Optimization-Based Motion Planning and Control for Autonomous Driving

Rudolf Reiter

Systems Control and Optimization Laboratory, University of Freiburg

IfA Coffee Talk
ETH Zürich
June 7, 2024

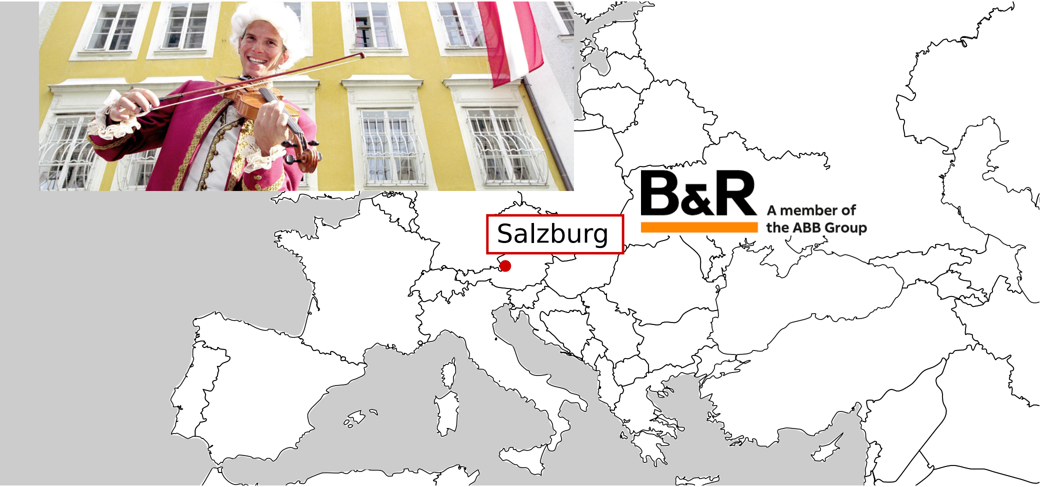
ufr ELO-X ARG autonomous racing·graz virtual vehicle

Outline

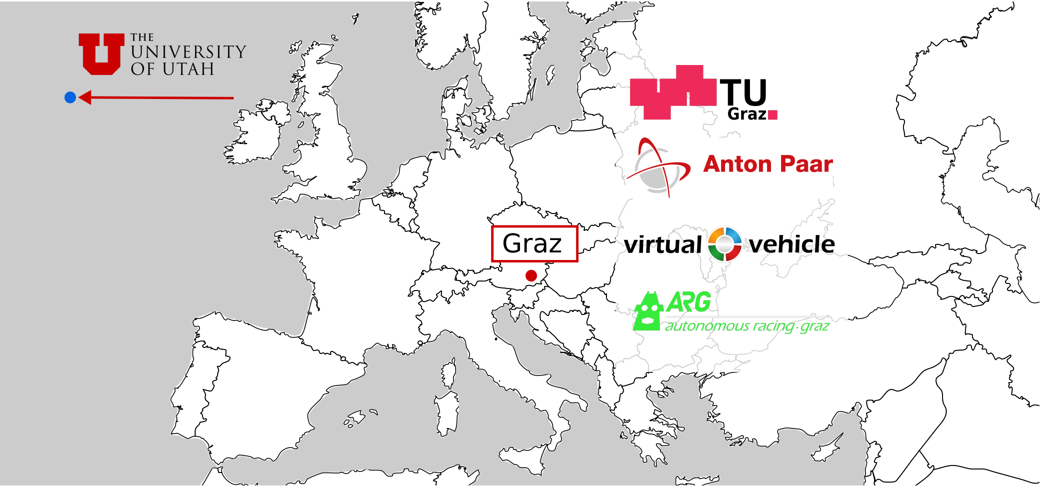
1. Personal introduction
2. Vehicle model in the Frenet coordinate frame
3. Challenges with obstacle avoidance
   - 3.1 obstacle shape
   - 3.2 non homeomorphic planning space
   - 3.3 obstacle prediction and interaction

Personal Introduction

Salzburg, Austria: until 2009

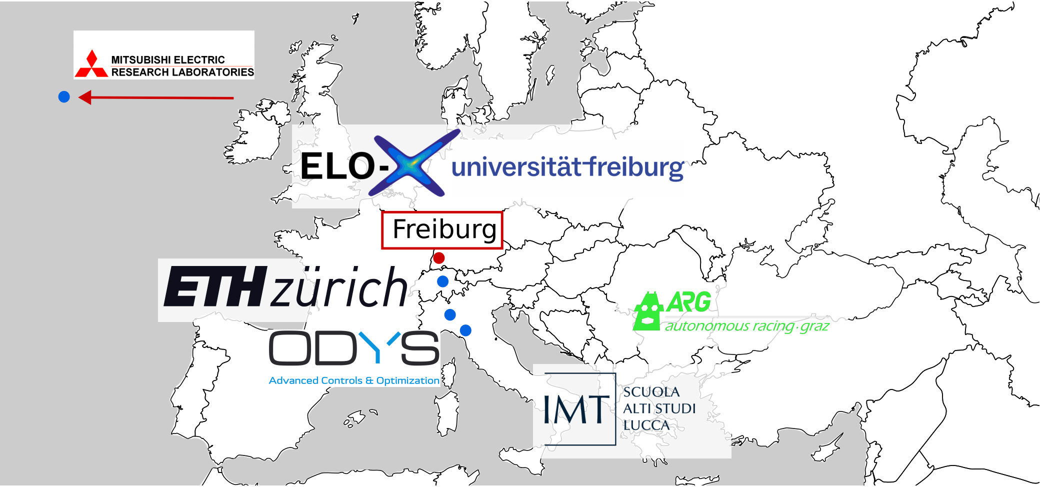


Graz, Austria: until 2021



Personal Introduction

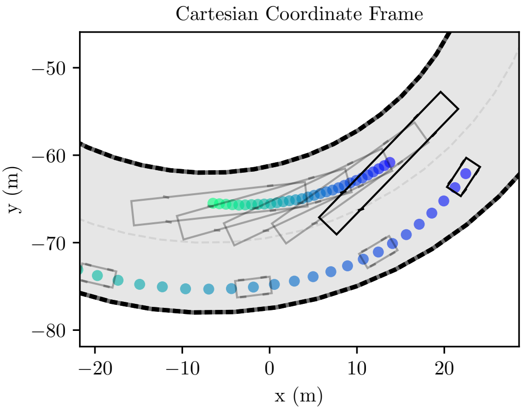
Freiburg, Germany: until ∼2024

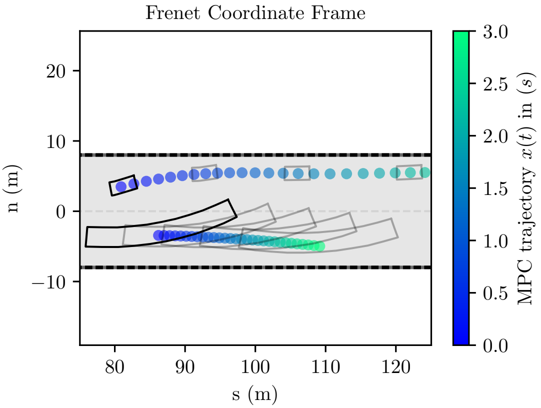


Vehicle Model

Modeling in two coordinate frames

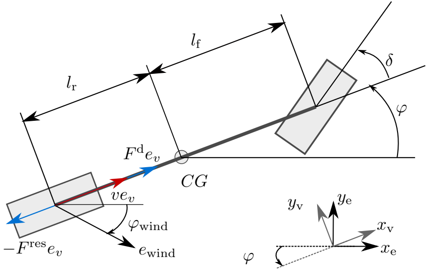
Comparison





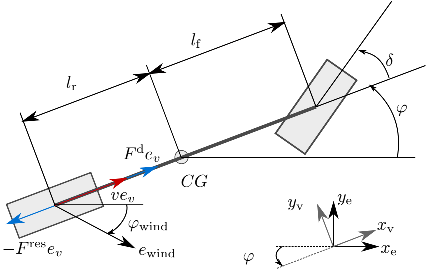
Single-track model in Cartesian coordinate frame (CCF)

- Cartesian states $x^{\mathrm{c,C}} = [p_x, p_y, \varphi]^\top \in \mathbb{R}^3$



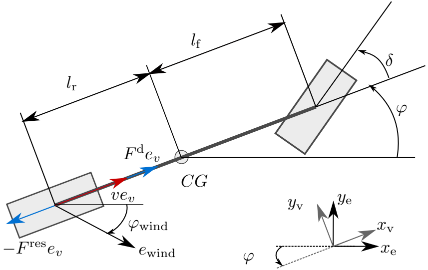
Single-track model in Cartesian coordinate frame (CCF)

- Cartesian states $x^{\mathrm{c,C}} = [p_x, p_y, \varphi]^\top \in \mathbb{R}^3$
- Some states are CF independent:
  $x^{\neg \mathrm{c}} = [v, \delta]^\top \in \mathbb{R}^2$



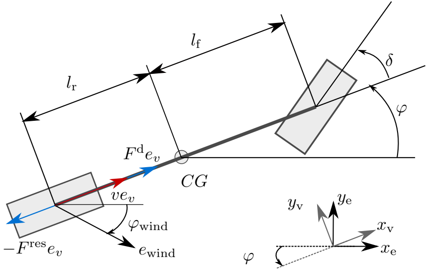
Single-track model in Cartesian coordinate frame (CCF)

- Cartesian states $x^{\mathrm{c,C}} = [p_x, p_y, \varphi]^\top \in \mathbb{R}^3$
- Some states are CF independent: $x^{\neg\mathrm{c}} = [v, \delta]^\top \in \mathbb{R}^2$
- Full state vector: $x^{\mathrm{C}} = [x^{\mathrm{c,C}\top} \quad x^{\neg\mathrm{c}\top}]^\top$



Single-track model in Cartesian coordinate frame (CCF)

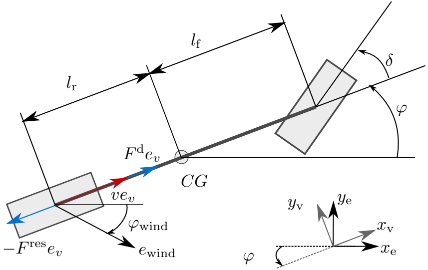
- Cartesian states $x^{\mathrm{c,C}} = [p_x, p_y, \varphi]^\top \in \mathbb{R}^3$
- Some states are CF independent: $x^{\neg\mathrm{c}} = [v, \delta]^\top \in \mathbb{R}^2$
- Full state vector: $x^{\mathrm{C}} = [x^{\mathrm{c,C}\top} \quad x^{\neg\mathrm{c}\top}]^\top$
- Inputs CF independent: $u = [F^{\mathrm{d}} \quad r]^\top \in \mathbb{R}^2$



Single-track model in Cartesian coordinate frame (CCF)

- Cartesian states $x^{\mathrm{c,C}} = [p_x, p_y, \varphi]^\top \in \mathbb{R}^3$
- Some states are CF independent:
  $x^{\neg\mathrm{c}} = [v, \delta]^\top \in \mathbb{R}^2$
- Full state vector: $x^{\mathrm{C}} = [x^{\mathrm{c,C}\top} \quad x^{\neg\mathrm{c}\top}]^\top$
- Inputs CF independent: $u = [F^{\mathrm{d}} \quad r]^\top \in \mathbb{R}^2$
- Dynamics of CCF dependent states

$$\dot{x}^{\mathrm{c,C}} = f^{\mathrm{c,C}}(x^{\mathrm{C}}, u) = \begin{bmatrix} v\cos(\varphi) \\ v\sin(\varphi) \\ \frac{v}{l}\tan(\delta) \end{bmatrix}$$



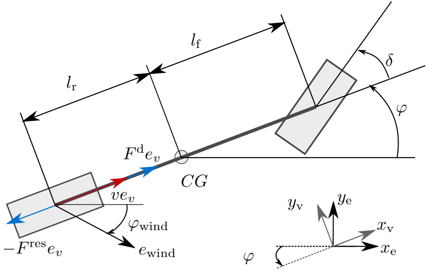
Single-track model in Cartesian coordinate frame (CCF)

- Cartesian states $x^{\mathrm{c,C}} = [p_x, p_y, \varphi]^\top \in \mathbb{R}^3$
- Some states are CF independent: $x^{\neg\mathrm{c}} = [v, \delta]^\top \in \mathbb{R}^2$
- Full state vector: $x^{\mathrm{C}} = [x^{\mathrm{c,C}\top} \quad x^{\neg\mathrm{c}\top}]^\top$
- Inputs CF independent: $u = [F^{\mathrm{d}} \quad r]^\top \in \mathbb{R}^2$
- Dynamics of CCF dependent states

$$\dot{x}^{\mathrm{c,C}} = f^{\mathrm{c,C}}(x^{\mathrm{C}}, u) = \begin{bmatrix} v\cos(\varphi) \\ v\sin(\varphi) \\ \frac{v}{l}\tan(\delta) \end{bmatrix}$$

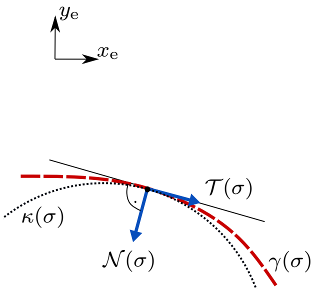
- Dynamics of CCF independent states

$$\dot{x}^{\neg\mathrm{c}} = f^{\neg\mathrm{c}}(x^{\neg\mathrm{c}}, u, \varphi) = \begin{bmatrix} \frac{1}{m}(F^{\mathrm{d}} - F^{\mathrm{wind}}(v,\varphi) - F^{\mathrm{roll}}(v)) \\ r \end{bmatrix}$$



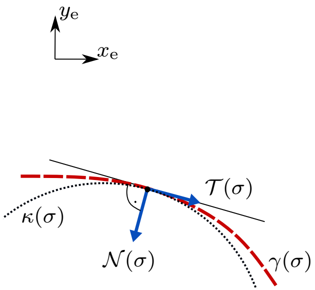
Frenet coordinate frame (FCF)

- Let $\gamma(t)$ be a cont. smooth diff. curve in $\mathbb{R}^2$



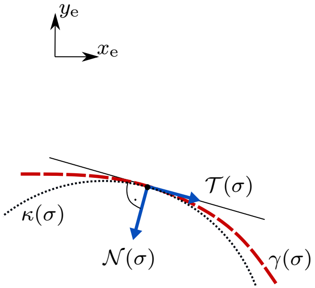
Frenet coordinate frame (FCF)

- Let $\gamma(t)$ be a cont. smooth diff. curve in $\mathbb{R}^2$
- Arc length: $\sigma(t) = \int_0^t \|\dot{\gamma}(\tau)\|_2 \, d\tau$



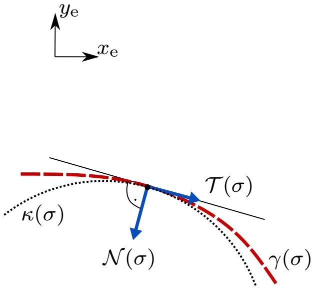
Frenet coordinate frame (FCF)

- Let $\gamma(t)$ be a cont. smooth diff. curve in $\mathbb{R}^2$
- Arc length: $\sigma(t) = \int_0^t \|\dot{\gamma}(\tau)\|_2 \, d\tau$
- Can parameterize curve by arc length $\rightarrow \gamma(\sigma)$



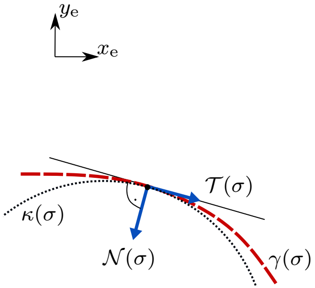
Frenet coordinate frame (FCF)

- Let $\gamma(t)$ be a cont. smooth diff. curve in $\mathbb{R}^2$
- Arc length: $\sigma(t) = \int_0^t \|\dot{\gamma}(\tau)\|_2 \, d\tau$
- Can parameterize curve by arc length $\rightarrow \gamma(\sigma)$
- Tangent unit vector $\mathcal{T}(\sigma) := \gamma'(\sigma)$,
  Curvature vector $\mathcal{T}'(\sigma)$,
  Curvature $\kappa(\sigma) = \|\mathcal{T}'(\sigma)\| = (\varphi^{\gamma}(\sigma))'$
  Normal unit vector $\mathcal{N}(\sigma) := \frac{\mathcal{T}'(\sigma)}{\|\mathcal{T}'(\sigma)\|}$



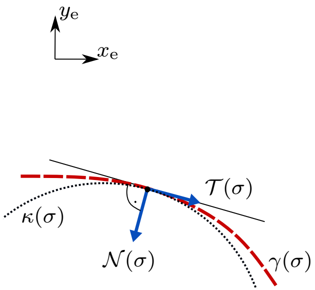
Frenet coordinate frame (FCF)

- Let $\gamma(t)$ be a cont. smooth diff. curve in $\mathbb{R}^2$
- Arc length: $\sigma(t) = \int_0^t \|\dot{\gamma}(\tau)\|_2 \, d\tau$
- Can parameterize curve by arc length $\rightarrow \gamma(\sigma)$
- Tangent unit vector $\mathcal{T}(\sigma) := \gamma'(\sigma)$,
  Curvature vector $\mathcal{T}'(\sigma)$,
  Curvature $\kappa(\sigma) = \|\mathcal{T}'(\sigma)\| = (\varphi^{\gamma}(\sigma))'$
  Normal unit vector $\mathcal{N}(\sigma) := \frac{\mathcal{T}'(\sigma)}{\|\mathcal{T}'(\sigma)\|}$
- Frenet-Serret frame (2D): $\mathcal{T}(\sigma), \mathcal{N}(\sigma)$



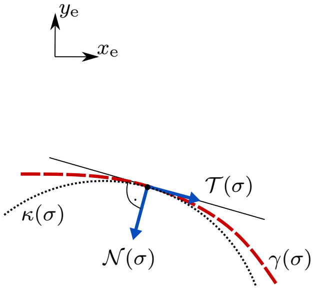
Frenet coordinate frame (FCF)

- Let $\gamma(t)$ be a cont. smooth diff. curve in $\mathbb{R}^2$
- Arc length: $\sigma(t) = \int_0^t \|\dot{\gamma}(\tau)\|_2 \, d\tau$
- Can parameterize curve by arc length $\rightarrow \gamma(\sigma)$
- Tangent unit vector $\mathcal{T}(\sigma) := \gamma'(\sigma)$,
  Curvature vector $\mathcal{T}'(\sigma)$,
  Curvature $\kappa(\sigma) = \|\mathcal{T}'(\sigma)\| = (\varphi^{\gamma}(\sigma))'$
  Normal unit vector $\mathcal{N}(\sigma) := \frac{\mathcal{T}'(\sigma)}{\|\mathcal{T}'(\sigma)\|}$
- Frenet-Serret frame (2D): $\mathcal{T}(\sigma), \mathcal{N}(\sigma)$
- Frenet-Serret frame yields an orthonormal basis along the reference curve



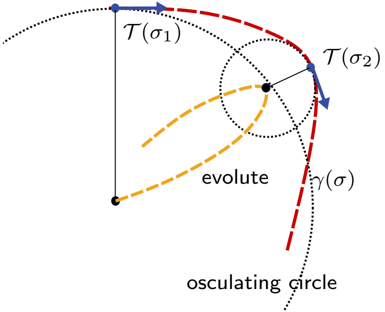
Frenet coordinate frame (FCF)

- Let $\gamma(t)$ be a cont. smooth diff. curve in $\mathbb{R}^2$
- Arc length: $\sigma(t) = \int_0^t \|\dot{\gamma}(\tau)\|_2 \, d\tau$
- Can parameterize curve by arc length $\to \gamma(\sigma)$
- Tangent unit vector $\mathcal{T}(\sigma) := \gamma'(\sigma)$,
  Curvature vector $\mathcal{T}'(\sigma)$,
  Curvature $\kappa(\sigma) = \|\mathcal{T}'(\sigma)\| = (\varphi^{\gamma}(\sigma))'$
  Normal unit vector $\mathcal{N}(\sigma) := \frac{\mathcal{T}'(\sigma)}{\|\mathcal{T}'(\sigma)\|}$
- Frenet-Serret frame (2D): $\mathcal{T}(\sigma), \mathcal{N}(\sigma)$
- Frenet-Serret frame yields an orthonormal basis along the reference curve
- Curvature defines the curve uniquely up to rigid motion



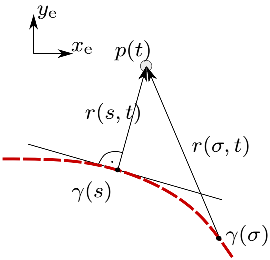
Frenet coordinate frame: Osculating circle and evolute

- The osculating circle is the circle that has the same tangent vector $\mathcal{T}(\sigma)$ and curvature $\kappa(\sigma)$ at the point $\gamma(\sigma)$
- The curve that connects the center points of all osculating circles is the evolute



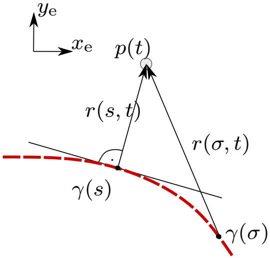
Motion in the Frenet coordinate frame

- Consider the motion of point $p(t)$



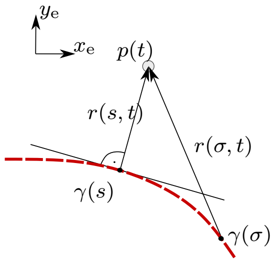
Motion in the Frenet coordinate frame

- Consider the motion of point $p(t)$
- The distance of point $p(t)$ to the curve $\gamma(\sigma)$ is $r(\sigma,t) = p(t) - \gamma(\sigma)$



Motion in the Frenet coordinate frame

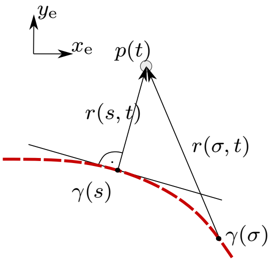
- Consider the motion of point $p(t)$
- The distance of point $p(t)$ to the curve $\gamma(\sigma)$ is $r(\sigma, t) = p(t) - \gamma(\sigma)$
- The closest point is $s(t) = \arg\min_{\sigma} \frac{1}{2} \left\| r(\sigma, t) \right\|_2^2$



Motion in the Frenet coordinate frame

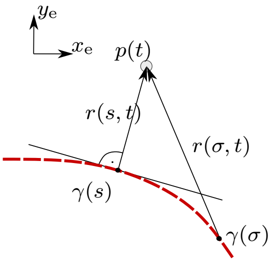
- Consider the motion of point $p(t)$
- The distance of point $p(t)$ to the curve $\gamma(\sigma)$ is $r(\sigma, t) = p(t) - \gamma(\sigma)$
- The closest point is $s(t) = \arg\min_\sigma \frac{1}{2} \|r(\sigma, t)\|_2^2$
- FONC:

$$0 = \frac{d}{d\sigma}\left(\frac{1}{2}\|r(\sigma, t)\|_2^2\right) = r(s, t)^\top \mathcal{T}(s)$$



Motion in the Frenet coordinate frame

- Consider the motion of point $p(t)$
- The distance of point $p(t)$ to the curve $\gamma(\sigma)$ is $r(\sigma,t) = p(t) - \gamma(\sigma)$
- The closest point is $s(t) = \arg\min_\sigma \frac{1}{2}\left\|r(\sigma,t)\right\|_2^2$
- FONC:
$$0 = \frac{d}{d\sigma}\left(\frac{1}{2}\left\|r(\sigma,t)\right\|_2^2\right) = r(s,t)^\top \mathcal{T}(s)$$
- Assume, that we know $s(t=0)$ is optimal, we enforce optimality along the trajectory, by
$$0 = \frac{d}{dt} r(s,t)^\top \mathcal{T}(s),$$



Motion in the Frenet coordinate frame

- Consider the motion of point $p(t)$
- The distance of point $p(t)$ to the curve $\gamma(\sigma)$ is $r(\sigma, t) = p(t) - \gamma(\sigma)$
- The closest point is $s(t) = \arg\min_\sigma \frac{1}{2} \left\| r(\sigma, t) \right\|_2^2$
- FONC:

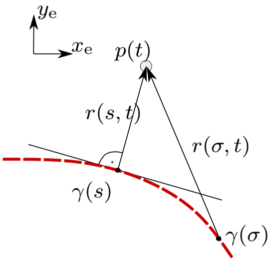
$$0 = \frac{d}{d\sigma}\left(\frac{1}{2} \left\| r(\sigma, t) \right\|_2^2\right) = r(s, t)^\top \mathcal{T}(s)$$

- Assume, that we know $s(t = 0)$ is optimal, we enforce optimality along the trajectory, by

$$0 = \frac{d}{dt} r(s, t)^\top \mathcal{T}(s),$$

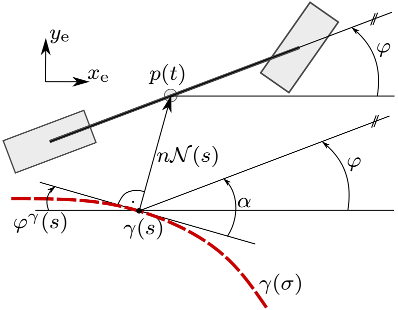
- from which it follows

$$\dot{s}(t) = \frac{\left(\frac{dp(t)}{dt}\right)^\top \mathcal{T}(s)}{1 - \kappa(s) r(s, t)^\top \mathcal{N}(s)}$$



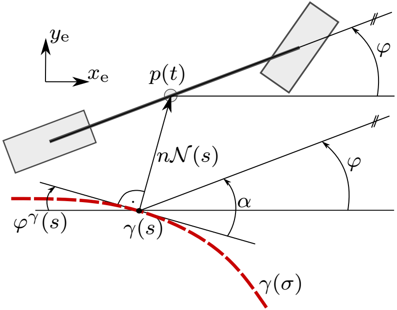
Vehicle model in the Frenet coordinate frame

- Recall vehicle model. Identify $p(t)$ as the Cartesian vehicle position



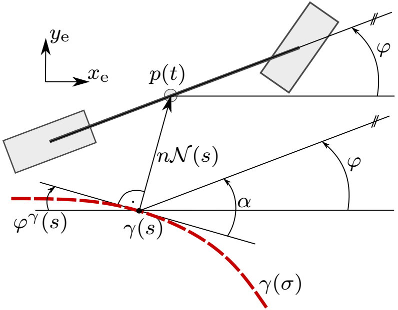
Vehicle model in the Frenet coordinate frame

- Recall vehicle model. Identify $p(t)$ as the Cartesian vehicle position
- Call $s(t)$ longitudinal position state



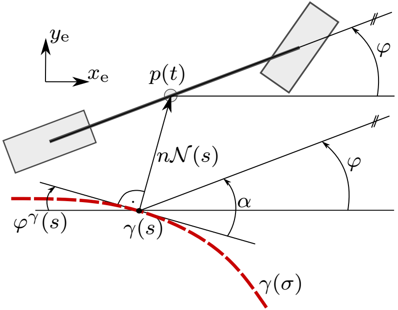
Vehicle model in the Frenet coordinate frame

- Recall vehicle model. Identify $p(t)$ as the Cartesian vehicle position
- Call $s(t)$ longitudinal position state
- Call $\alpha(t) := \varphi(t) - \varphi^{\gamma}(s(t))$ heading angle mismatch state



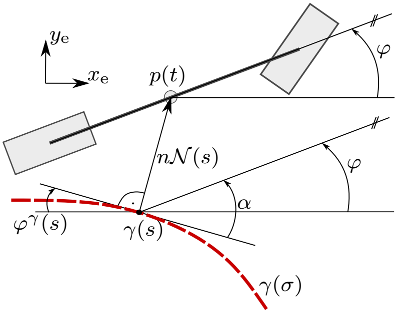
Vehicle model in the Frenet coordinate frame

- Recall vehicle model. Identify $p(t)$ as the Cartesian vehicle position
- Call $s(t)$ longitudinal position state
- Call $\alpha(t) := \varphi(t) - \varphi^{\gamma}(s(t))$ heading angle mismatch state
- Identify $v(t) = \left(\frac{dp(t)}{dt}\right)$ as the vehicle velocity



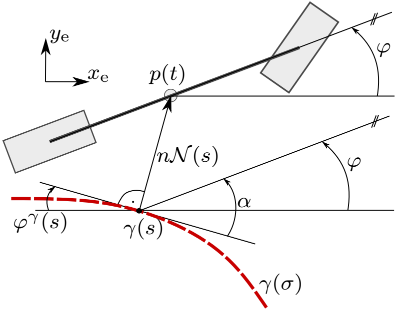
Vehicle model in the Frenet coordinate frame

- Recall vehicle model. Identify $p(t)$ as the Cartesian vehicle position
- Call $s(t)$ longitudinal position state
- Call $\alpha(t) := \varphi(t) - \varphi^{\gamma}(s(t))$ heading angle mismatch state
- Identify $v(t) = \left(\frac{dp(t)}{dt}\right)$ as the vehicle velocity
- Call $n(t) := r(s(t), t)^{\top} \mathcal{N}(s(t))$ lateral position state



Vehicle model in the Frenet coordinate frame

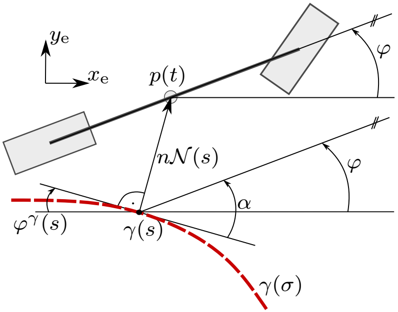
- Recall vehicle model. Identify $p(t)$ as the Cartesian vehicle position
- Call $s(t)$ longitudinal position state
- Call $\alpha(t) := \varphi(t) - \varphi^{\gamma}(s(t))$ heading angle mismatch state
- Identify $v(t) = \left(\frac{dp(t)}{dt}\right)$ as the vehicle velocity
- Call $n(t) := r(s(t),t)^{\top}\mathcal{N}(s(t))$ lateral position state
- Frenet states $x^{\mathrm{c,F}} = \mathcal{F}_{\gamma}(x^{\mathrm{c,C}}) = [s,n,\alpha]^{\top} \in \mathbb{R}^3$



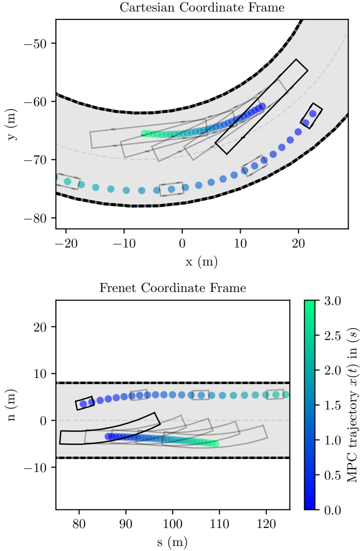
Vehicle model in the Frenet coordinate frame

- Recall vehicle model. Identify $p(t)$ as the Cartesian vehicle position
- Call $s(t)$ longitudinal position state
- Call $\alpha(t) := \varphi(t) - \varphi^\gamma(s(t))$ heading angle mismatch state
- Identify $v(t) = \left(\frac{dp(t)}{dt}\right)$ as the vehicle velocity
- Call $n(t) := r(s(t),t)^\top \mathcal{N}(s(t))$ lateral position state
- Frenet states $x^{\mathrm{c,F}} = \mathcal{F}_\gamma(x^{\mathrm{c,C}}) = [s,n,\alpha]^\top \in \mathbb{R}^3$
- Dynamics of FCF dependent states

$$\dot{x}^{\mathrm{c,F}} = f^{\mathrm{c,F}}(x^{\mathrm{F}},u) = \begin{bmatrix} \frac{v\cos(\alpha)}{1-n\kappa(s)} \\ v\sin(\alpha) \\ \frac{v}{l}\tan(\delta) - \frac{\kappa(s)v\cos(\alpha)}{1-n\kappa(s)} \end{bmatrix}$$

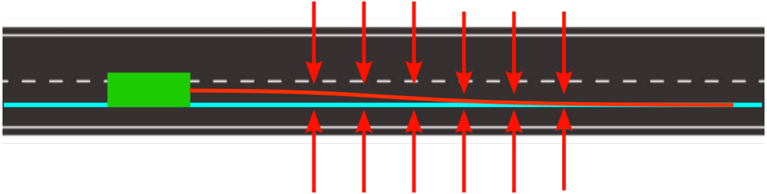


Feature	CCF	FCF
reference definition	✗	✓
boundary constraints	✗	✓
obstacle specification	✓	✗
disturbance specification	✓	✗



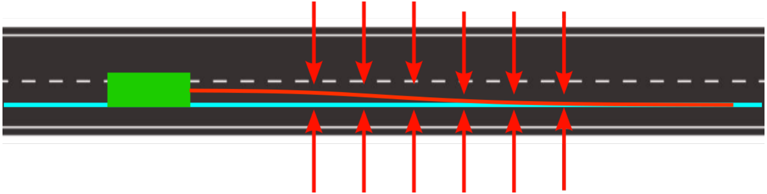
Frenet Coordinate Frame: Reference curve

- Transformation along a reference curve $\gamma(\sigma)$
- How to choose this curve?
  - Tracking of a center line

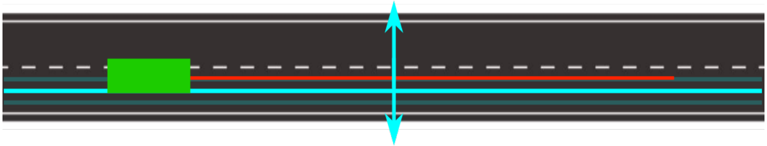


Frenet Coordinate Frame: Reference curve

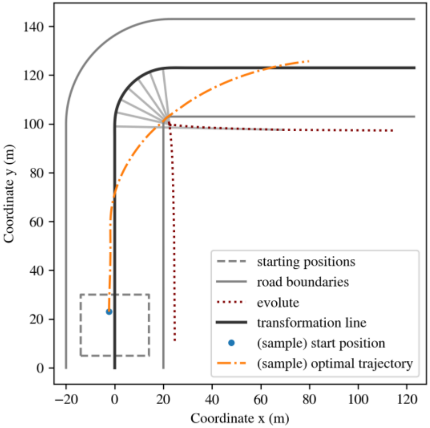
- Transformation along a reference curve $\gamma(\sigma)$
- How to choose this curve?
  - Tracking of a center line



  - Racing: free to choose



Optimizing the Reference

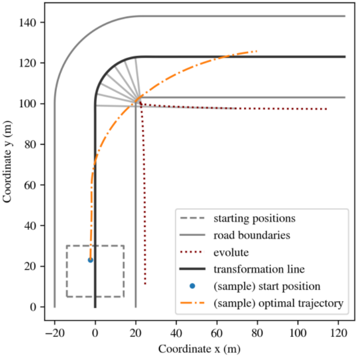


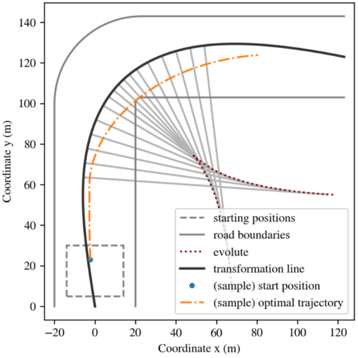
- The transformation has one big issue!
- Singular subspace at points $[s, n]^\top$, with $1 - n\kappa(s) = 0$

- Usually no problem, since curvature is small $n\kappa(s) \ll 1$
- Can even use the free choice of the reference to our advantage

Frenet Coordinate Frame: Reference curve

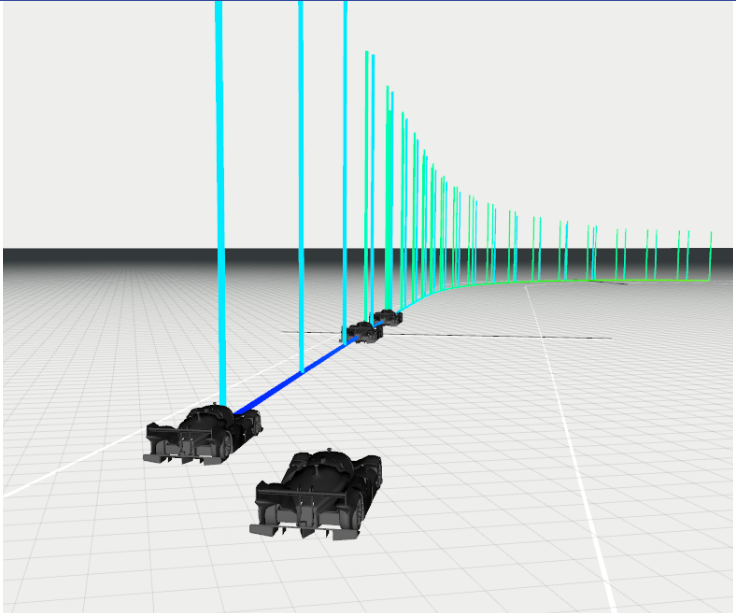
Solving a priori an optimization problem to obtain $\gamma(\sigma)$ that pushes the evolute outside and increases other favorable numercial properties for NMPC.[1]





[1] Rudolf Reiter and Moritz Diehl. "Parameterization Approach of the Frenet Transformation for Model Predictive Control of Autonomous Vehicles". In: *2021 European Control Conference (ECC)*. 2021, pp. 2414–2419. DOI: 10.23919/ECC54610.2021.9655053.

Challenges with obstacle avoidance



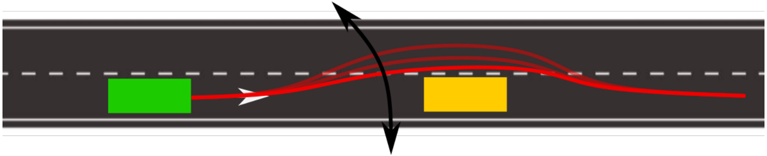
- Obstacle shape

Challenges with obstacle avoidance

Problem Statement

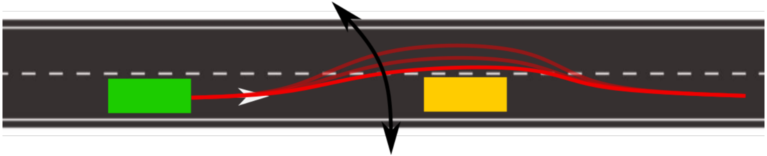
- Obstacle shape

- Nonconvex non-homeomorphic planning space

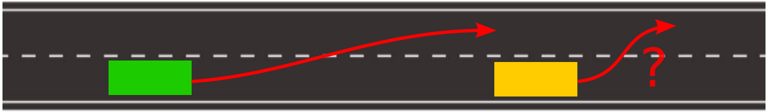


- Obstacle shape

- Nonconvex non-homeomorphic planning space



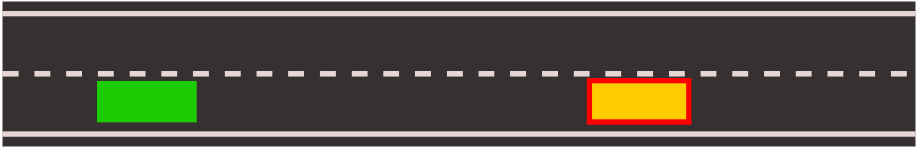
- Interaction and prediction



Challenge 1: Obstacle shape

Collision avoidance on a straight road in Cartesian coordinates

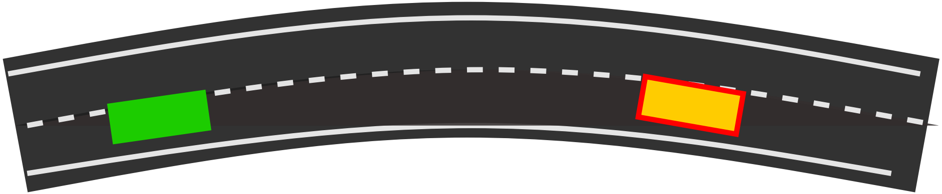
- convex obstacle shape ✓
- state independent shape ✓



Challenge 1: Obstacle shape

Collision avoidance on a curvy road in Cartesian coordinates

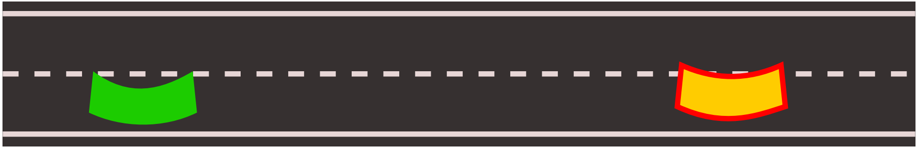
- convex obstacle shape ✓
- state independent shape ✓



Challenge 1: Obstacle shape

Collision avoidance on a curvy road in Frenet coordinates

- nonconvex obstacle shape ✗
- state dependent shape ✗



Combining states of Cartesian and Frenet frame

Goal:

- Reference definition, boundary constraints $\rightarrow$ Frenet Coordinate Frame (FCF)
- Obstacle specification, Cartesian disturbance (e.g., wind force) $\rightarrow$ Cartesian Coordinate Frame (CCF)

Ways to combine both models

Conventional

Possible formulations of NMPC:

- Use only one CF, approximate and simplify non-smooth constraints
  - Dynamics model in CCF: approximate $\mathcal{F}_\gamma$ with artificial path state (MPCC) *(Not reviewed here)*
  - Dynamics model in FCF: convex over-approximate obstacles $\rightarrow$ conventional
- Model dynamics in *one* CF, use $\mathcal{F}_\gamma$ or $\mathcal{F}_\gamma{}^{-1}$ to obtain *other* states
- Model dynamics redundantly in *both* CFs

Possible formulations of NMPC:

- Use only one CF, approximate and simplify non-smooth constraints
- Model dynamics in *one* CF, use $\mathcal{F}_\gamma$ or ${\mathcal{F}_\gamma}^{-1}$ to obtain *other* states
  - Dynamics model in CCF ✗: $\mathcal{F}_\gamma$ is an nonlinear optimization problem by itself
  - Dynamics model in FCF ✓: ${\mathcal{F}_\gamma}^{-1}$ can be obtained efficiently → direct elimination[2]
- Model dynamics redundantly in *both* CFs

[2] Rudolf Reiter et al. "Frenet-Cartesian model representations for automotive obstacle avoidance within nonlinear MPC". In: *European Journal of Control* (2023), p. 100847. ISSN: 0947-3580.

Ways to combine both models

Lifting

Possible formulations of NMPC:

- Use only one CF, approximate and simplify non-smooth constraints
- Model dynamics in *one* CF, use $\mathcal{F}_\gamma$ or ${\mathcal{F}_\gamma}^{-1}$ to obtain *other* states
- Model dynamics redundantly in *both* CFs
  - Lifting to higher dimension
  - Number of states n_x increases from 5 to 8 $\rightarrow$ lifting[3]

[3]Reiter et al., "Frenet-Cartesian model representations for automotive obstacle avoidance within nonlinear MPC".

NMPC Problem

Direct elimination

$x^{\mathrm{F}} \in \mathbb{R}^5 \ldots$ Frenet states, $x^{\mathrm{c,F}} \in \mathbb{R}^3 \ldots$ Frenet position states, $\mathcal{P} \ldots$ obstacle-free set
$\theta \ldots$ hyperplane variables, $\mathcal{F}_\gamma{}^{-1} \ldots$ inverse Frenet transformation, $\Phi^{\mathrm{F}}(\cdot) \ldots$ integrator

$$
\begin{aligned}
\min_{\substack{x_0^{\mathrm{F}},\ldots,x_N^{\mathrm{F}},\\ u_0,\ldots,u_{N-1}\\ \theta_1,\ldots,\theta_{n_{\mathrm{opp}}}}} \quad & \sum_{k=0}^{N-1} \|u_k\|_R^2 + \left\|x_k^{\mathrm{F}} - x_{\mathrm{ref},k}^{\mathrm{F}}\right\|_Q^2 + \left\|x_N^{\mathrm{F}} - x_{\mathrm{ref},N}^{\mathrm{F}}\right\|_{Q_N}^2 \\
\text{s.t.} \quad & x_0^{\mathrm{F}} = \hat{x}_0^{\mathrm{F}}, \\
& x_{i+1}^{\mathrm{F}} = \Phi^{\mathrm{F}}(x_i^{\mathrm{F}}, u_i, \Delta t), \qquad i = 0,\ldots,N-1, \\
& \underline{u} \leq u_i \leq \overline{u}, \qquad i = 0,\ldots,N-1, \\
& \underline{x}^{\mathrm{F}} \leq x_i^{\mathrm{F}} \leq \overline{x}^{\mathrm{F}}, \qquad i = 0,\ldots,N, \\
& \underline{x}^{\mathrm{c,C}} \leq \mathcal{F}_\gamma{}^{-1}(x^{\mathrm{c,F}}) \leq \overline{x}^{\mathrm{c,C}}, \qquad i = 0,\ldots,N, \\
& \underline{a}^{\mathrm{lat}} \leq a_{\mathrm{lat}}^{\mathrm{F}}(x_i) \leq \overline{a}^{\mathrm{lat}}, \qquad i = 0,\ldots,N, \\
& v_N \leq \overline{v}_N, \\
& \mathcal{F}_\gamma{}^{-1}(x^{\mathrm{c,F}}) \in \mathcal{P}(x_i^{\mathrm{c,opp,j}}, \theta_j), \qquad i = 0,\ldots,N-1, \\
& \qquad j = 1,\ldots,n_{\mathrm{opp}}.
\end{aligned}
$$

NMPC Problem
Lifted

$x^{\mathrm{F}} \in \mathbb{R}^5 \ldots$ Frenet states, $x^{\mathrm{d}} \in \mathbb{R}^8 \ldots$ lifted states, $\mathcal{P} \ldots$ obstacle-free set
$\theta \ldots$ hyperplane variables, $\Phi^{\mathrm{d}}(\cdot) \ldots$ model integration function

$$
\begin{aligned}
\min_{\substack{x_0^{\mathrm{d}},\ldots,x_N^{\mathrm{d}},\\ u_0,\ldots,u_{N-1}\\ \theta_1,\ldots,\theta_{n_{\mathrm{opp}}}}} \quad & \sum_{k=0}^{N-1} \|u_k\|_R^2 + \left\|x_k^{\mathrm{F}} - x_{\mathrm{ref},k}^{\mathrm{F}}\right\|_Q^2 + \left\|x_N^{\mathrm{F}} - x_{\mathrm{ref},N}^{\mathrm{F}}\right\|_{Q_N}^2 \\
\text{s.t.} \quad & x_0^{\mathrm{d}} = \hat{x}_0^{\mathrm{d}}, \\
& x_{i+1}^{\mathrm{d}} = \Phi^{\mathrm{d}}(x_i^{\mathrm{d}}, u_i, \Delta t), \quad i = 0, \ldots, N-1, \\
& \underline{u} \leq u_i \leq \overline{u}, \quad i = 0, \ldots, N-1, \\
& \underline{x}^{\mathrm{d}} \leq x_i^{\mathrm{d}} \leq \overline{x}^{\mathrm{d}}, \quad i = 0, \ldots, N, \\
& \underline{a}^{\mathrm{lat}} \leq a_{\mathrm{lat}}(x_i^{\mathrm{d}}) \leq \overline{a}^{\mathrm{lat}}, \quad i = 0, \ldots, N, \\
& v_N \leq \overline{v}_N, \\
& x_i^{\mathrm{c,C}} \in \mathcal{P}(x_i^{\mathrm{c,opp,j}}, \theta_j), \quad i = 0, \ldots, N-1, \\
& \qquad j = 1, \ldots, n_{\mathrm{opp}}.
\end{aligned}
$$

Evaluation

Setup:

- Simulation on randomized scenarios with three obstacles to overtake
- acados, 6s horizon length, 50 discr. points
- Obstacle formulations:
  - Ellipsoids
  - Covering circles (1,3,5,7)
  - Separating hyper-planes
- Coordinate formulations:
  - Conventional (over-approximation)
  - Direct elimination
  - Lifted ODE

Evaluation:

- Computation time
- Maximum progress

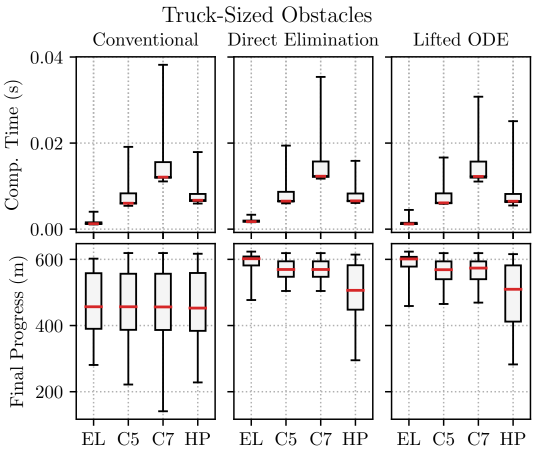


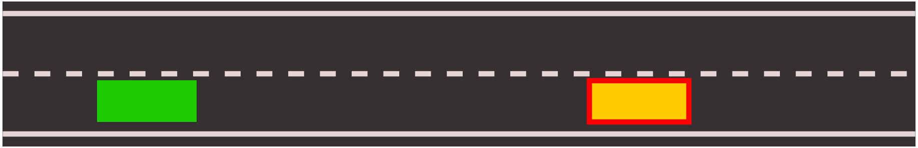
Figure: Box-plot comparison of the NMPC solution timings for each real-time iteration and the final progress after 20 seconds for different obstacle formulations for truck-sized vehicles.

Computation times (ms) for truck-sized obstacles					
	Conventional	Direct Elimination		Lifted ODE	
EL	1.5 ± 0.4	1.9 ± 0.2	28.9%	$\mathbf{1.4 \pm 0.3}$	$\mathbf{-6.6\%}$
C5	7.2 ± 1.9	7.6 ± 1.7	5.5%	7.2 ± 1.8	-0.0%
C7	14.0 ± 3.2	14.0 ± 2.8	-0.1%	13.9 ± 2.9	-0.4%
HP	7.5 ± 1.5	7.5 ± 1.5	-0.1%	7.4 ± 1.7	-1.6%
car-sized obstacles					
EL	1.5 ± 0.5	2.0 ± 0.4	29.6%	$\mathbf{1.4 \pm 0.4}$	$\mathbf{-5.7\%}$
C1	1.4 ± 0.4	1.9 ± 0.4	34.0%	1.4 ± 0.4	-3.5%
C3	3.6 ± 1.1	4.0 ± 1.0	12.4%	3.6 ± 1.1	0.6%
HP	8.0 ± 2.3	7.9 ± 1.9	-0.6%	7.7 ± 2.0	-4.0%

Table: Mean and standard deviation of computation times for different scenarios, obstacle formulations and lifting formulations. Additionally, the difference in percent to the conventional formulation is given.

Challenge 1: Obstacle shape

Collision avoidance constraint can be represented as ∞-norm



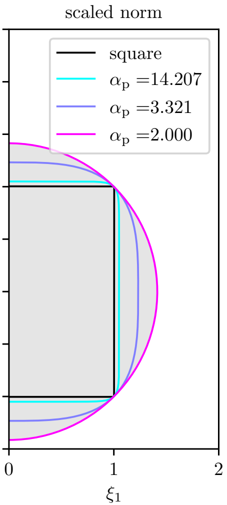
Challenge 1: Obstacle shape

∞-norm: Problem with linearization - we create distinct local minima

Challenge 1: Obstacle shape

2-norm: takes a major share of the free planning space

Challenge 1: Idea: p-Norm homotopy



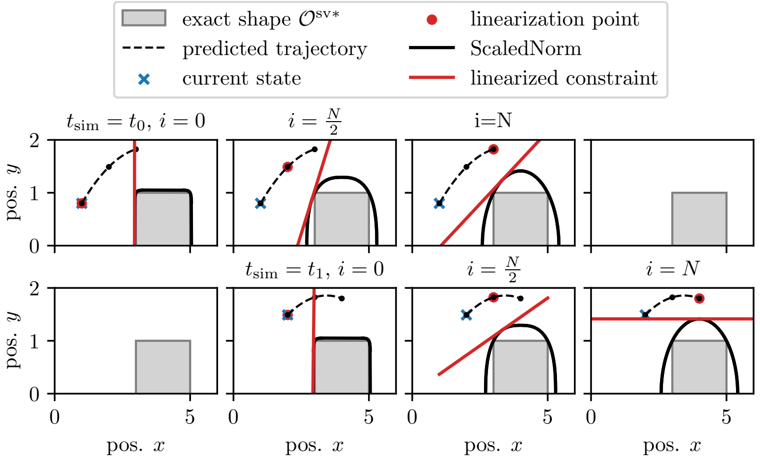
- Norm obstacle constraint in scaled coordinates ξ given by

$$1 \geq o^{\mathrm{p}}(\xi; \alpha_{\mathrm{p}}) = \left(\frac{1}{n} \sum_{i=1}^{n} |\xi|^{\alpha_{\mathrm{p}}} \right)^{\frac{1}{\alpha_{\mathrm{p}}}}$$

- Homotopies are often used to successively add nonlinearity
- Solve sequence of optimization problems and increase norm value α_{p} in each problem
- Problem: We want to use RTI, i.e., only solve one QP in each iteration!

Challenge 1: Idea: *Progressive smoothing*[4]

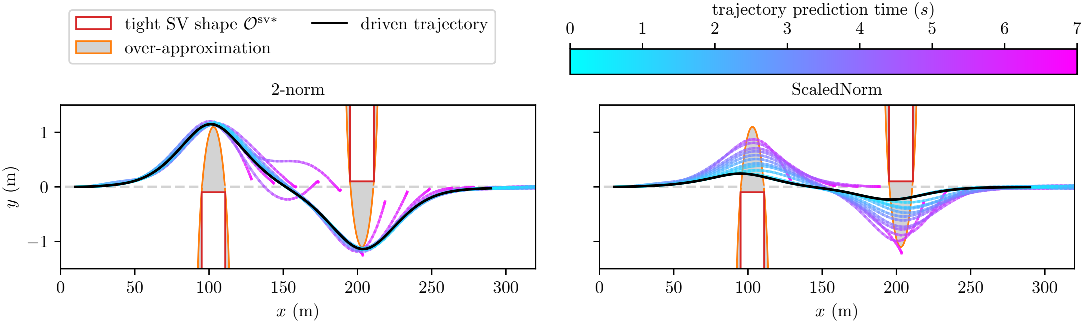
Tighten obstacle along prediction horizon towards closer predictions



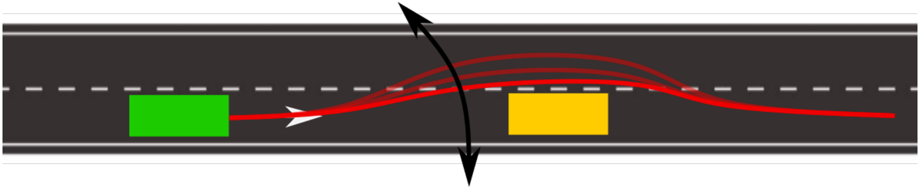
[4] Rudolf Reiter et al. *Progressive Smoothing for Motion Planning in Real-Time NMPC*. 2024. arXiv: 2403.01830 [eess.SY].

Challenge 1: Visualisation of *Progressive smoothing*

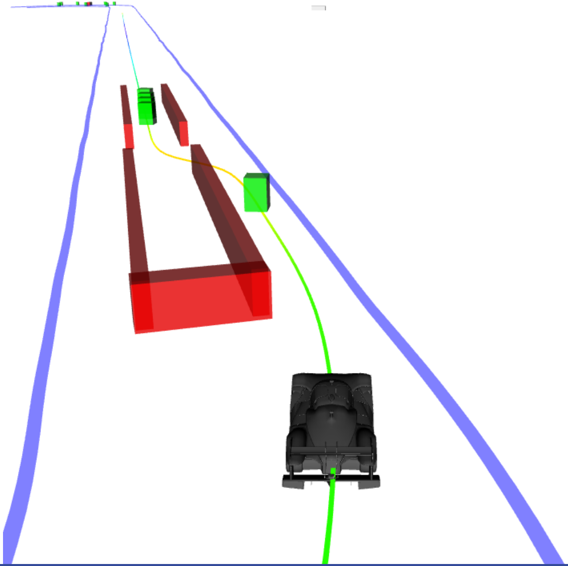
Vehicle passing two obstacles with 2-norm vs. progressive smoothing (scaled norm)
Planned trajectories in each RTI step:



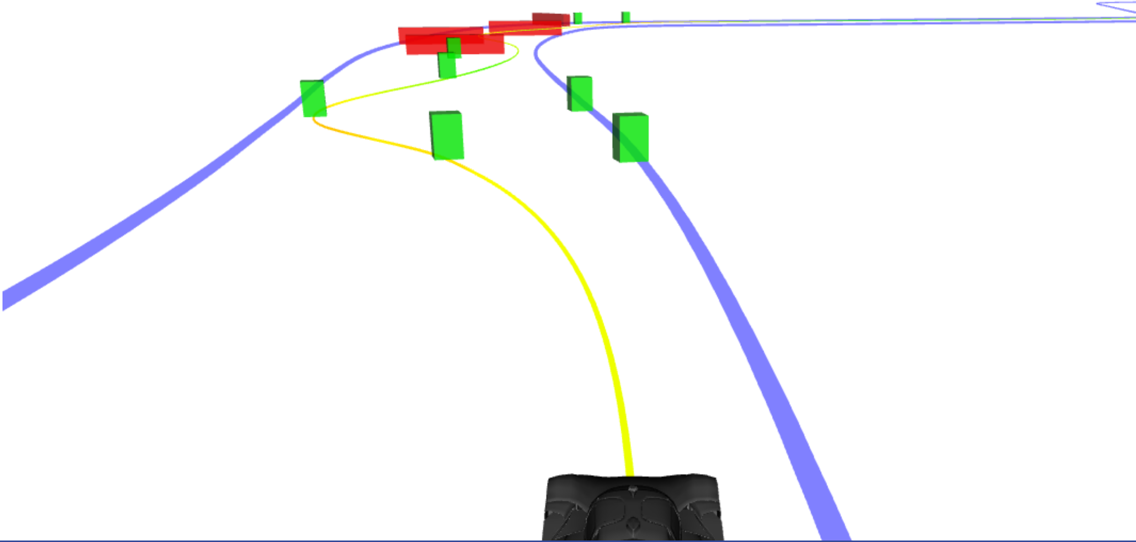
Challenge 2: Nonconvex non-homeomorphic planning space



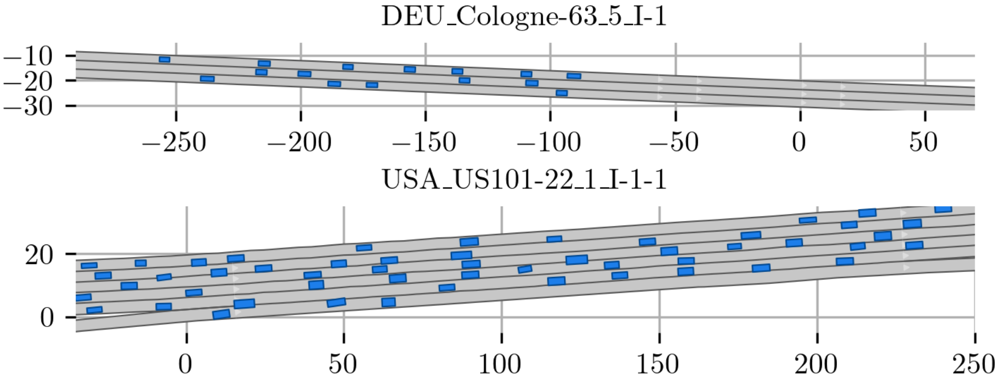
Challenge 2: Example 1



Challenge 2: Example 2



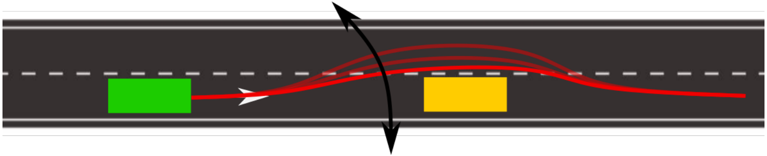
Challenge 2: Example 3



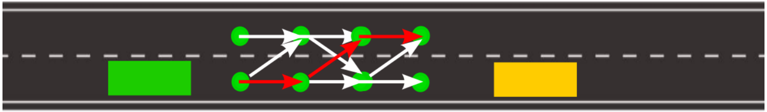
Global optimization for obstacle avoidance

Mixed-integer optimization

- Gradient-based optimization only works for local solutions in continuous space



- Approach 1: search in a discrete space



- Approach 2 : search in a mixed continuous-discrete space **mixed integer optimization**

Mixed-integer optimization for motion planning

- Appealing mathematical concept: discrete and continuous controls and states
- Building on powerful commercial solvers (Gurobi, CPLEX, ...)
- solves to global optimum

Mixed-integer optimization for motion planning

- Exponential worst-case computation time
- Reasonable performance only for mixed-integer quadratic programs

Mixed-integer quadratic programming for motion planning[5]

Simplifications

- Obtain linear model either through linearization or point-mass model

[5] Rien Quirynen, Sleiman Safaoui, and Stefano Di Cairano. *Real-time Mixed-Integer Quadratic Programming for Vehicle Decision Making and Motion Planning*. 2023. arXiv: 2308.10069 [math.OC].

Mixed-integer quadratic programming for motion planning[5]

Simplifications

- Obtain linear model either through linearization or point-mass model
- Linear boundary constraints through planning in Frenet frame

[5] Quirynen, Safaoui, and Cairano, *Real-time Mixed-Integer Quadratic Programming for Vehicle Decision Making and Motion Planning*.

Mixed-integer quadratic programming for motion planning[5]

Simplifications

- Obtain linear model either through linearization or point-mass model
- Linear boundary constraints through planning in Frenet frame
- Linear dynamic constraints require conservativeness

[5] Quirynen, Safaoui, and Cairano, *Real-time Mixed-Integer Quadratic Programming for Vehicle Decision Making and Motion Planning*.

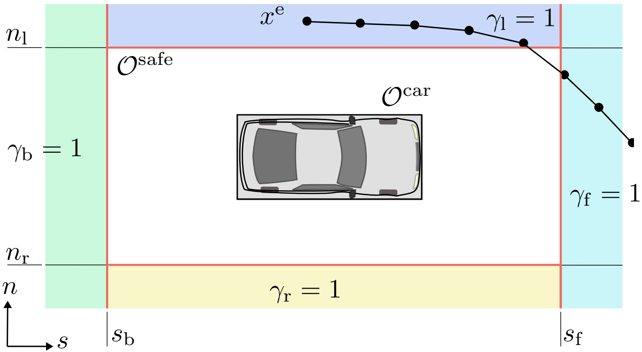
Mixed-integer quadratic programming for motion planning[5]

Simplifications

- Obtain linear model either through linearization or point-mass model
- Linear boundary constraints through planning in Frenet frame
- Linear dynamic constraints require conservativeness
- Defining convex disjunctive free spaces $\rightarrow$ integer variables

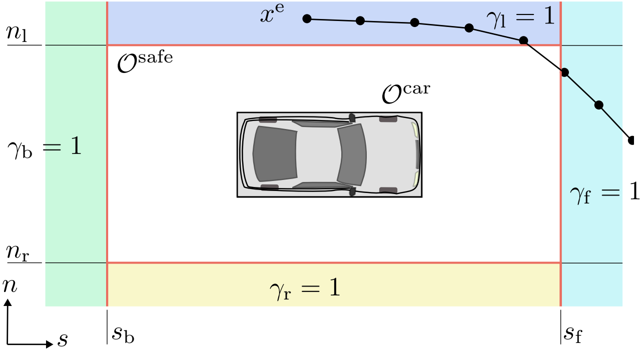
[5] Quirynen, Safaoui, and Cairano, *Real-time Mixed-Integer Quadratic Programming for Vehicle Decision Making and Motion Planning*.

Convex disjunctive free spaces



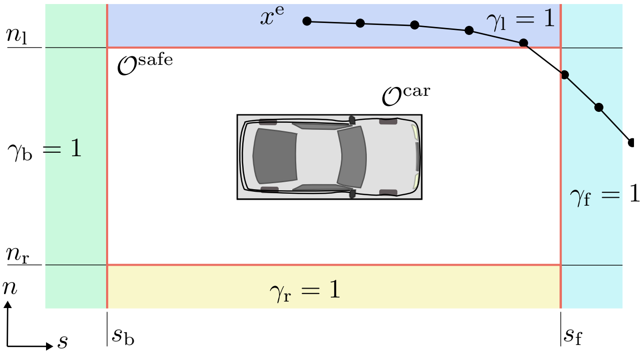
- Over-approximating obstacle $\mathcal{O}^{\mathrm{car}}$ by $\mathcal{O}^{\mathrm{safe}}$

Convex disjunctive free spaces



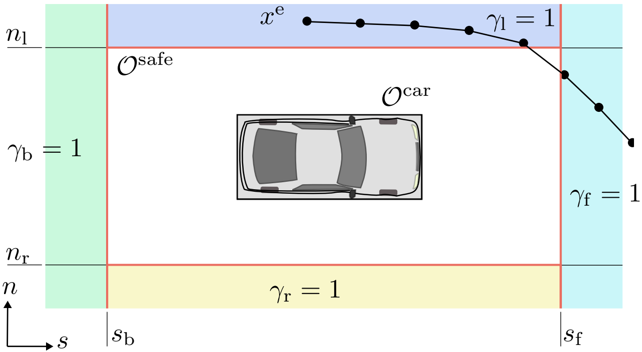
- Over-approximating obstacle $\mathcal{O}^{\mathrm{car}}$ by $\mathcal{O}^{\mathrm{safe}}$
- Split into 4 convex regions (left, right, front, back)

Convex disjunctive free spaces



- Over-approximating obstacle $\mathcal{O}^{\mathrm{car}}$ by $\mathcal{O}^{\mathrm{safe}}$
- Split into 4 convex regions (left, right, front, back)
- Assign binary indicator variables $\gamma_{\mathrm{l}}, \gamma_{\mathrm{r}}, \gamma_{\mathrm{f}}, \gamma_{\mathrm{b}} \in \{0, 1\}$

Convex disjunctive free spaces



- Over-approximating obstacle $\mathcal{O}^{\mathrm{car}}$ by $\mathcal{O}^{\mathrm{safe}}$
- Split into 4 convex regions (left, right, front, back)
- Assign binary indicator variables $\gamma_{\mathrm{l}}, \gamma_{\mathrm{r}}, \gamma_{\mathrm{f}}, \gamma_{\mathrm{b}} \in \{0, 1\}$
- Planning trajectory $x^{\mathrm{e}} = [x_0^{\mathrm{e}}, \ldots, x_N^{\mathrm{e}}]$ with $x_i^{\mathrm{e}} \notin \mathcal{O}^{\mathrm{safe}}$

Convex disjunctive free spaces

Binary variables to constrain region[6]

- Given compact set $\mathcal{X} \subset \mathbb{R}$ and continuous function $f : \mathcal{X} \to \mathbb{R}$
- Find bounds $\overline{M} \geq \max_{x \in \mathcal{X}} f(x)$ and $\underline{M} \leq \min_{x \in \mathcal{X}} f(x)$

Property

The implication $[f(x) > 0] \implies [\gamma = 1]$ of a binary variable $\gamma \in \{0, 1\}$ that gets activated if constraint $f(x) \geq 0$ is valid, is formulated as

$$f(x) \leq \overline{M}\gamma.$$

Property

The implication $[\gamma = 1] \implies [f(x) \geq 0]$ of a binary variable $\gamma \in \{0, 1\}$ that activates constraint $f(x) \geq 0$, is formulated as

$$f(x) \geq \underline{M}(1 - \gamma).$$

- construct disjunction for each region and add for each obstacle j and time k the constraint

$$(\gamma_{\mathrm{r}})_{j,k} + (\gamma_{\mathrm{l}})_{j,k} + (\gamma_{\mathrm{f}})_{j,k} + (\gamma_{\mathrm{b}})_{j,k} = 1, \quad j = 1, \ldots, N_{\mathrm{obs}},\ k = 1, \ldots, N$$

- Requires 4 binary variables per obstacle per time step $(N_{\mathrm{bin}} = 4N_{\mathrm{obs}}N) \to$ too many

[6] H. P. Williams. *Model building in mathematical programming*. Hoboken, N.J.: Wiley, 2013. ISBN: 9781118443330 1118443330.

Simplifying the problem further

- Static obstacle?
  1. Spatial reformulation of dynamics[7]
  2. Binary decisions only in lateral dimension[8] $\rightarrow N'_{\mathrm{bin}} = \mathcal{O}(N_{\mathrm{obs}})$, before $N_{\mathrm{bin}} = \mathcal{O}(N_{\mathrm{obs}}N)$

[7] Robin Verschueren et al. "Time-optimal Race Car Driving using an Online Exact Hessian based Nonlinear MPC Algorithm". In: *2021 European Control Conference (ECC)*. 2016.

[8] Rudolf Reiter et al. "Mixed-integer optimization-based planning for autonomous racing with obstacles and rewards". In: *IFAC-PapersOnLine* 54.6 (2021). 7th IFAC Conference on NMPC, pp. 99–106. ISSN: 2405-8963.

Simplifying the problem further

- Static obstacle?
  1. Spatial reformulation of dynamics[7]
  2. Binary decisions only in lateral dimension[8] $\rightarrow N'_{\text{bin}} = \mathcal{O}(N_{\text{obs}})$, before $N_{\text{bin}} = \mathcal{O}(N_{\text{obs}}N)$
- Multi-lane road environment?
  1. Optimize for transitions in the spatio-temporal coordinates *(submitted to IEEE TIV)*
  2. Binary decisions related to *gaps* on each lane $\rightarrow N''_{\text{bin}} = \mathcal{O}(N_{\text{obs}} + N)$

[7] Verschueren et al., "Time-optimal Race Car Driving using an Online Exact Hessian based Nonlinear MPC Algorithm".

[8] Reiter et al., "Mixed-integer optimization-based planning for autonomous racing with obstacles and rewards".

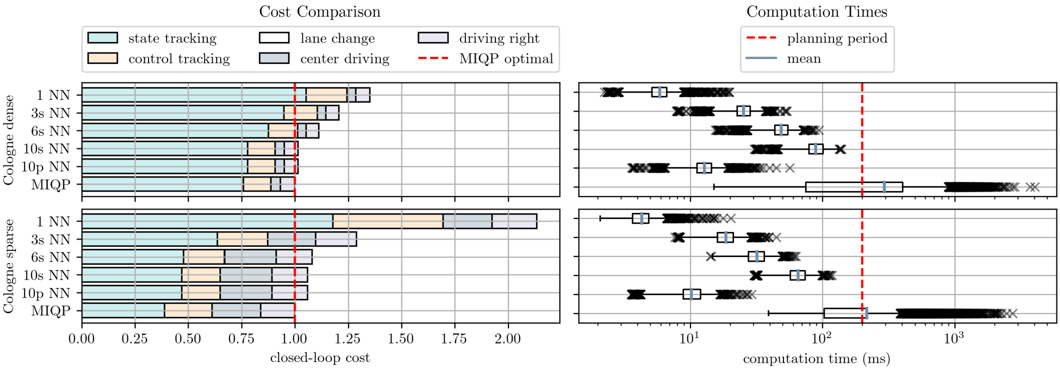
Simplifying the problem further

- Static obstacle?
  1. Spatial reformulation of dynamics[7]
  2. Binary decisions only in lateral dimension[8] $\rightarrow N'_{\text{bin}} = \mathcal{O}(N_{\text{obs}})$, before $N_{\text{bin}} = \mathcal{O}(N_{\text{obs}} N)$
- Multi-lane road environment?
  1. Optimize for transitions in the spatio-temporal coordinates *(submitted to IEEE TIV)*
  2. Binary decisions related to *gaps* on each lane $\rightarrow N''_{\text{bin}} = \mathcal{O}(N_{\text{obs}} + N)$
- Otherwise?
  1. Use machine learning to replace combinatorial part of optimizer *(submitted to IEEE TCST)*
  2. Train predictor for binary variables
  3. Fix binary variables
  4. Solve remaining QP

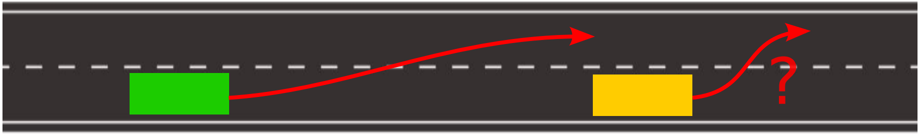
[7] Verschueren et al., "Time-optimal Race Car Driving using an Online Exact Hessian based Nonlinear MPC Algorithm".

[8] Reiter et al., "Mixed-integer optimization-based planning for autonomous racing with obstacles and rewards".

Results: Learning of combinatorial part of MIQP

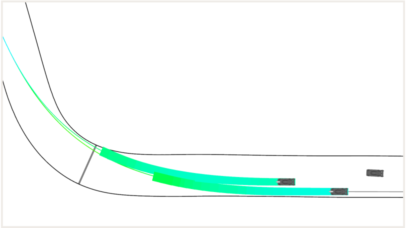


Challenge 3: Prediction and interaction

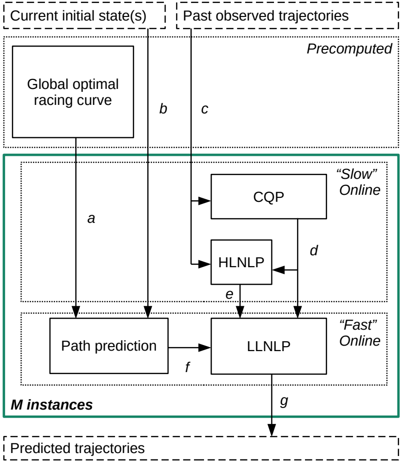


Obstacle prediction by inverse optimal control[9]

- Including a physics-based parametric model of the opponent inducing a *racing intention*
- The racing intention is modeled by means of a parametric nonlinear low-level program (LLNLP) for progress maximization
- The estimation of the parameters is performed by solving an inverse optimal control (IOC) problem, which enforces the optimality conditions for the LLNLP as constraints
- Output: Predicted trajectories (non-interactive)



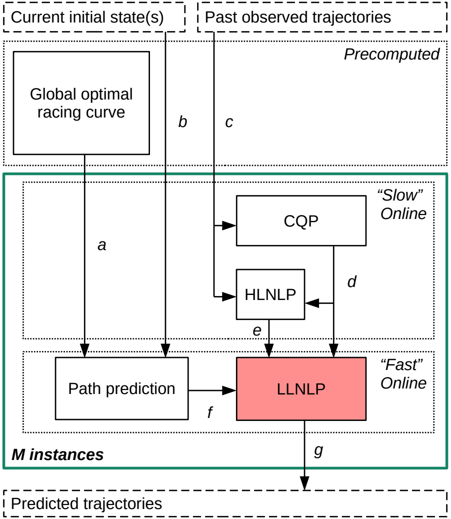
[9] Rudolf Reiter et al. "An Inverse Optimal Control Approach for Trajectory Prediction of Autonomous Race Cars". In: *2022 European Control Conference (ECC)*. 2022, pp. 146–153. DOI: 10.23919/ECC55457.2022.9838100.



a: global racing path
b: initial state $\bar{x}_0$
c: trajectory data samples
d: constraints $a_{\max}$
e: weights w
f: Cartesian coordinates and curvature parameters of blended path segment $\bar{\kappa}$
g: predicted trajectory

The Prediction Algorithm

Low-Level Program for Trajectory Prediction (LLNLP)

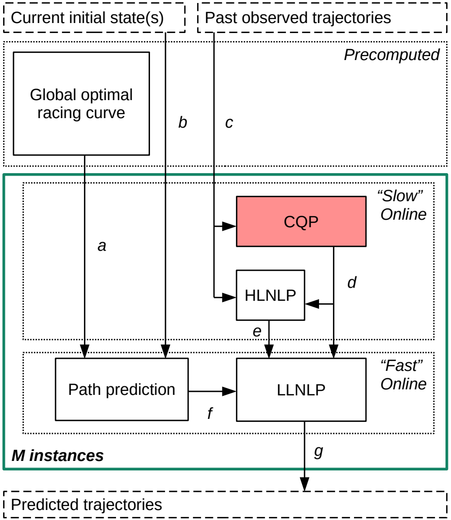


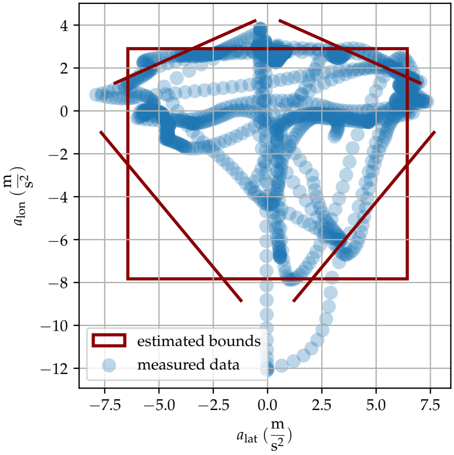
- Nonlinear program to maximize progress (x_N) along given path
- Weights Q, R, q_N estimated by HLNLP
- Acceleration constraints $h_a(x_k, \bar{\kappa}, a_{\max})$ estimated by CQP

$$\begin{aligned}
\min_{\substack{x_0,\dots,x_N,\\ U_0,\dots,U_{N-1}\\ s_0,\dots,s_N}} \quad & \sum_{k=0}^{N-1} \|x_k - x_k^{\mathrm{r}}\|_{2,Q}^2 + \|U_k - U_k^{\mathrm{r}}\|_{2,R}^2 + {q_N}^\top x_N + \sum_{k=0}^{N} \alpha_1 \mathbf{1}^\top s_{\mathrm{LL},k} + \alpha_2 \|s_{\mathrm{LL},k}\|_2^2 \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& x_{k+1} = F(x_k, U_k, \Delta t), \qquad k = 0, \dots, N-1 \\
& \underline{x} \preccurlyeq x_k \preccurlyeq \overline{x} \\
& 0 \preccurlyeq h_a(x_k, \bar{\kappa}, a_{\max}) + s_{\mathrm{LL},k} \\
& 0 \preccurlyeq s_{\mathrm{LL},k}, \qquad k = 0, \dots, N,
\end{aligned}$$

The Prediction Algorithm

Quadratic Program for Constraint Estimation

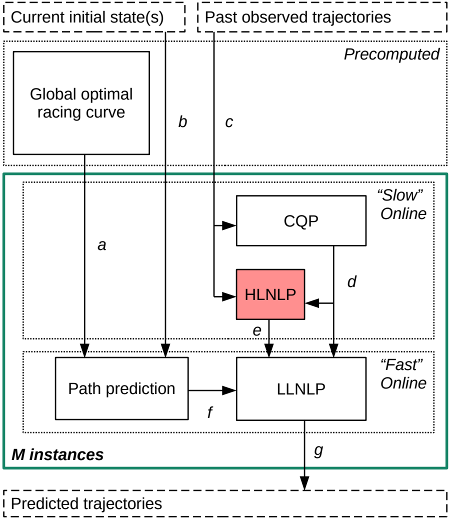




- Constraints are estimated separatly from the weights
- Symmetric polytope with 8 bounds (5 independent) fitted to data
- Iterative QP, with previously estimated value as "arrival term" (moving horizon estimation)

The Prediction Algorithm

High Level Program for Weight Estimation (HLNLP)



The Prediction Algorithm

High Level Program for Weight Estimation (HLNLP)

- We optimize for the weights $w = [Q, R, q_N]$ of the LLNLP
- L2 loss on observed trajectories and predicted trajectories
- We use only states x and controls u that are solutions of the LLNLP $P_{\mathrm{LL}}(w, \bar{x}_0, \bar{\kappa}, a_{\max})$
- $\rightarrow$ bi-level optimization problem

$$\begin{aligned} \min_{X, U, w} \quad & \sum_{k=1}^{N_T - 1} \|x_k - \bar{x}_k\|_{2, Q_k}^2 + \|w - \hat{w}\|_{2, P^{-1}}^2 \\ \text{s.t.} \quad & X, U \in \operatorname{argmin} P_{\mathrm{LL}}(w, \bar{x}_0, \bar{\kappa}, a_{\max}) \\ & w \succcurlyeq 0 \end{aligned}$$

- We use the the KKT conditions of the LLNLP as constraints in the HLNLP
- Homotopy on penalized relaxation
- Arrival cost with weights P^{-1}

The simulation:

- Simulation framework with dynamic vehicle model
- Comparisons with Notebook
- Hardware-in-the-loop for competitions
- Las Vegas race track
- 1k randomly parameterized test runs
- (Due Covid currently only simulated races)

The setup:

- Hardware: HP Elitebook, Intel Core i7-8550 CPU (1.8 GHz) and Nvidia Drive PX2
- The used frequency for the synchronous LLNLP was 10 Hz
- The HLNLP and CQP ran asynchronously
- 200 seconds until HLNLP converged

Results

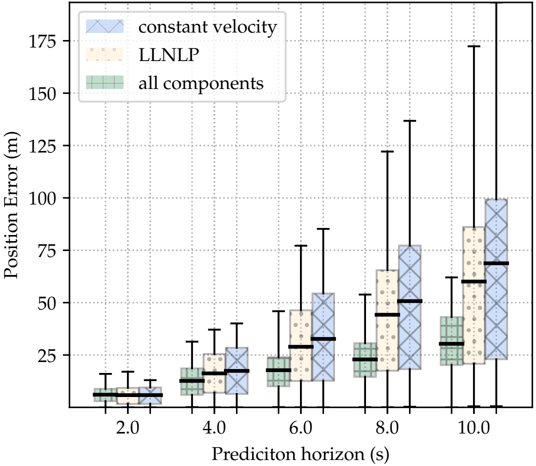
Timings

Table: Solver timing statistics (Nvidia Drive PX2)

Component	Solver	t_{max} (ms)	t_{ave} (ms)	fail rate (%)
PP	none	< 1	< 1	0
CQP	OSQP	15.5	8.1	0
HLNLP	IPOPT	6237	520	5
LLNLP	acados, HPIPM	2748	91	0.2

Results

Final Prediction Errors by Prediction Horizon (converged)



Challenge 3: Interaction

- Strategic planning of autonomous race cars → blocking agents, efficient overtaking
- Other agents have static policies (otherwise game theoretic problem)

- Approach 1: use optimization-based control (NMPC)
- Problem: complex prediction model inside optimization problem

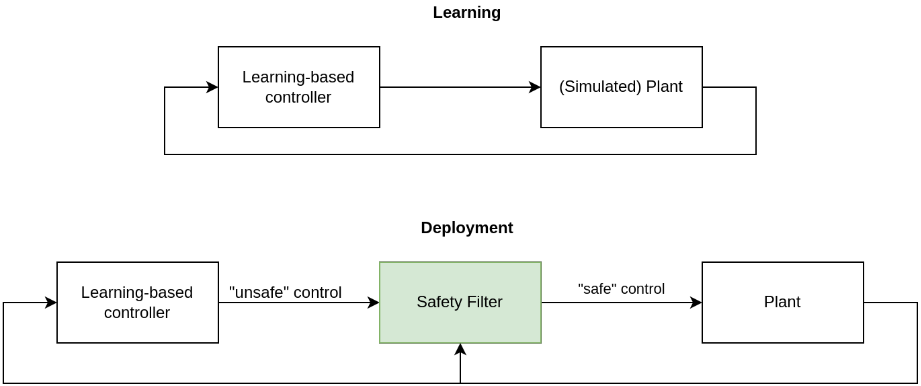
- Approach 2: use reinforcement learning
- Problem: can hardly account for safety, loads of data needed for simple maneuvers

- Our approach[10]: Combine reinforcement learning and NMPC hierarchically

[10] Rudolf Reiter et al. "A Hierarchical Approach for Strategic Motion Planning in Autonomous Racing". In: *2023 European Control Conference (ECC)*. 2023, pp. 1–8. DOI: 10.23919/ECC57647.2023.10178143.

Relation to the safety filter[11]
Original

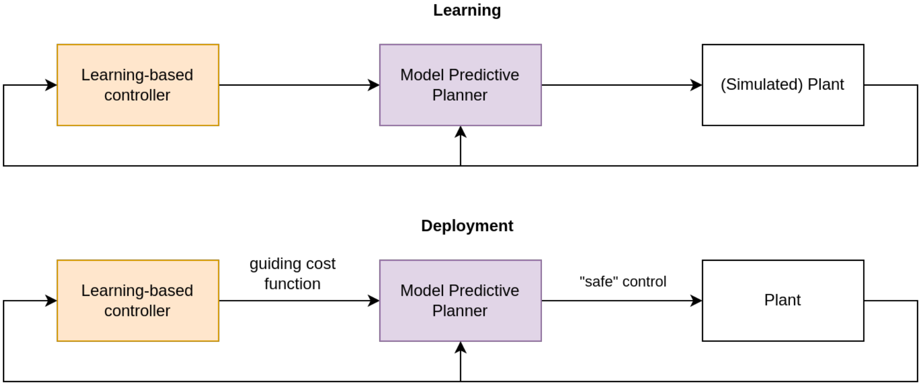
The safety filter uses an NLP to project controls onto safe sets



[11] Kim Peter Wabersich and Melanie N. Zeilinger. "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems". In: *Automatica* 129 (2021), p. 109597. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2021.109597.

Relation to the safety filter

Our approach



Safety Filter

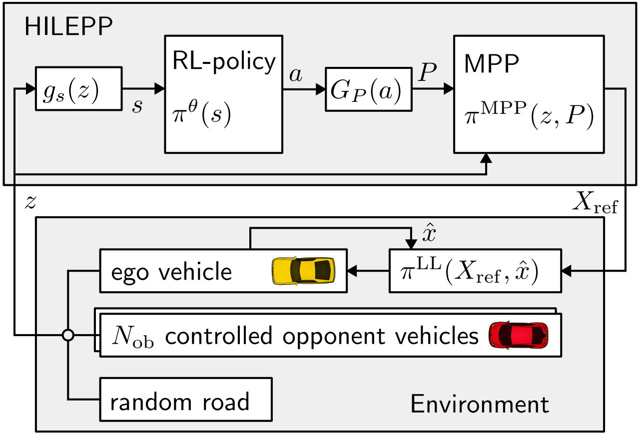
$$
\begin{aligned}
\min_{X,U} \quad & \|u_0 - \bar{a}\|_R^2 \\
\text{s.t.} \quad & x_0 = \bar{x}_0, \quad x_N \in \mathcal{S}^{\mathrm{t}} \\
& x_{i+1} = F(x_i, u_i), \quad i = 0, \dots, N-1 \\
& x_i \in \mathcal{X}, u_i \in \mathcal{U}, \quad i = 0, \dots, N-1
\end{aligned}
$$

HILEPP (ours)

$$
\begin{aligned}
\min_{X,U} \quad & L(X, U, a) \\
\text{s.t.} \quad & x_0 = \hat{x}_0, \quad x_N \in \mathcal{S}^{\mathrm{t}} \\
& x_{i+1} = F(x_i, u_i), \quad i = 0, \dots, N-1 \\
& x_i \in \mathcal{X}, u_i \in \mathcal{U}, \quad i = 0, \dots, N-1,
\end{aligned}
$$

Architecture

Details



Invariance pre-conditioning function $g_s(z)$ sets inputs s to RL policy $a = \pi^\Theta(s)$. Function $G_P(a)$ transforms RL actions a to MPP parameters P. Policy $\pi^{\text{MPP}}(z, P)$ solves NLP and outputs safe reference X^{ref}.

NMPC (MPP) formulation

Gerneral

- MPP is a NMPC used as planner
- Kinematic vehicle model in Frenet coordinate frame
- Obstacle avoidance with ellipses - circles
- Obstacle prediction in two modes (Defined according to racing rules):
  - *Follower: generously* assuming straight linear motion in Frenet coordinate frame
  - *Leader: evasively* allowing only decelerating linear motion

Cost parameterization through RL actions:

$$G_P(a) : a \to \left(\xi_{\mathrm{ref},0}(a), \ldots, \xi_{\mathrm{ref},N}(a), Q_{\mathrm{w}}(a)\right)$$

$$\xi_{\mathrm{ref},k}(a) = \begin{bmatrix}0 & n & 0 & v & 0\end{bmatrix}^\top \in \mathcal{R}^{n_x}$$

$$Q_{\mathrm{w}}(a) = \mathrm{diag}(\begin{bmatrix}0 & w_n & 0 & w_v & 0\end{bmatrix})$$

NMPC (MPP) parameterized cost:

$$L(X, U, a, \Xi) = \sum_{k=0}^{N-1} \|x_k - \xi_{\mathrm{ref},k}(a)\|^2_{Q_{\mathrm{w}}(a)} + \|u_k\|^2_R$$

$$+ \|x_N - \xi_{\mathrm{ref},N}(a)\|^2_{Q^t} + \sum_{k=0}^{N} \|\sigma_k\|^2_{Q_{\sigma,2}} + |q_{\sigma,1}^\top \sigma_k|.$$

We compare two action vectors (with or without setting weights):

- HILEPP-I: $a_{\mathrm{I}} := [n, v]^\top$
- HILEPP-II: $a_{\mathrm{II}} := [n, v, w_n, w_v]^\top$

General

- Markov assumption, state space $\mathcal{S}$, action space $\mathcal{A}$, looking for policy $\pi^\theta : \mathcal{S} \mapsto \mathcal{A}$, reward function $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$
- We use a *soft actor critic* algorithm with actor π^θ and a critic Q^ϕ

Specific

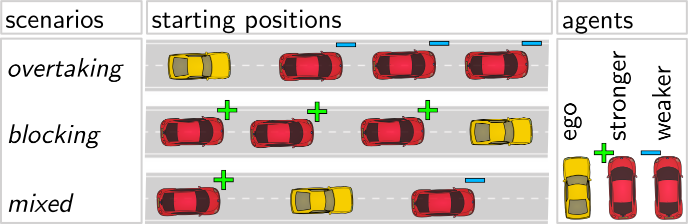
- Pre-processing function from ego state $s = [n, v, \alpha]^\top$, road curvature evaluations $\kappa(\cdot)$ and obstacle states z to (partly) invariant RL states $s_{\mathrm{ob}_i} = [\zeta_{\mathrm{ob}_i} - \zeta, n_{\mathrm{ob}_i}, v_{\mathrm{ob}_i}, \alpha_{\mathrm{ob}_i}]^\top$

$$s_k = g_s(z_k) = [\kappa(\zeta + d_i), \ldots, \kappa(\zeta + d_N), s^\top, s_{\mathrm{ob}_1}^\top, \ldots, s_{\mathrm{ob}_N}^\top]^\top$$

- We use the reward for center line speed $\dot{s}$ and the total rank, with

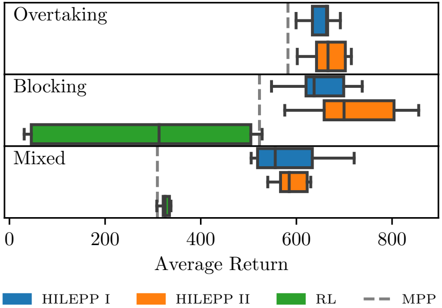
$$R(s, a) = \frac{\dot{\zeta}}{200} + \sum_{i=1}^{N_{\mathrm{ob}}} 1_{\zeta_k > \zeta_k^{\mathrm{ob}_i}}$$

- Training of $\sim 10^6$ steps in randomized simulated scenarios
- Only the ego agent is trained, opponents only use MPP
- Three different scenario types



- Comparison of
  - MPP
  - RL
  - HILEPP-I (only reference states)
  - HILEPP-II (reference states and weights)

- pure RL *struggled* to keep up even with MPP
- overtaking does not require much strategy
- HILEPP-II performs better than HILEPP-I



Module	Mean± Std.	Max
MPP	5.45 ± 2.73	8.62
RL policy	0.13 ± 0.01	0.26
HILEPP-I	6.90 ± 3.17	9.56
HILEPP-II	7.41 ± 2.28	9.21

Table: Computation times (ms) of modules.

Conclusion and discussion

- Motion planning with collision avoidance is challenging due to
  - nonconvexity
  - interaction
  - uncertainty
  - safety requirements
  - real-time requirements
- Approaches from different communities
  - Continuous optimization
  - Discrete optimization $\rightarrow$ graph search, tree search
  - Mixed-integer optimization
  - Reinforcement learning
- It seems promising to combine approaches based on individual strengths
  - MPC+RL
  - Safety filter
  - Tailored mixed-integer programming
  - Learning-based mixed-integer programming
  - MINLP?

Thanks to all supervisors and colleagues!

Thank you for your attention!