Raceline Optimization

Hongrui Zheng (hongruiz@seas.upenn.edu)

Penn Engineering

Acknowledgements

- This lecture is based on the following works from TUM and Penn.
- https://www.researchgate.net/publication/352350798_Autonomous_Driving_Software_Engineering_-_Lecture_06_Planning_I_-_Global_Planning
- https://par.nsf.gov/servlets/purl/10221876
- https://arxiv.org/pdf/2202.13525.pdf

Overview

- Problem Definition
- Raceline Optimization w/ Explicit Optimization
- Raceline Optimization w/ Evolutionary Strategy
- Example: Levine Hall

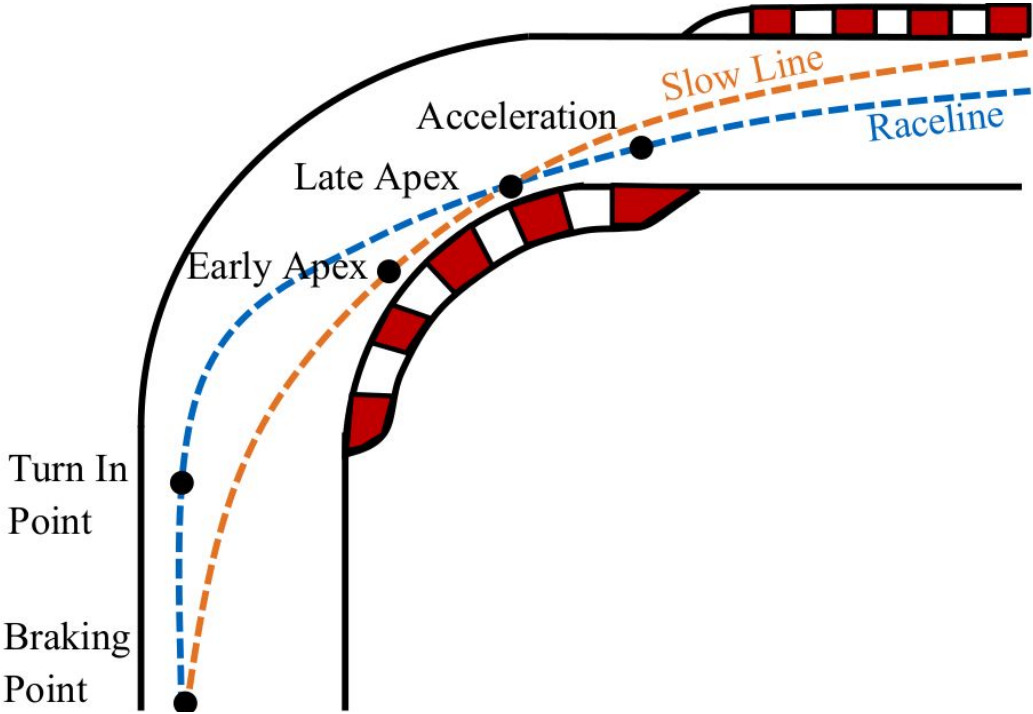
Problem Definition

The Raceline: Problem and Solution

Problem:
What is the fastest way to get around the track?

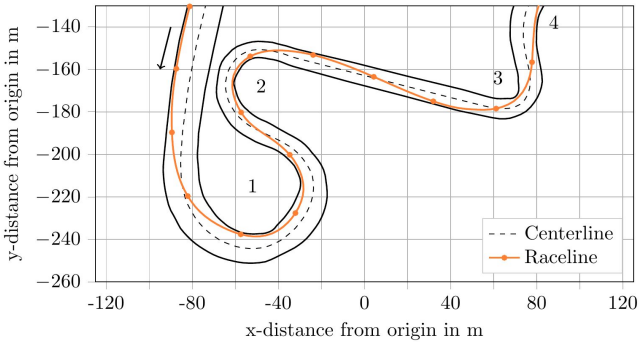
Problem (again):
How to find the best configuration of a global plan that gives us the lowest lap time. A search problem!

How:
Optimization of some function over some variables

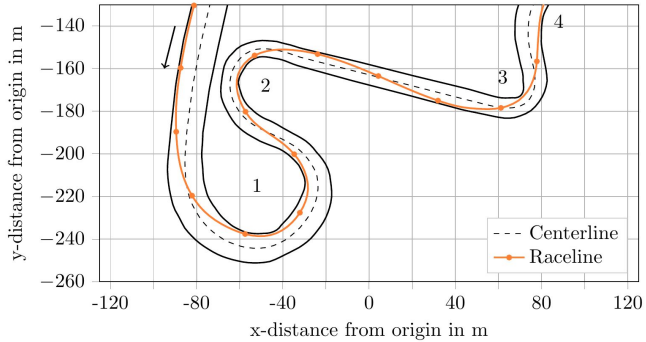


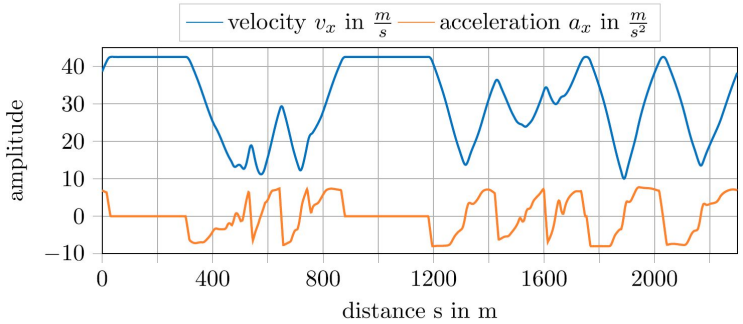
Review: Path vs. Trajectory

Path planning: [x, y], spatial



Trajectory planning: [x, y, v], spatial-temporal



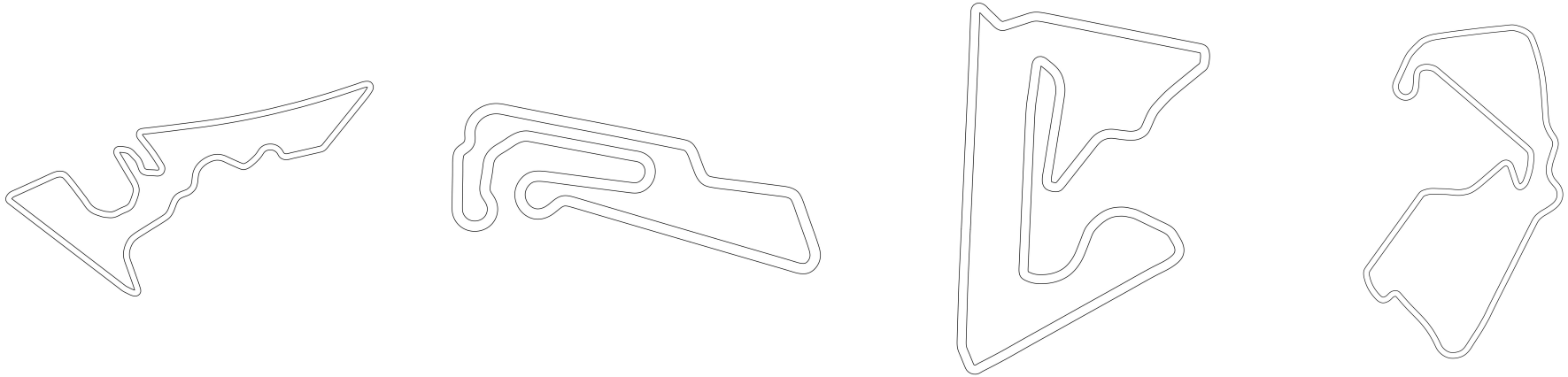


Penn Engineering

F1TENTH Track Database

F1TENTH Track Database:

- We have a database of downscaled real-world racetracks
- Currently **20 race tracks** available
- Real racetrack layouts (x- and y-waypoints)
- Global optimal racelines for each racetrack
- https://github.com/f1tenth/f1tenth_racetracks



Raceline Optimization

Raceline Optimization

As a search problem

Search variables:

Path + Velocities, (sometimes) hardware configuration, (sometimes) control strategies

Constraints:

Vehicle dynamic limits, static obstacles

How:

QPs: Min. distance, min. curvature, Evolution Strategies, Optimal Control: min. time (won't discuss today)

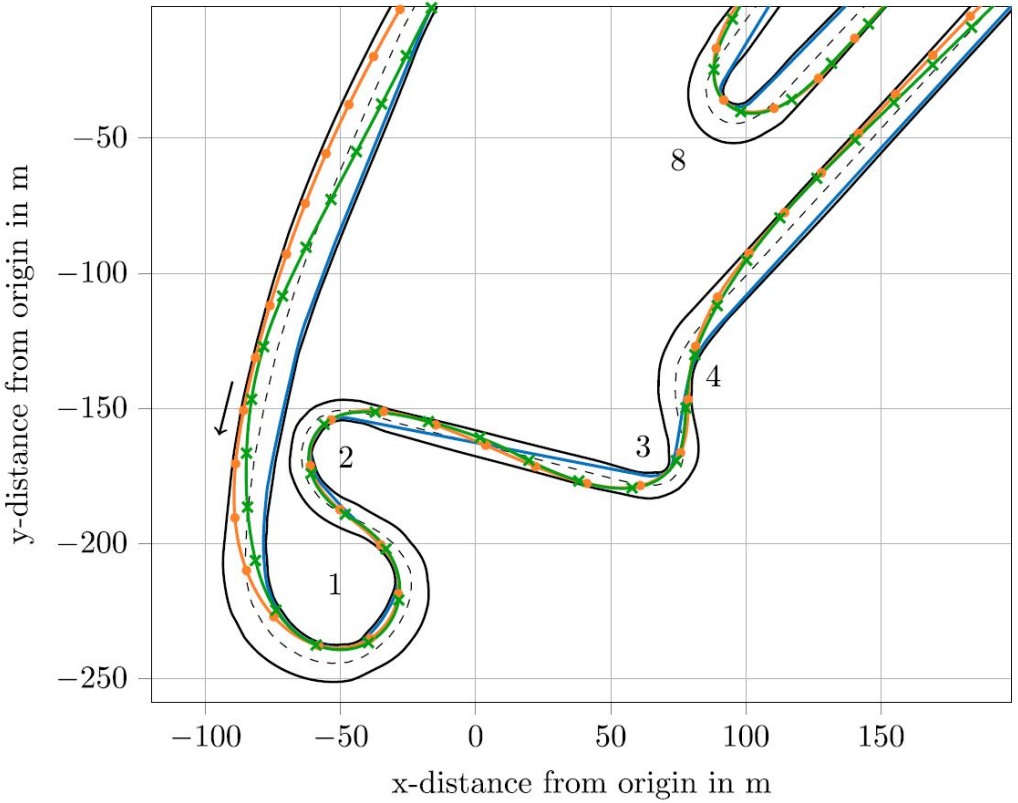
Raceline Optimization: Explicit Methods

Different approaches possible: Optimization

- Shortest Path problem (geometric QP)
- Minimum curvature problem (geometric QP)
- Minimum time problem (optimal control)

Solutions differ in computation runtime

Solutions differ in vehicle dynamics depth modelling



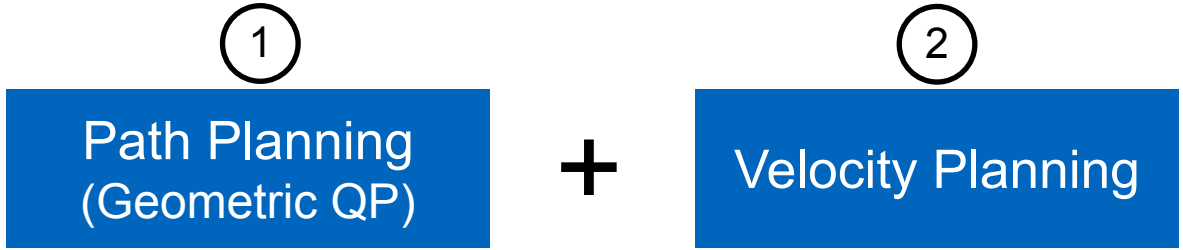
Penn Engineering

Source: Minimum curvature trajectory planning and control for an autonomous race car, DOI: 10.1080/00423114.2019.1631455

Breaking Down the Problem

Creation of the trajectory by means of optimization:

Solving **two** subsequent sub-problems:



- Minimum distance path(Geometric QP) + Velocity Planning
- Minimum curvature path(Geometric QP) + Velocity Planning

Advantages of Geometric QP problems:

- Few parameters required
- Fast calculation times

QP = Quadratic Programming

Optimization Problem: Quadratic Programming

- Quadratic programming is a process to solve mathematical optimization problems **involving quadratic functions**
- The method minimizes or maximizes a target value/objective function
- **Goal:** Finding a vector x that minimizes a quadratic function
- Certain boundary conditions must be maintained during optimization

$$\min f(x) = \frac{1}{2} x^T Q x + c^T x$$

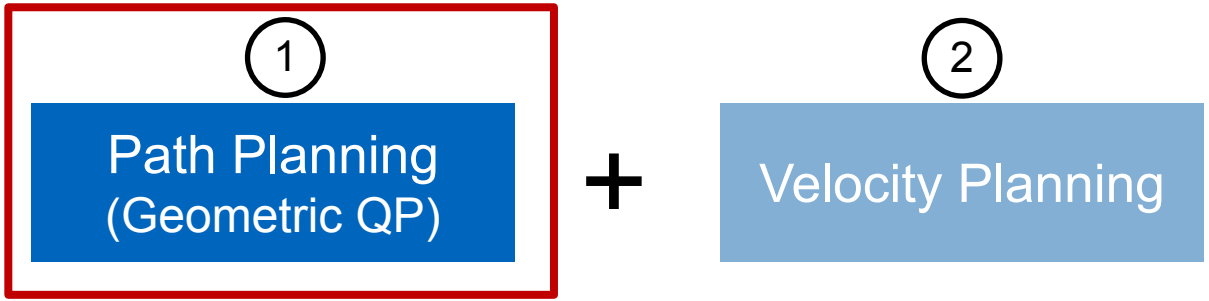
$$subject\ to: \quad Ax \leq b$$

$$x \geq 0,\ c, x \in \mathbb{R}^n, A_{m \times n}, Q_{m \times n}$$

Raceline Optimization: Minimum Curvature

Creation of the trajectory by means of optimization:

Solving **two** subsequent sub-problems:



- Minimum curvature line (Geometric QP) + Velocity Planning

QP = Quadratic Programming

Track representation

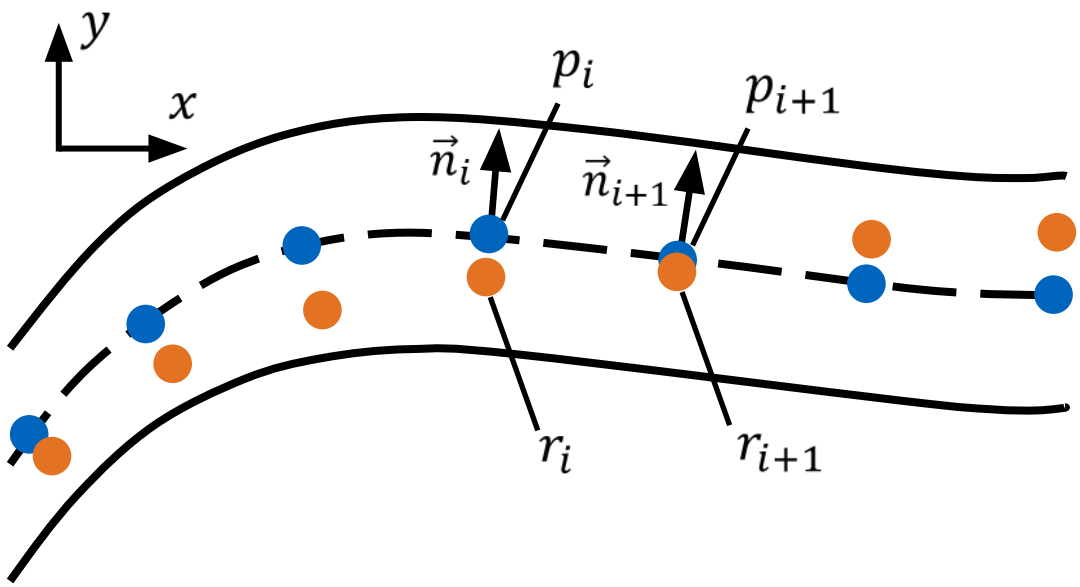
- Representation by centerline points and track widths
- Discretization distance dependent on use case (size of track)

Penn Engineering

Raceline Representation

Shifting each point along its normal vector within the track widths (variable α in our case)

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



<u>Minimization of a given objective function</u>:

- E.g.: Minimizing the sum of the curvature values for all discretization points
- Consideration of the vehicle width (+ safety distance)

$$\alpha_i \in \left[-w_{\text{tr,left},i} + \frac{w_{\text{veh}}}{2}, w_{\text{tr,right},i} - \frac{w_{\text{veh}}}{2} \right]$$

i: discretization step
w_{veh}: vehicle width
w_{tr}: track width

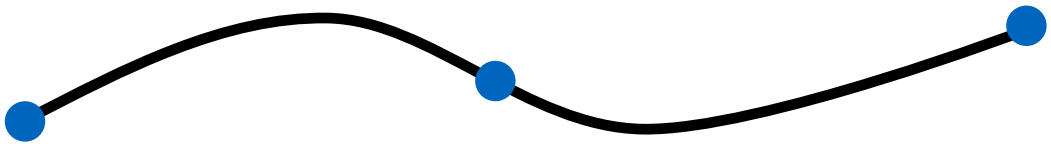
Penn Engineering

Review: Curvature Continuous Cubic Splines

- Many approaches in the field of path planning are based on splines
- Third order polynomials = continuous description
  → Possible to assure heading and curvature continuity
- Analytical calculation of the exact derivatives for…
  - tangent vectors,
  - normal vectors,
  - heading,
  - curvature

  …possible.

→ This is **the prerequisite** for our approach.



Curvature Continuous Cubic Splines

- Mostly, we rely on separated polynomials to represent x- and y-coordinates of the i-th segment / spline (in the further process only the x-component is shown):

$$x_i(t) = a_i + b_i t + c_i t^2 + d_i t^3$$

$$x_i'(t) = b_i + 2c_i t + 3d_i t^2$$

$$x_i''(t) = 2c_i + 6d_i t$$

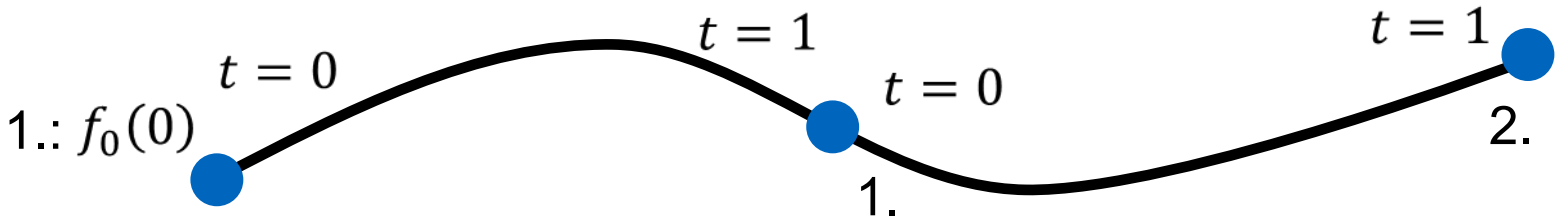
- The equations set up in dependence of the normalized curvilinear parameter t, which is defined as follows (s is the curvilinear distance along the centerline):

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

Curvature Continuous Cubic Splines

We have several boundary conditions:

1. Start of spline should be placed on current discretization point
2. End of spline should be placed on subsequent discretization point
3. Heading continuity, i.e. heading at the end of the spline should be equal to the heading at the beginning of the next spline
4. Curvature continuity, i.e. curvature at the end of the spline should be equal to the curvature at the beginning of the next spline



2.: $f_0(1) = f_1(0)$

3.: $f_0'(1) = f_1'(0)$

4.: $f_0''(1) = f_1''(0)$

Setting up linear Equation System

- Converting the boundary conditions into equations:

1. $t = 0$: $a_i = x_i$
2. $t = 1$: $a_i + b_i + c_i + d_i = x_{i+1}$
3. $t = 1$ and $t = 0$: $b_i + 2 \cdot c_i + 3 \cdot d_i - b_{i+1} = 0$
4. $t = 1$ and $t = 0$: $2 \cdot c_i + 6 \cdot d_i - 2 \cdot c_{i+1} = 0$

- The equations can be split up into a coefficient matrix M and the parameter matrix A (b contains the right hand side):

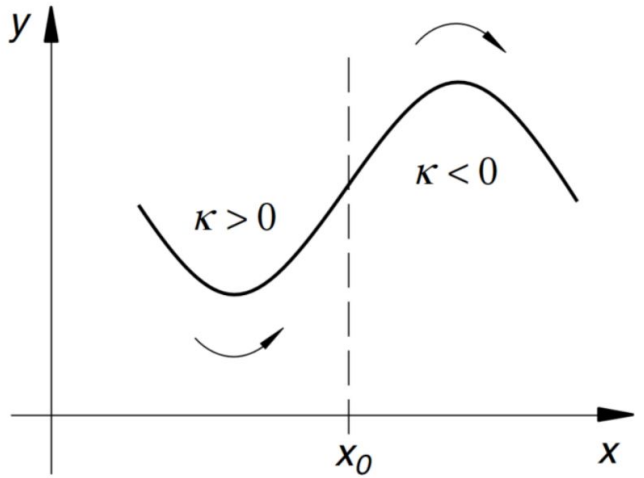
$$M \cdot A = b$$

Setting up linear Equation System

- Having the spline parameters available, we can determine heading and curvature in the 2-D space as follows:
  - Heading: $\psi = \text{atan2}\left(\frac{y'}{x'}\right)$
  - Curvature: $\kappa = \frac{x'y'' - y'x''}{\left(x'^2 + y'^2\right)^{\frac{3}{2}}}$

$\kappa > 0 \leftrightarrow Left\ curvature$

$\kappa < 0 \leftrightarrow Right\ curvature$



Optimization: Shortest Path QP

- Optimization problem:

$$\text{minimize } [\alpha_i \dots \alpha_N] \sum_{i=1}^{N} d_i^2(t)$$

$$\text{subject to} \quad \alpha_i \in [\alpha_{i,\min}, \alpha_{i,\max}] \quad \forall\, 1 \le i \le N$$

- Evaluation solely at the discretization points: $t = 0$

- Calculation of the squared distances:

normal vectors

$$\Delta P_{x,i} = x_{i+1} - x_i + \alpha_{i+1}\Delta x_{i+1} - \alpha_i \Delta x_i$$

$$d_i^2 = \Delta P_{x,i}^T \Delta P_{x,i} + \Delta P_{y,i}^T \Delta P_{y,i}$$

- Several matrix reshaping operations lead to an implementable formulation of the problem (see code on GitHub)

Optimization: Minimum Curvature QP

- Optimization problem:

$$\text{minimize } [\alpha_i \dots \alpha_N] \sum_{i=1}^{N} \kappa_i^2(t)$$

$$\text{subject to} \quad \alpha_i \in [\alpha_{i,\min}, \alpha_{i,\max}] \quad \forall\, 1 \le i \le N$$

- Evaluation solely at the discretization points: $t = 0$

- Calculation of the squared curvature values

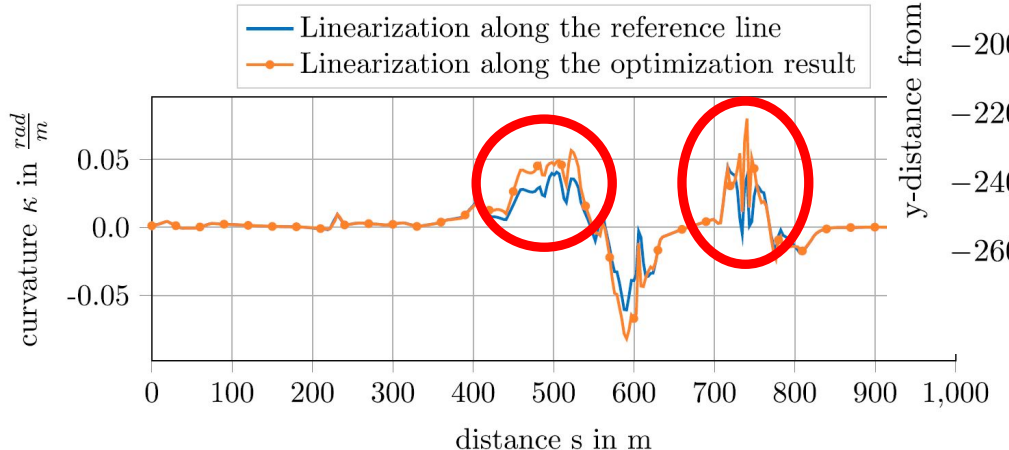
$$\kappa_i = \frac{x_i' y_i'' - y_i' x_i''}{\left(x_i'^2 + y_i'^2\right)^{\frac{3}{2}}} \quad \Rightarrow \quad \kappa_i^2 = \frac{x_i'^2 y_i''^2 - 2x_i' x_i'' y_i' y_i'' + y_i'^2 x_i''^2}{\left(x_i'^2 + y_i'^2\right)^3}$$

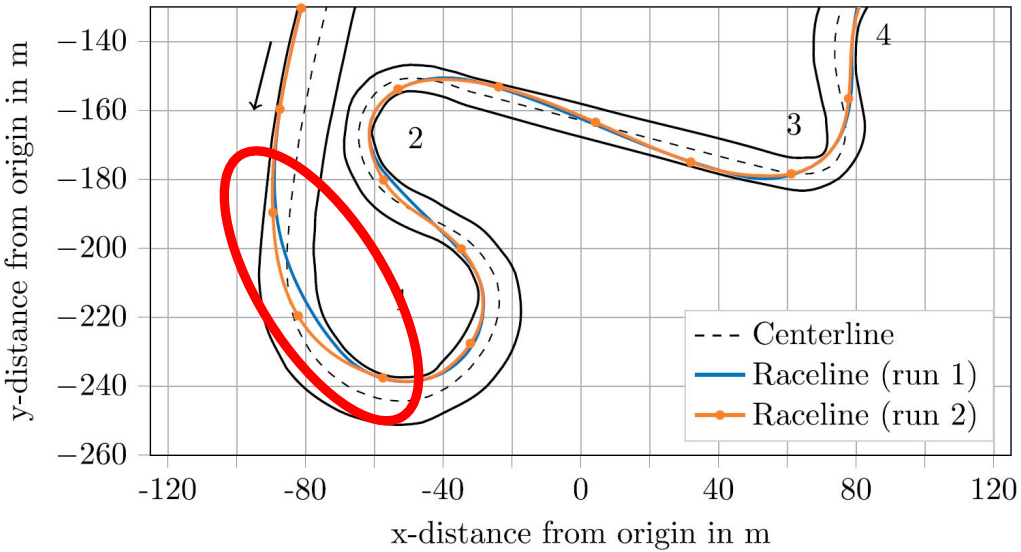
- Several matrix reshaping operations lead to an implementable formulation of the problem (see code on GitHub)

Minimum Curvature Problems

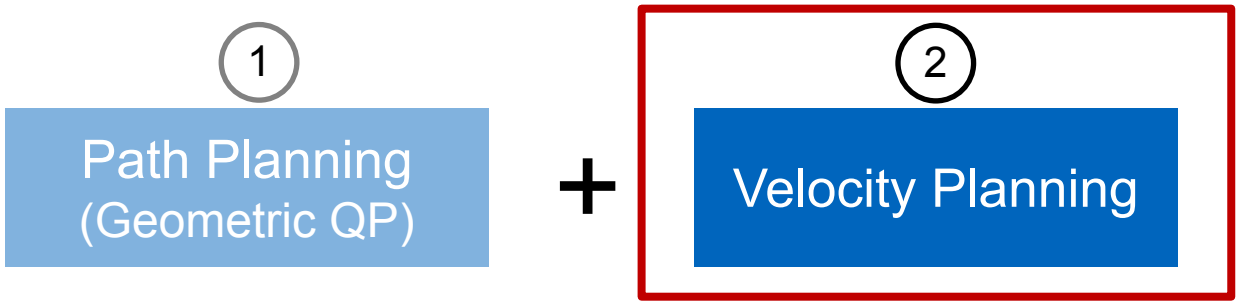
- Linearization errors lead to suboptimal results (and non-compliance with the boundary conditions)

- Can be solved by an iterative invocation of the optimization problem





Velocity Profile Planning



Typical solver types for planning a velocity profile on the race track

- Quasi-steady state solvers → **Forward-Backward Solver**
- Optimization → SpeedOpt (Boyd)

Forward-Backward:

- Subsequent points are solved independently of each other
- Advantages: fast, accurate, less complicated

Velocity Profile Planning

Assumption: The race car is constantly dirven at the vehicle dynamics limits:

- **Longitudinal acceleration a_x** limits result from engine limits (positive) and tires/brakes (negative)
- **Lateral acceleration a_y** limits result from maximum transmittable tire forces

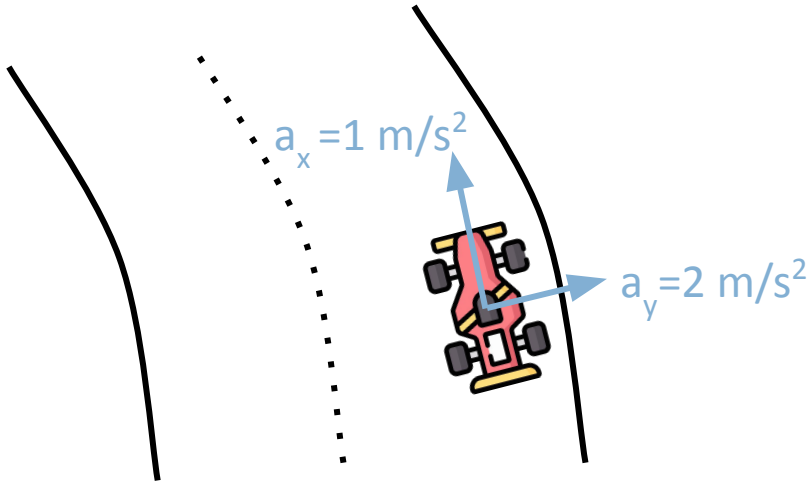
Important Equations:

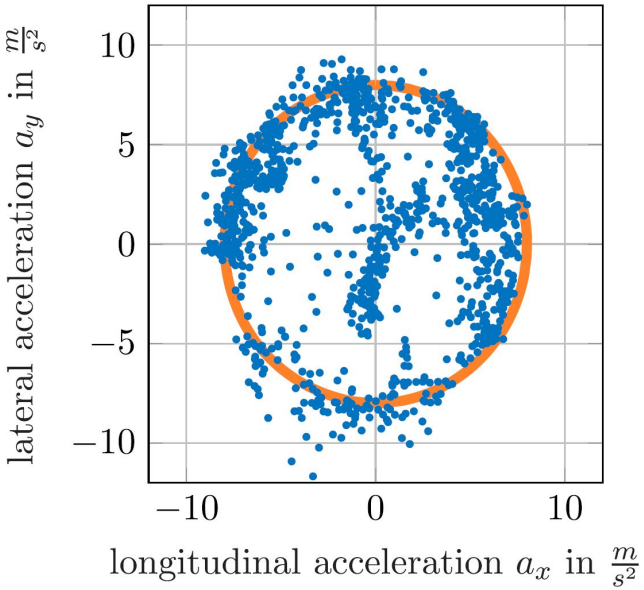
Acting Lateral Acceleration: $$a_y = \frac{v^2}{R} = v^2 \cdot \kappa$$

Possible Longitudinal Acceleration: $$a_x = \frac{F_x}{m}$$

The g-g diagramm

- Simple approaches are usually based on a **g-g diagram,** which connects the possible longitudinal and lateral acceleration (simplified vehicle dynamics)





- A **g-g-v diagram** is an extension that takes the velocity into account as a third dimension (relevant if we want to consider downforce and drag)

Forward-Backward Solver

1. Get first velocity profile estimate based on the maximum possible lateral acceleration for the curvature at every discretization point

2. Clip the profile where the velocity is above the maximum velocity

3. Loop through the discretization points and apply possible positive longitudinal acceleration → **Forward** part

4. Loop through the discretization points and apply possible negative longitudinal acceleration → **Backward** part

Forward-Backward Solver

- In steps 3. and 4. we derive the velocity in the next discretization point as follows (simplified version):
  1. Calculate acting lateral acceleration:

$$a_{y,i} = v_i^2 \cdot \kappa_i$$

  2. Determine remaining potential for longitudinal acceleration:

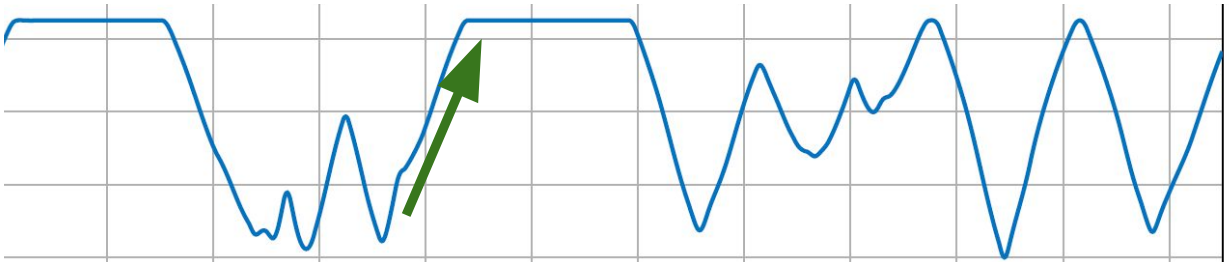
$$a_{x,i} = a_{x,max} \sqrt{1 - \left(\frac{a_{y,i}}{a_{y,max}} \right)^2}$$

  3. Calculate velocity in the next discretization point assuming a constant longitudinal acceleration over element length l_i:

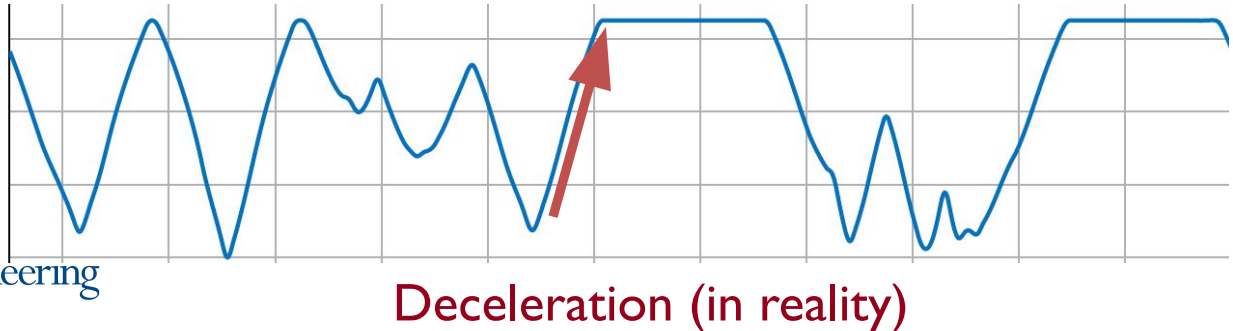
$$v_{i+1} = \sqrt{v_i^2 + 2 \cdot a_{x,i} \cdot l_i}$$

Forward-Backward Solver

- **Inversion** of the velocity profile (and track) allows us to consider the negative longitudinal acceleration (during braking) as if it was a positive acceleration
- **Normal profile** and track for the forward calculation:



- **Inverse profile** and track for the **backward** calculation:

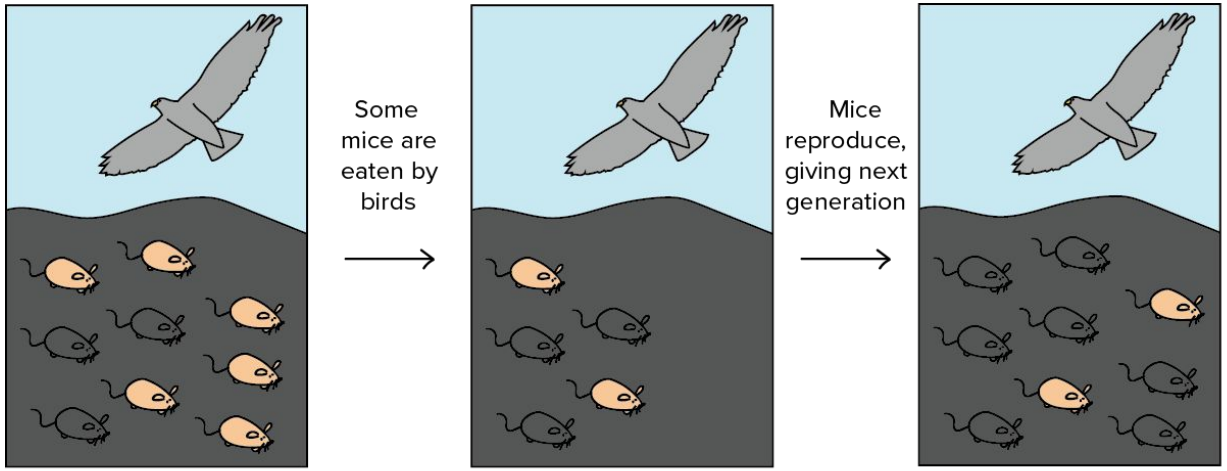


Penn Engineering

Raceline Optimization by Evolution

- Objective function not defined explicitly
- Constraints handled by simulation
- Directly optimizes lap time

Evolution Strategies



A population of mice has moved into a new area where the rocks are very dark. Due to natural genetic variation, some mice are black, while others are tan.

Tan mice are more visible to predatory birds than black mice. Thus, tan mice are eaten at higher frequency than black mice. Only the surviving mice reach reproductive age and leave offspring.

Because black mice had a higher chance of leaving offspring than tan mice, the next generation contains a higher fraction of black mice than the previous generation.

Evolution Strategies

- Gradients for some function f(x) not available, but can still be evaluated deterministically given any x.
- Our belief in the probability distribution over x is $p_\theta(x)$ (fixed format, usually Gaussian).
- Updates θ using the principle of survival of the fittest.

Evolution Strategies

1. Generate a population of samples $D=\{x_i, f(x_i)\}$ where $x_i \sim p_\theta(x)$
2. Evaluate the 'fitness' of samples in D
3. Select the best subset of individuals and use them to update θ
4. Repeat steps 1-3 until convergence

Simple Gaussian Evolution Strategies

- Models $p_\theta(x)$ as n-dim isotropic Gaussian
- $\theta = (\mu, \sigma), p_\theta(x) \sim \mathcal{N}(\mu, \sigma^2 I) \sim \mu + \sigma\mathcal{N}(0, I)$
- Generate offsprings by sampling the Gaussian
- $D^{(t+1)} = \left\{ x_i^{(t+1)} \mid x_i^{(t+1)} = \mu^{(t)} + \sigma^{(t)} y_i^{(t+1)} \text{ where } y_i^{(t+1)} \sim \mathcal{N}(x \mid 0, \mathbf{I}), ; i = 1, \ldots, \Lambda \right\}$
- The λ top samples are marked as D_{elite}
- Update with following rules

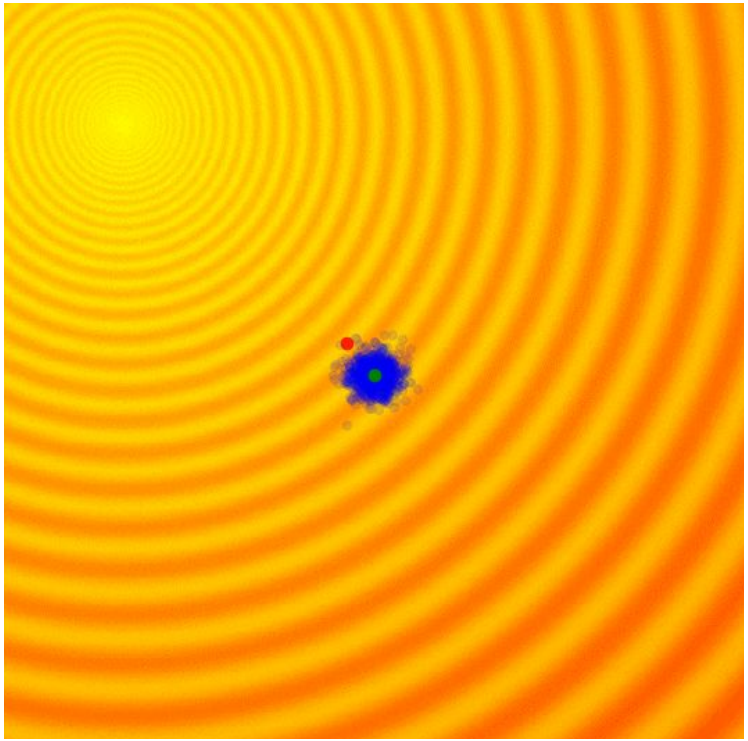
$$\mu^{(t+1)} = \text{avg}\left(D_{\text{elite}}\right) = \frac{1}{\lambda}\sum_{i=1}^{\lambda} x_i^{(t+1)} \qquad \sigma^{(t+1)^2} = \text{var}\left(D_{\text{elite}}\right) = \frac{1}{\lambda}\sum_{i=1}^{\lambda}\left(x_i^{(t+1)} - \mu^{(t)}\right)^2$$

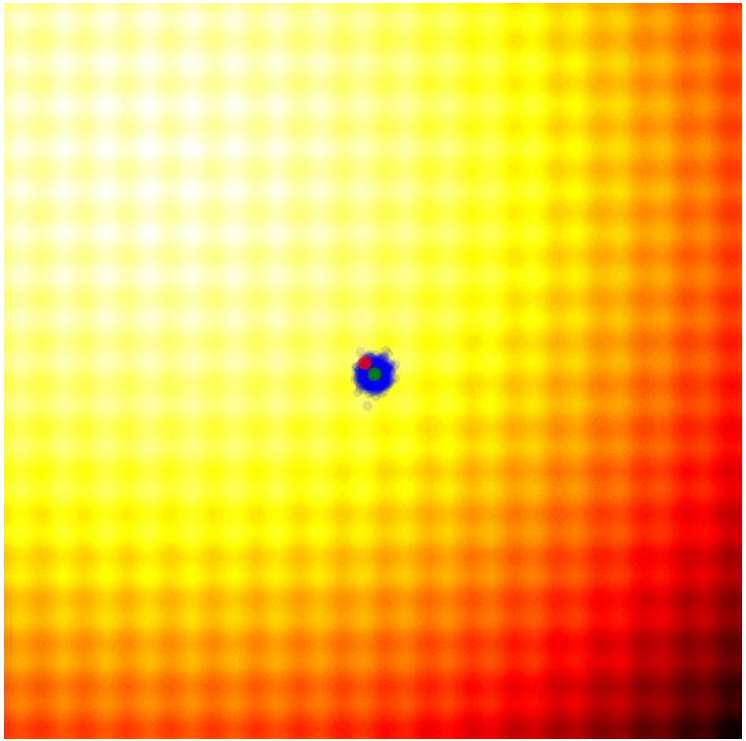
CMA-ES (Covariance Matrix Adaptation Evolution Strategy)

- In simple Gaussian ES, σ_{t+1} highly correlated with σ_t
- Fixes the problem by tracking pairwise dependencies between samples with a covariance matrix C.
- $\theta = (\mu, \sigma, C), p_\theta(x) \sim \mathcal{N}(\mu, \sigma^2 C) \sim \mu + \sigma\mathcal{N}(0, C)$
- Update rule:
- $$\mu^{(t+1)} = \mu^{(t)} + \alpha_\mu \frac{1}{\lambda} \sum_{i=1}^{\lambda} \left(x_i^{(t+1)} - \mu^{(t)} \right)$$
- $$C_\lambda^{(t+1)} = \frac{1}{\lambda} \sum_{i=1}^{\lambda} y_i^{(t+1)} y_i^{(t+1)\top} = \frac{1}{\lambda\sigma^{(t)2}} \sum_{i=1}^{\lambda} \left(x_i^{(t+1)} - \mu^{(t)} \right) \left(x_i^{(t+1)} - \mu^{(t)} \right)^\top$$

More detail and very good explanation here: https://lilianweng.github.io/posts/2019-09-05-evolution-strategies/

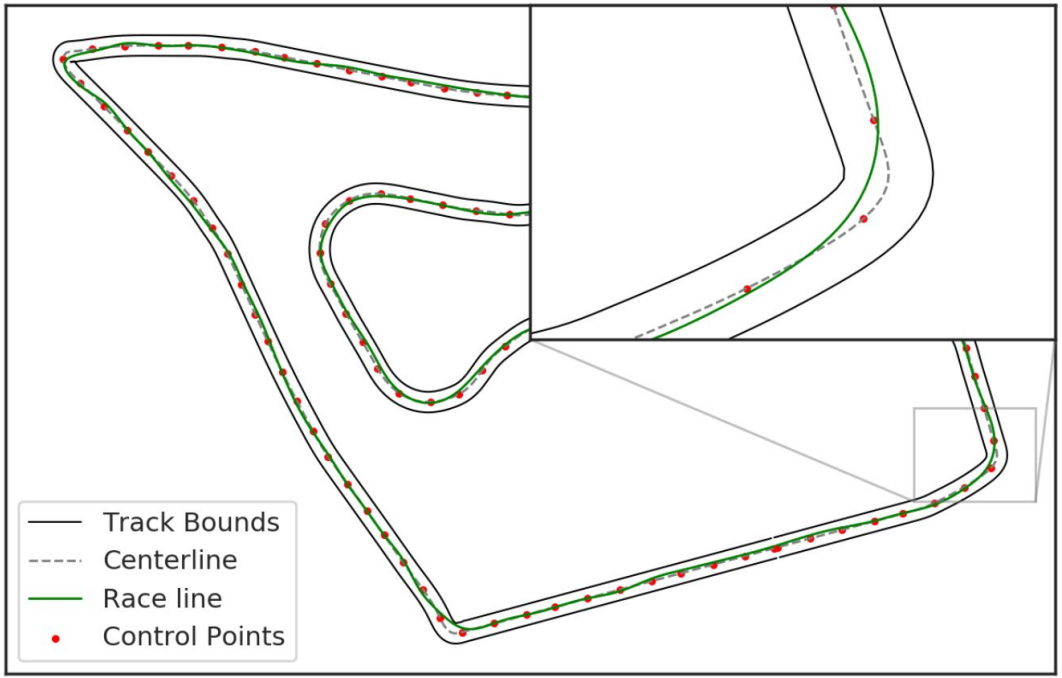
CMA-ES





Source: David Ha: A Visual Guide to Evolution Strategies

Parameterizing the genome

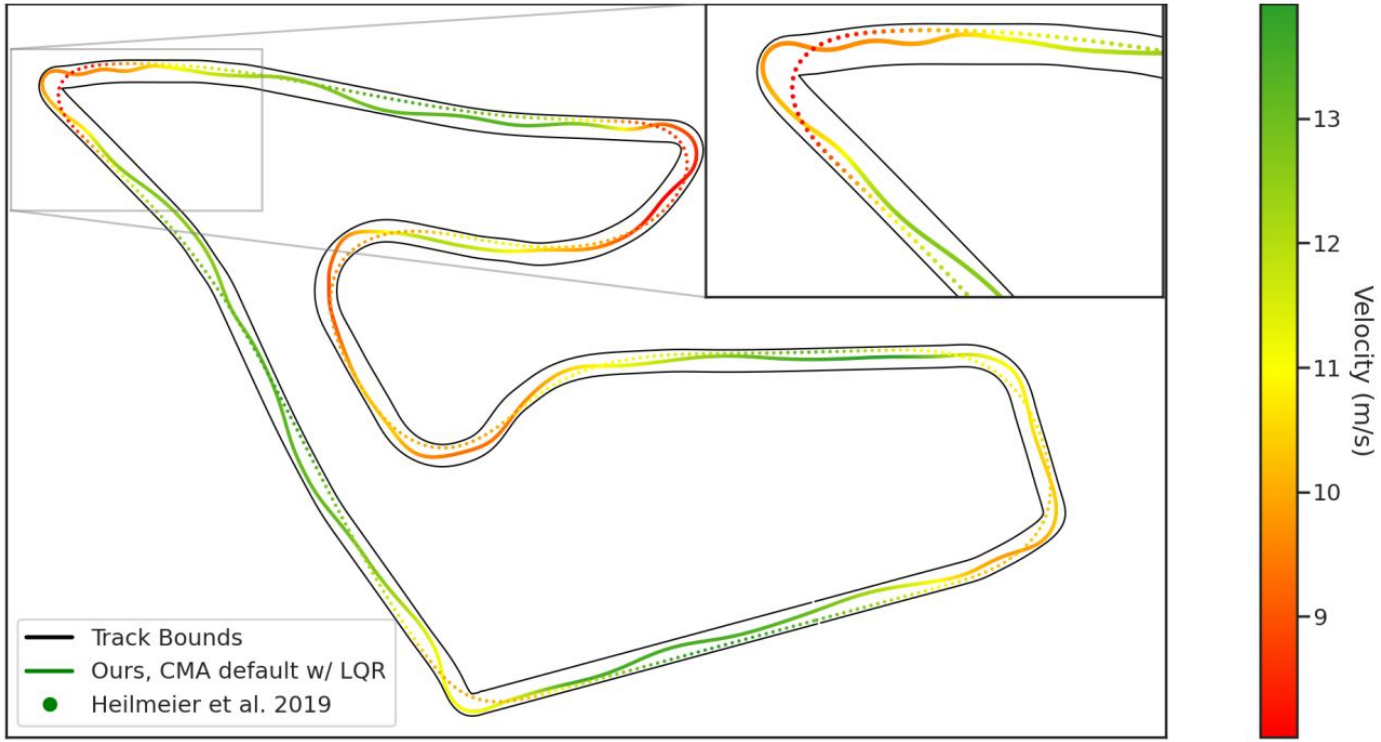


- Similar to the approach in geometric QPs
- Shift control points of spline along the normal of centerline

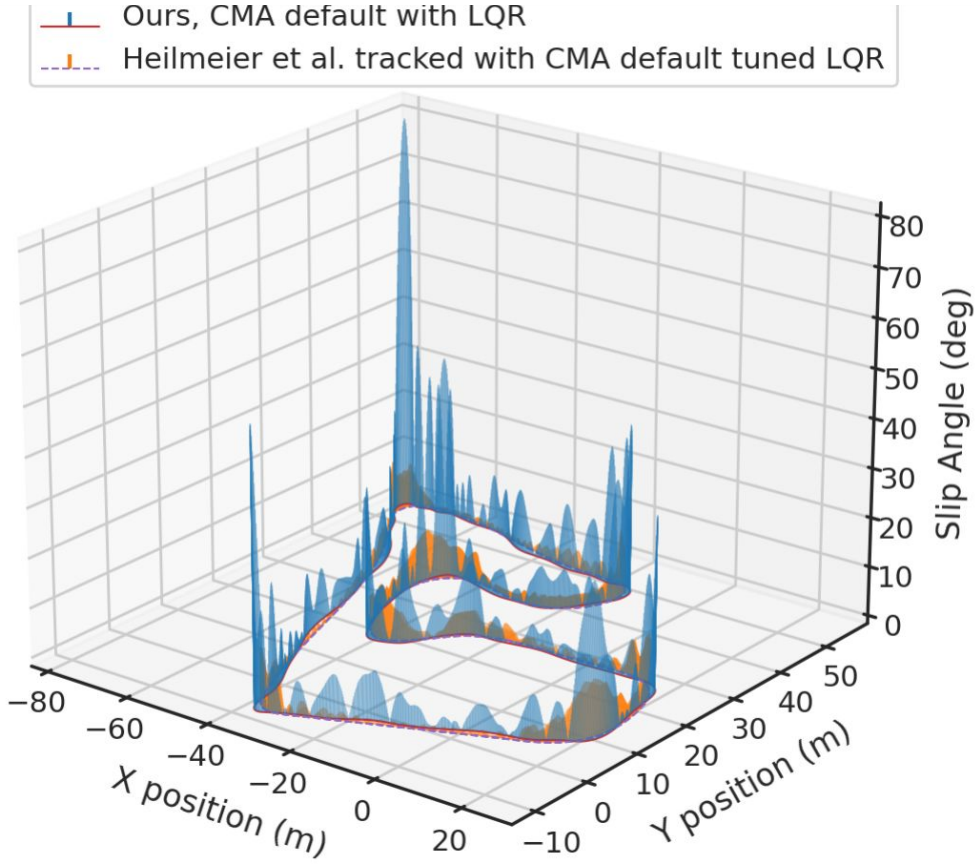
CMA-ES: Fitness function

- A fitness function evaluates the performance of a candidate in the generation
- In our racing context (single car time trials), the lap time is a great candidate for the fitness function
- A more performant candidate will have a lower lap time
- Lap times can be obtained by running simulations

Optimized Trajectories



Exploiting Simulations

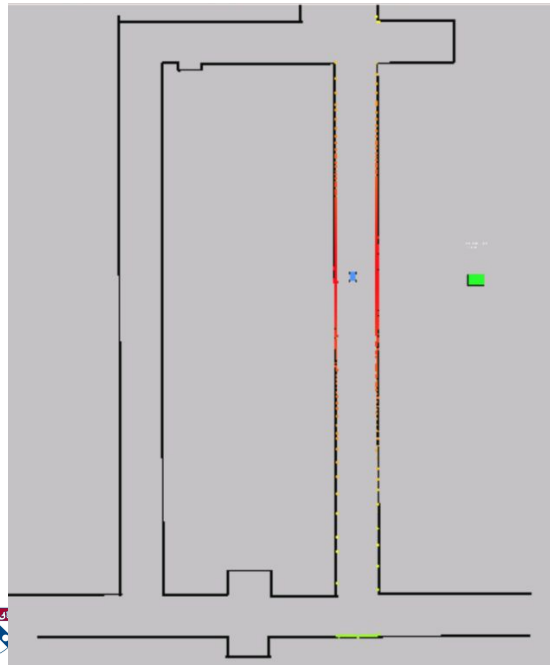


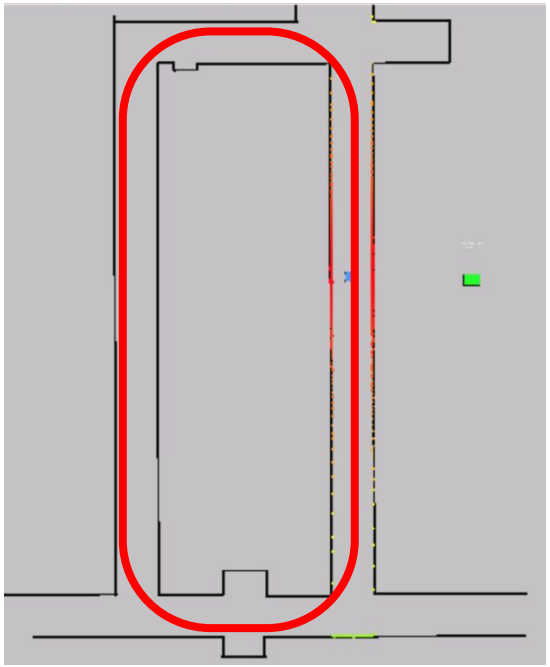
Penn Engineering

Example:
Levine Hall

Example: Global Trajectory Planning Software

- Global trajectory planner: https://github.com/TUMFTM/global_racetrajectory_optimization





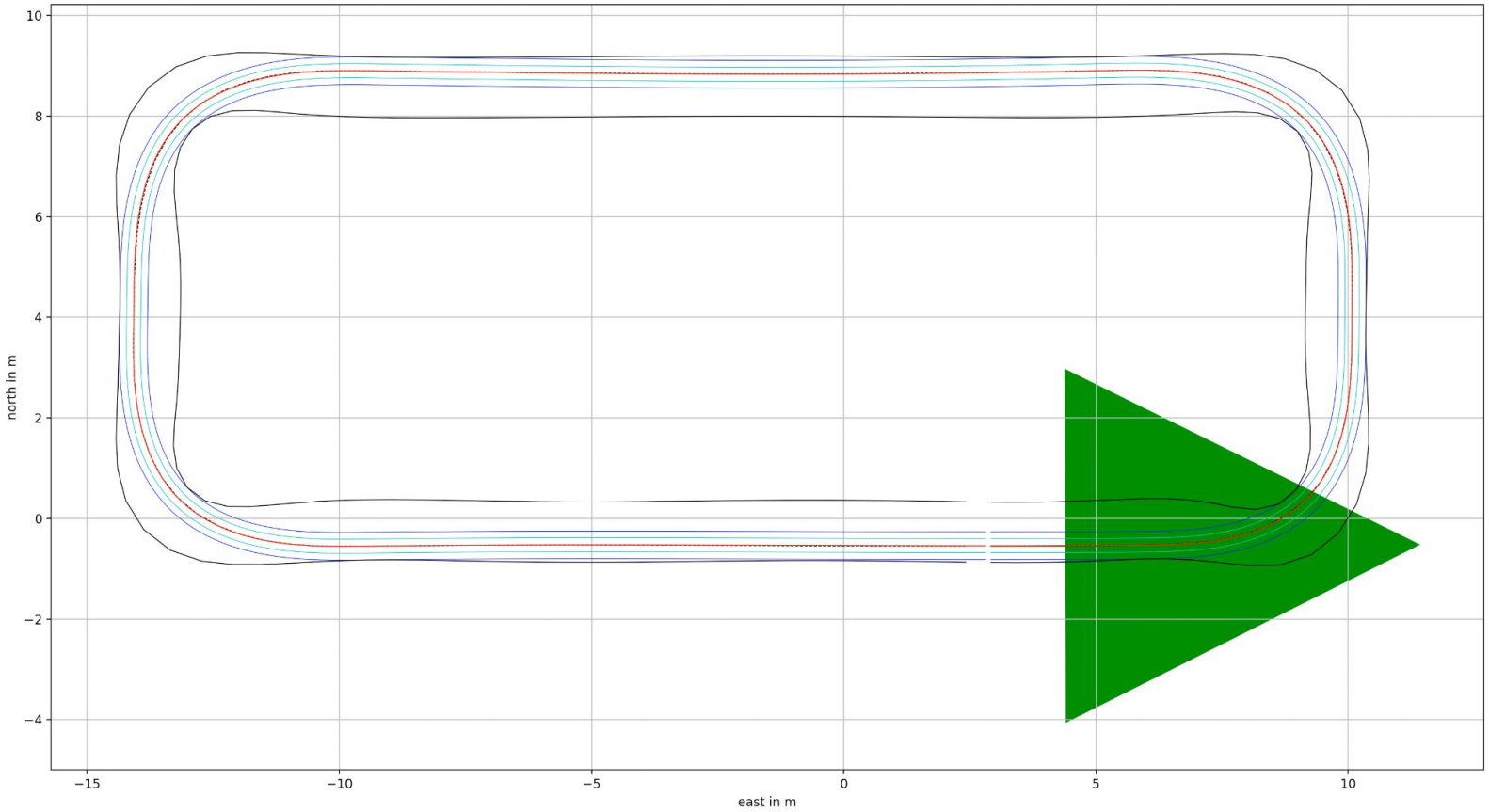
Levine Hall Optimization

	Shortest Path Optimization	Minimum Curvature Optimization	Minimum Time Optimization
Calculation time in seconds	1s	3s	9s
Laptime Output	**10.1s**	**8.94s**	**9.61s**

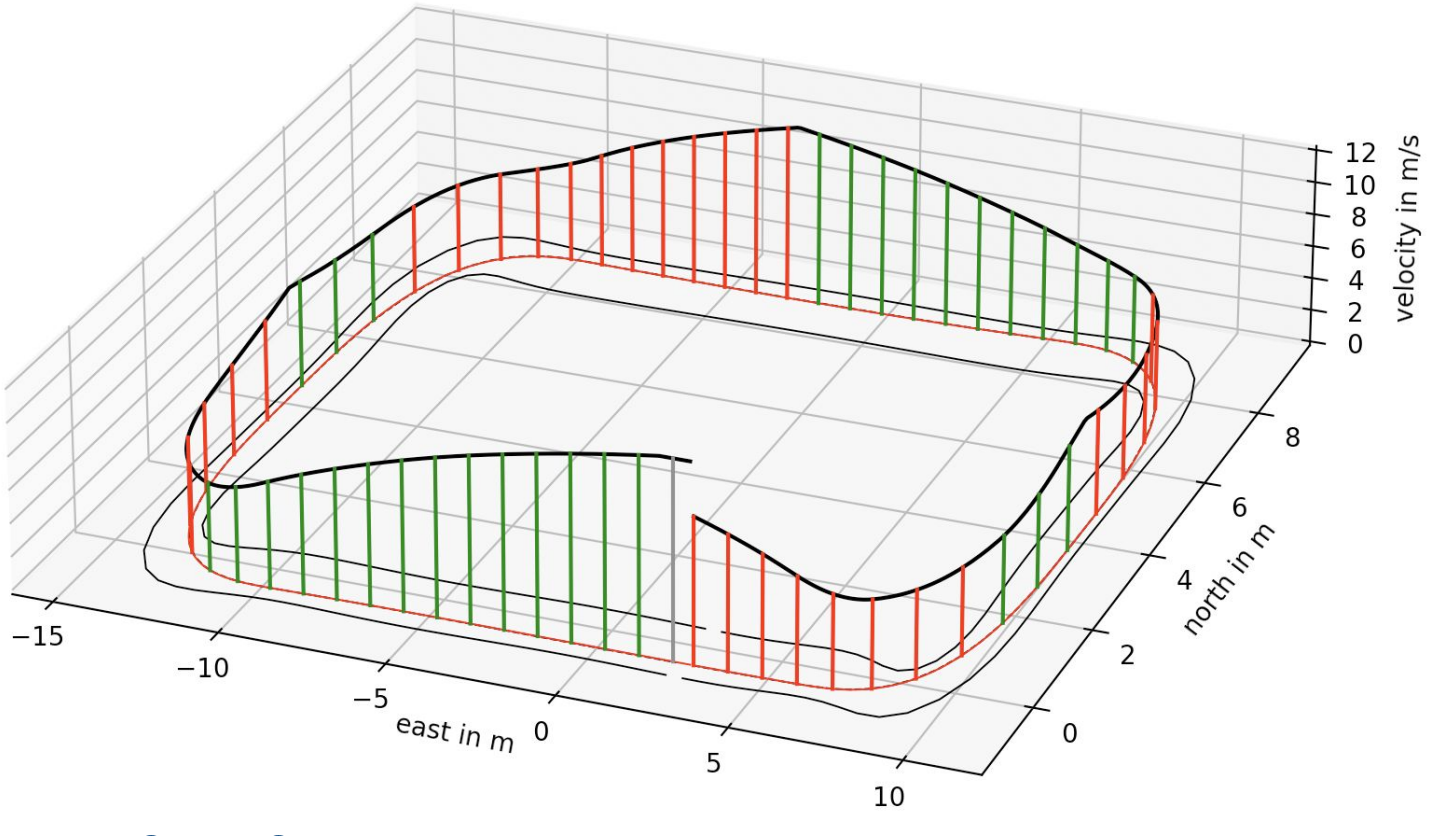
Optimization Parameters:

- Levine Track
- Discretization step size: 0.5m
- Vmax = 12m/s
- a_long_max = 12m/s2
- a_lat_max = 12m/s2

Levine Hall Optimization: Path



Levine Hall Optimization: Velocity



Read the Paper

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,

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:

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

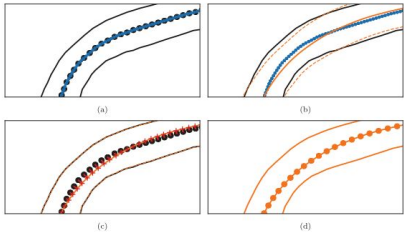


Figure 6. Four steps of reference line preprocessing (top left to bottom right). (a) Linear interpolation (b) Spline approximation (c) Correction of track widths (d) Spline interpolation

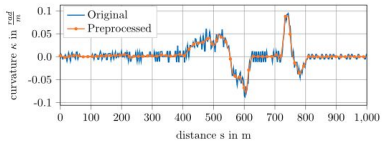


Figure 7. Curvature profiles of the original and the preprocessed reference lines (first 1000 m of the Berlin Formula E track).

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

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

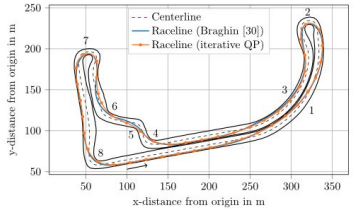


Figure 13. Comparison of the racelines for the Upper Heyford test track between the original formulation in [30] and the iterative QP formulation.

https://www.researchgate.net/publication/333869327_Minimum_curvature_trajectory_planning_and_control_for_an_autonomous_race_car

Penn Engineering

Questions?

Hidden extra slides

Raceline Optimization: Minimum Time

There are also other methods to get an optimal trajectory on a race track. Unlike the slide before, the problem can also be solved in a single optimization problem.

Solving **one** single problem: simultaneous Path + Velocity Profile Planning

- Minimum time trajectory planning **(Optimal Control)** (1)

For more information about the method please refer to the attached paper.

Advantages of minimum time optimal control problems:

- Objective equals the goal that is actually pursued on the race track
- Depending on the model, considerably more boundary conditions can be considered, e.g., energy limitations