track.combFact	Time optimal combination factor.

Table 3.1: Structure of the parameterized track.

3.2 Online calculation

3.2.1 Build track_opt

The optimization will later be carried out for a finite horizon N_{hor}. The track_opt has the same structure as the track in Table 3.1 but contains only the data for the appropriate N_{hor} steps.
The function "build track_opt" determines the first sample (blue circle in Figure 3.2) before the actual position (red cross). If the position is not too close to an existing sample, the algorithm creates a new starting sample through the actual position, as seen in the right picture of Figure 3.2.
The next $N_{hor} - 1$ values are cut out from the struct describing the entire track and copied to the struct track_opt.

3.2.2 Trajectory constraints

In this project, three different constraints on the relative position (α) between the boarder lines of the track are introduced.
The first constraint is α_{start} and is obviously determined by the actual car position.
Whereto drive from the starting point are the second implemented constraints. The communication with the camera system should deliver information about the orientation and speed of the car. Equation (2.13) in the previous chapter showed the relation between speed, curvature radius and lateral acceleration. Since the maximal lateral acceleration is assumed to be known, the maximal curvature radius can be calculated and therefore constraints on α.
The last type of constraints are terminal constraints. Since the offline method provides

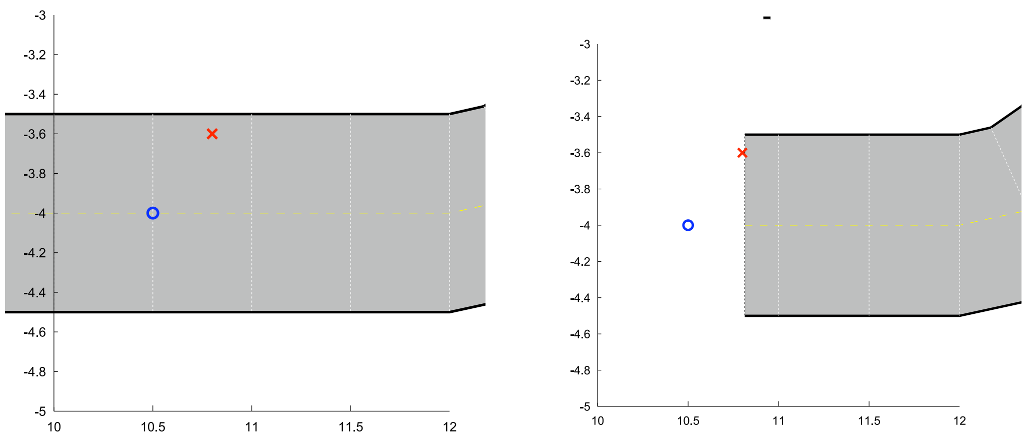


Figure 3.2: New sample through actual position (red).

a trajectory which gives the best lap time, it makes sense to require the solution of the finite horizon to coincide at the end with the best lap time trajectory. To adopt also the shape of this optimal trajectory, more than one α have to be set.

Figure 3.3 shows the optimized trajectory with all of the introduced constraints. The dashed blue line is the best lap time trajectory determined offline. The red dashed line the optimized trajectory for a horizon of 30 steps. The left hand side of Figure 3.3 is an enlarged view of the right hand side picture.
The purple line shows the actual orientation of the car, the dark circle the maximal curvature radius. When the dark circle crosses a discretized track piece, the relative distance α is constrained and marked with little yellow circles.
The terminal constraints force the red and blue trajectory to coincide at the end, as seen in the right hand side of the Figure.

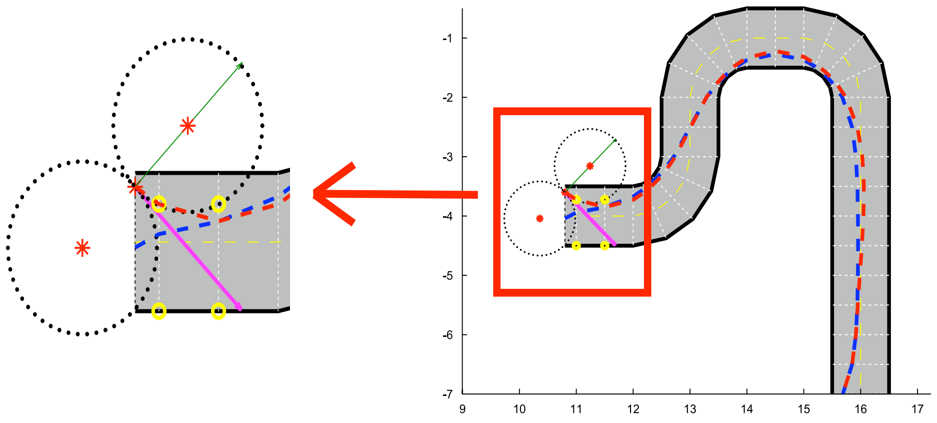


Figure 3.3: Optimized track with all constraints for a horizon of 30 steps.

3.2.3 Trajectory optimization

Chapter 2.3 showed that an iterative procedure is required to derive the least curvature trajectory. Figure 3.4 shows a detailed overview of the "Trajectory optimization" block. The shortest track method is independent of the actual trajectory parameterization and is therefore calculated once at the beginning. In a loop, the shortest track and least curvature matrices are combined with the already determined factors and passed to the QP-solver, which optimizes the problem with the above mentioned additional constraints. With the optimized variable α, the new trajectory centerline and their differences to each other (track.h) are calculated and scaled. After a defined number of iterations, the algorithms exits the loop.

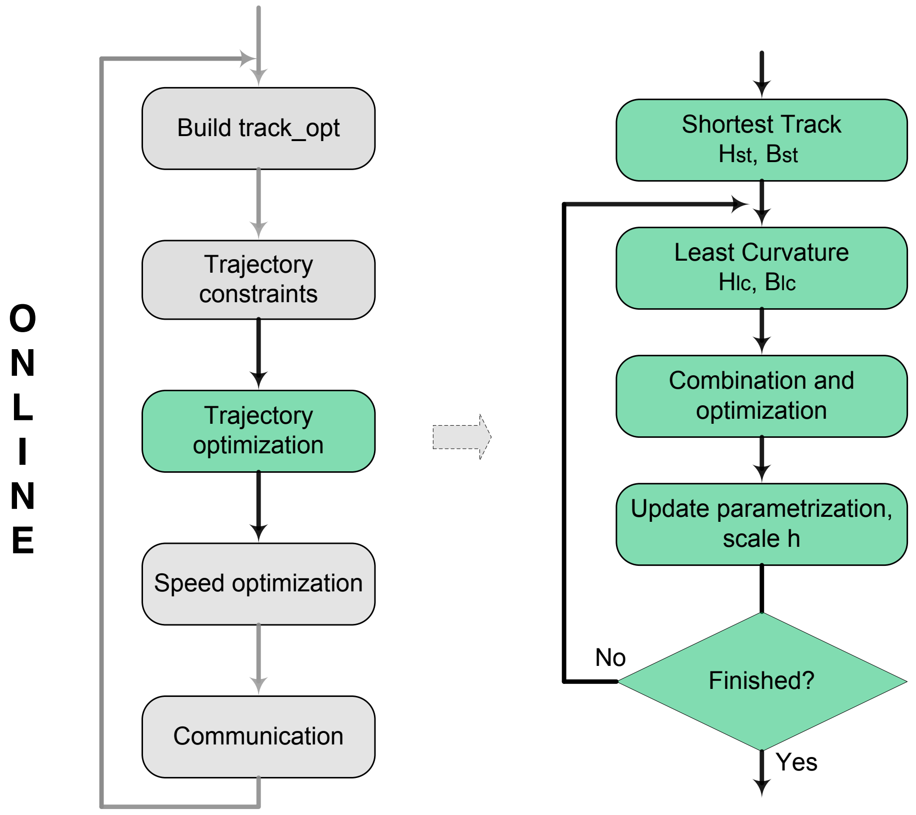


Figure 3.4: Detailed overview of the trajectory optimization block.

3.2.4 Communication

This part has not been implemented yet due to simultaneously ongoing projects on the camera system and controller implementation.
It contains two parts, deliver the optimized values and receive the actual position, speed and orientation. With this new information, the loop can be closed and the optimization

start again.

3.3 Measured time consumption

The code is implemented in MATLAB and C++. The time consumptions for all mentioned functions are shown in Table 3.2. The ”online method” in C++ is about 100 times faster than the MATLAB version and lasts 3.2 milliseconds. A 50 Hz controller would have another 16.8 millisecond left for communication, including other cars, etc.

Environmet	MATLAB	MATLAB	C++
Solver	Quadprog	Cplex	NAG
Test Platform	2.53 GHz PC	IfA Server	2.53 Ghz PC
Total Offline-calculations	37’161 ms	11’240 ms	7757 ms
Total Online-calculations	435 ms	336 ms	3.2 ms
Build track_opt	265 ms	250 ms	0.02 ms
Trajectory constraints	74 ms	63 ms	0.07 ms
Trajectory optimization	80 ms	17 ms	1.7 ms
Speed optimization	16 ms	6 ms	1.4 ms

Table 3.2: Time consumption for all functions in MATLAB and C++.

4 Conclusion and outlook

4.1 Summary

Simplifications introduced by a previous semester project made for convex and fast solvable problems were investigated. Most of them proved to be reasonable and only slightly suboptimal. Small changes and extensions were made to avoid ill conditioned matrices and reasonable combing the shortest track and least curvature method.
Each problem structures was defined in two ways, one with inserted equality constraints and therefore only depending on a low dimension optimizer variable, an other with extended optimizer variable, higher dimension, equality constraints and sparse structure. Compared solver time consumption for both cases showed that additional time due to higher dimension was greater than the saved time gained by the sparse structure.
Additional functions to determine the appropriate part of the track depended on the actual car position and constrains on the trajectory according to orientation, speed and end position were introduced. Finally, all these functions were implemented in an easy-to-use environment (MATLAB) and in a complex, but very fast executable program (C++). The C++ version needs 3.2 milliseconds which is far under the limit of the desired 20 - 50 Hz controller.

4.2 Outlook

The optimal path planning is part of an entire system including car modeling, tracking, controlling and communication between all these. A superior controller feeds the optimal path planning with information (speed, position and orientation) and receives the optimized variables.
The next possible step concerning the entire project is to merge all these mentioned projects. One routine which takes the picture, obtains the actual information, passes them to the path planning algorithm, takes the gained optimal values and feeds them into the controller and finally adjusts the right voltages for steering and driving the 1:43 scale race cars.
Concerning only the optimal path planning, more elaborate models of the car could impose additional constraints on the optimization problems. For example maximal steering angle, maximal curvature radius, speed depended maximal acceleration, etc.
To compete with other race cars simultaneously, position, speed, orientation and possible prediction of the competitors movement have to be transformed into additional constraints. The controller could also be extended to control several cars simultaneously.

List of Figures

Bibliography

[Col09] Colas, Stéphane: *A Convex Optimization Approach To Time-Optimum Path Planning*. Semester project, September 2009

[FB07] F. Braghin, S. Melzi E. S. F. Cheli C. F. Cheli: *Race driver model*. Politecnico di Milano, Department of Mechanics, Via La Masa 34, 20158 Milan, 2007

[QD] Q.T. Dinh, M. D.: *An Application of Sequential Convex Programming to Time Optimal Trajectory Planning for a Car Motion*. Optimization in Engineering Center (OPTEC) and Departtment of Electrical Engineering, Katholieke Universiteit Leuven, Belgium,

[Wik10] Wikipedia: *Spline Interpolation*. Website. http://en.wikipedia.org/wiki/Spline_interpolation. Version: March 2010. – Date: 04.06.2010