Beyond Nonlinear Model Predictive Control for Autonomous Driving

Rudolf Reiter

Systems Control and Optimization Laboratory (syscop)

Transportation Seminar
KTH Royal Institute of Technology
Stockholm, Freiburg
October 27, 2023

Introduction

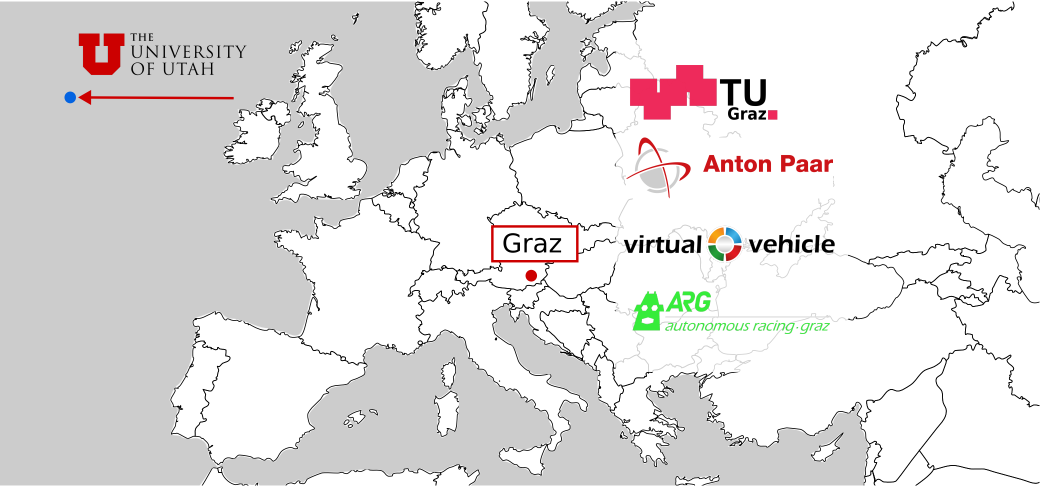
- Task: optimization based planning (and control) of autonomous vehicles
- Scenarios: autonomous racing and multi-lane traffic
- Challenges: interactions, combinatorial complexity, real-time requirements
- Tools: real-time optimization, combinatorial optimization and machine learning

Outline

1. Personal introduction
2. Bird's eye view on my research and outline
3. Preliminaries
   - Nonlinear model predictive control
4. Modeling
   - Frenet coordinate system
5. Obstacle constraints
   - Dual formulation
6. Obstacle prediction
   - Inverse optimal control formulation
7. Obstacle avoidance
   - Mixed-integer formulation
8. Strategic driving
   - Hierarchical algorithm with reinforcement learning

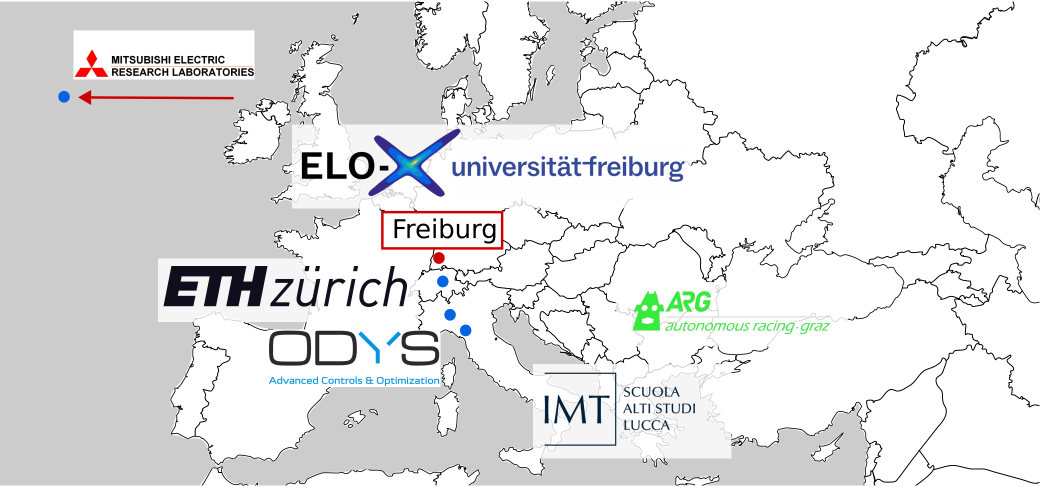
Personal Introduction

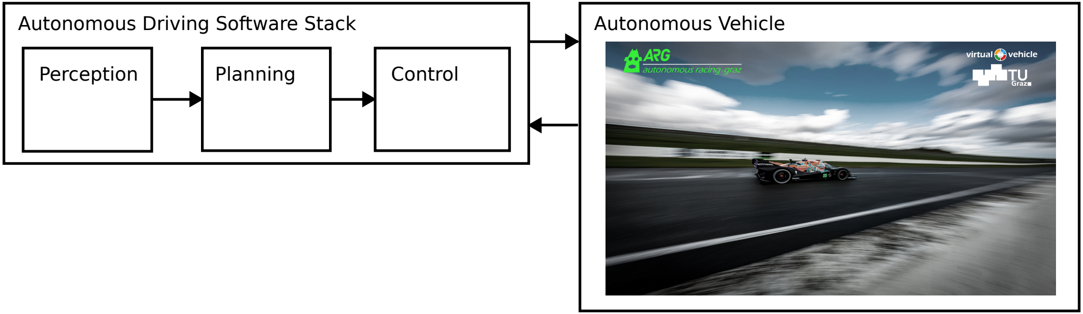
Salzburg, Austria: until 2009


THE UNIVERSITY OF UTAH
TU Graz
Anton Paar
Graz
virtual vehicle
ARG
autonomous racing·graz

Personal Introduction

Freibug, Germany: until ∼2024




Autonomous Driving Software Stack
Perception
Planning
Control
Autonomous Vehicle
ARG
autonomous racing graz
virtual vehicle
TU Graz

Bird's eye view on my research and outline

Autonomous Driving Software Stack

Perception → Planning → Control

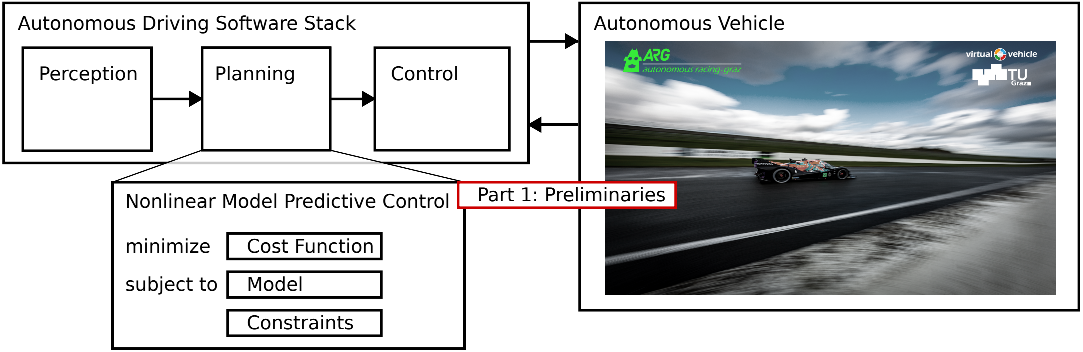
Nonlinear Model Predictive Control

minimize Cost Function

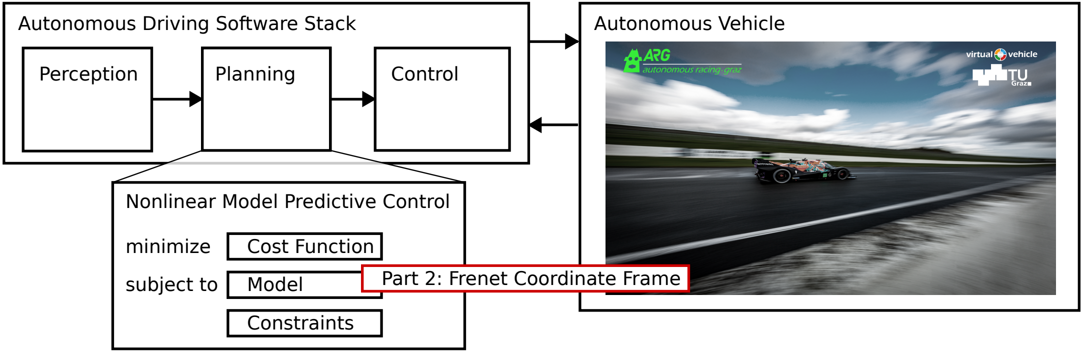
subject to Model

Constraints

Autonomous Vehicle


Autonomous Driving Software Stack
Perception
Planning
Control
Autonomous Vehicle
ARG
autonomous racing graz
virtual vehicle
TU Graz
Nonlinear Model Predictive Control
Part 1: Preliminaries
minimize
Cost Function
subject to
Model
Constraints

Bird’s eye view on my research and outline



Bird's eye view on my research and outline

Autonomous Driving Software Stack

Perception → Planning → Control

Autonomous Vehicle

Nonlinear Model Predictive Control

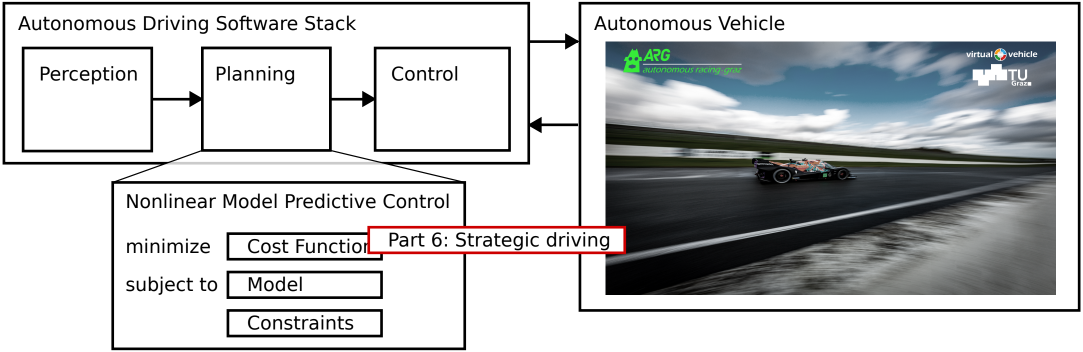
minimize Cost Function

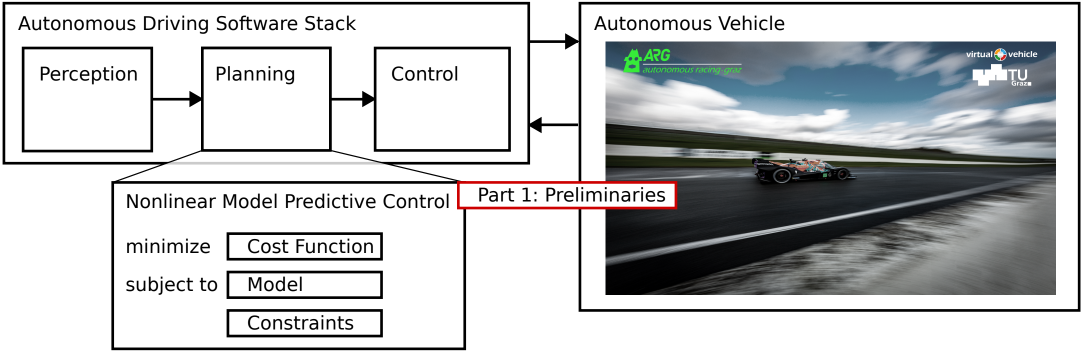
subject to Model

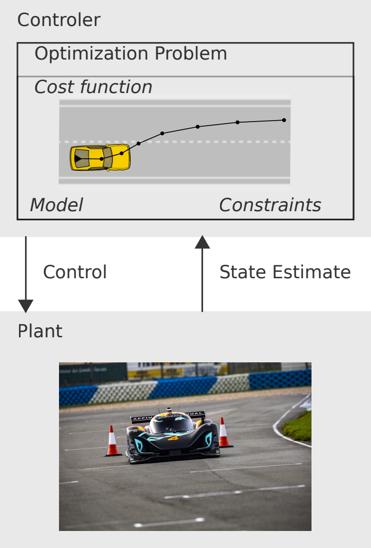
Constraints

Part 3,4 and 5:
Obstacle constraint formulation, prediction, global optimization

Bird's eye view on my research and outline




Autonomous Driving Software Stack
Perception
Planning
Control
Autonomous Vehicle
ARG
autonomous racing graz
virtual vehicle
TU Graz
Nonlinear Model Predictive Control
Part 1: Preliminaries
minimize
Cost Function
subject to
Model
Constraints



- Cost function: quadratic (reference tracking)
- Model: nonlinear (kinematic/dynamic single track)
- Constraints: non-convex (often concave due convex obstacle shapes)

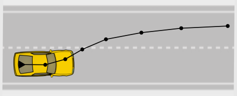
Sounds scary!

- Computation time?
- Solution Guarantee?
- Optimality?

Controler

Optimization Problem

Cost function



Model *Constraints*

Control

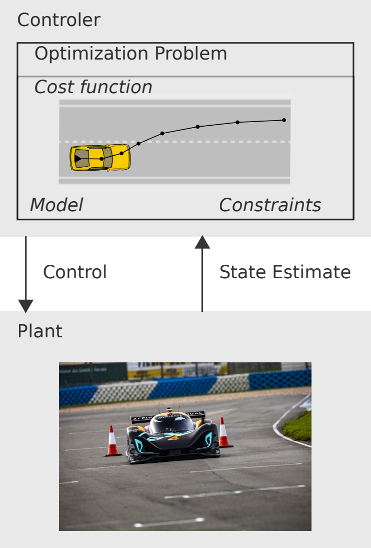
State Estimate

Plant

- Computation time?
  - ✓ fast structure exploiting QP solvers (e.g., HPIPM[a])
  - ✓ fast NLP solvers (e.g., acados[b])
  - ✓ real-time iterations
- Solution Guarantee?
- Optimality?

[a] Gianluca Frison and Moritz Diehl. "HPIPM: a high-performance quadratic prog model predictive control". In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World C ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2020.12.073.

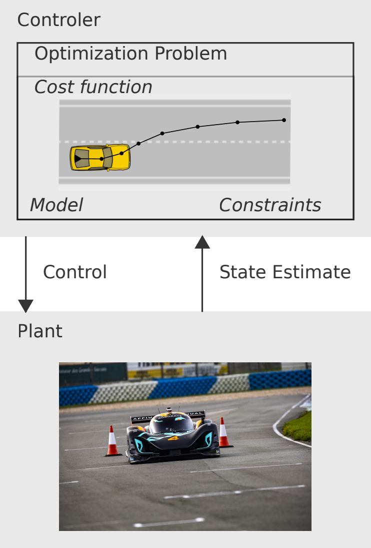
[b] Robin Verschueren et al. "acados – a modular open-source framework for fast e control". In: *Mathematical Programming Computation* (2021). ISSN: 1867-2957. D 10.1007/s12532-021-00208-8.



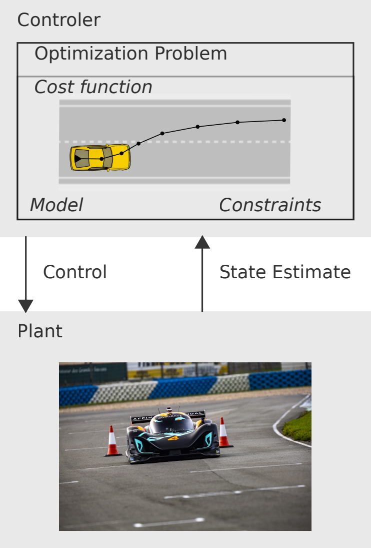
- Computation time?
- Solution Guarantee?
  - ✗ not directly
  - ✓ workarounds: saving last feasible trajectory, backup controller
- Optimality?

Preliminaries

Nonlinear Model Predictive Control



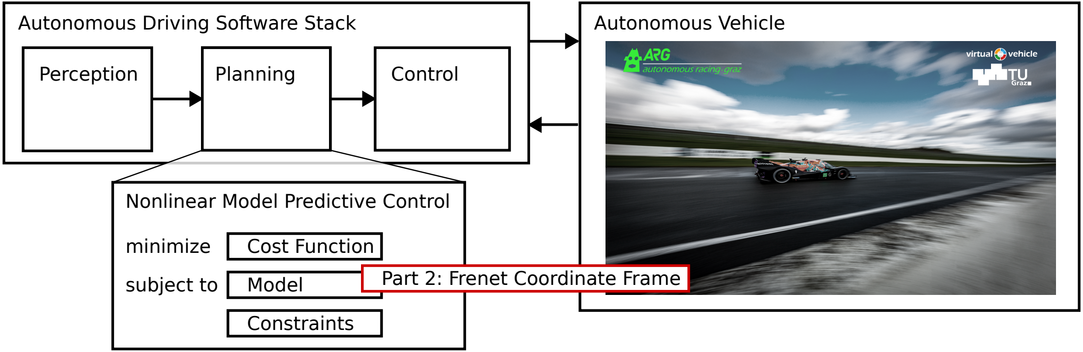
- Computation time?
- Solution Guarantee?
- Optimality?
  - ✗ local, given sufficiently close initial guess
  - ✓ local solutions are often good
  - ✓ initial guess provided by other module



Our usual setting for solving the nonlinear optimization problem for autonomous driving

- Direct multiple shooting formulation
- Gauss-Newton Hessian approximation
- No condensing of QP required
- RK4 integration, step size $20 - 100\text{ms}$
- Horizon of 10s
- Terminal safe set often for velocity$=0\frac{\text{m}}{\text{s}}$
- No globalization, full steps
- Slack variables for feasibility

Part 2: Model[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.

Modeling in two coordinate frames

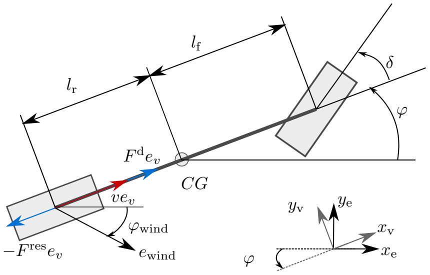
Kinematic single track model in Cartesian coordinate frame (CCF)

- 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

$$\dot{x}^{\mathrm{c,C}} = f^{\mathrm{c,C}}(x^{\mathrm{C}}, u) = \begin{bmatrix} v\cos(\varphi) \\ v\sin(\varphi) \\ \frac{v}{l}\tan(\delta) \end{bmatrix} \qquad (1)$$

- Dynamics of CCF independent states

$$\dot{x}^{\neg\mathrm{c}} = f^{\neg\mathrm{c}}(x^{\neg\mathrm{c}}, u, \varphi) = \begin{bmatrix} \frac{1}{m}(F^{\mathrm{d}} - F^{\mathrm{wind}}(v,\varphi) - F^{\mathrm{roll}}(v)) \\ r \end{bmatrix} \qquad (2)$$



Kinematic single track model in Frenet coordinate frame (FCF)

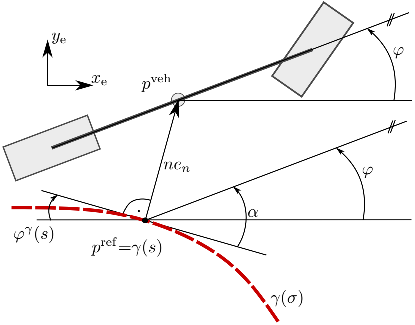
- Transformation:

$$x^{\mathrm{c,F}} = \mathcal{F}_\gamma(x^{\mathrm{c,C}}) = \begin{bmatrix} s^* \\ (p^{\mathrm{veh}} - \gamma(s^*))^\top e_n \\ \varphi^\gamma(s^*) - \varphi \end{bmatrix}, \quad (3)$$

$$s^*(p^{\mathrm{veh}}) = \arg\min_\sigma \left\| p^{\mathrm{veh}} - \gamma(\sigma) \right\|_2^2 . \quad (4)$$

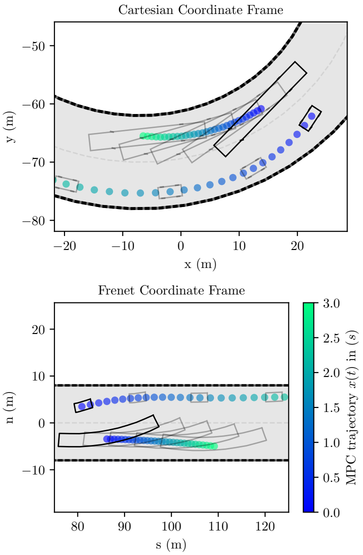
- Frenet states $x^{\mathrm{c,F}} = \mathcal{F}_\gamma(x^{\mathrm{c,C}}) = [s, n, \alpha]^\top \in \mathbb{R}^3$
- Full state vector: $x^{\mathrm{F}} = [x^{\mathrm{c,F}\top} \quad x^{\neg\mathrm{c}\top}]^\top$
- Dynamics of FCF dependent states

$$\dot{x}^{\mathrm{c,F}} = f^{\mathrm{c,F}}(x^{\mathrm{F}}, u) = \begin{bmatrix} \frac{v\cos(\alpha)}{1-n\kappa(s)} \\ v\sin(\alpha) \\ \frac{v}{l}\tan(\delta) - \frac{\kappa(s)v\cos(\alpha)}{1-n\kappa(s)} \end{bmatrix}. \quad (5)$$



Modeling in two coordinate frames

Comparison

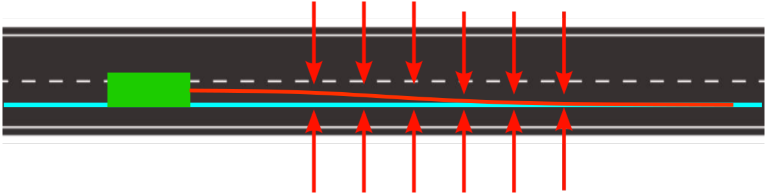


Feature	CCF	FCF
reference definition	✗	✓
boundary constraints	✗	✓
obstacle specification	✓	✗
disturbance specification	✓	✗

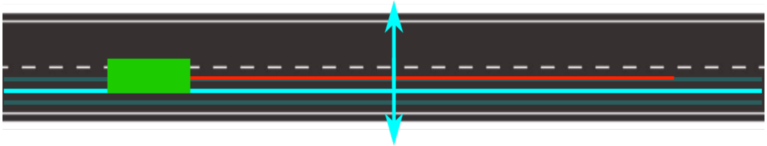
Modeling in two coordinate frames

Frenet Coordinate Frame Reference

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

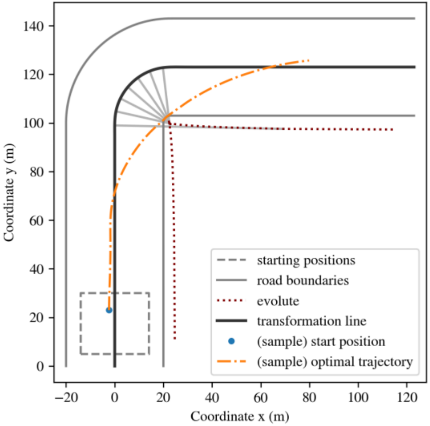


  - Racing: free to choose



Modeling in two coordinate frames

Frenet Coordinate Frame Reference

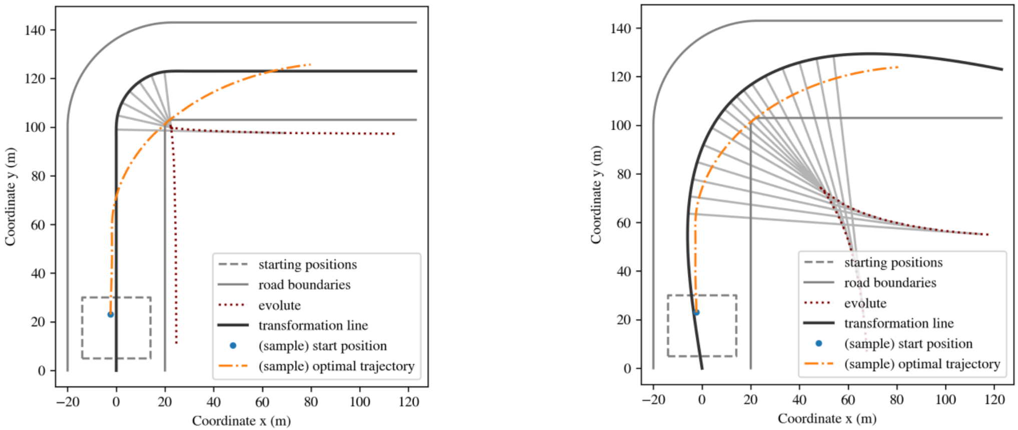


- The transformation has one big issue!
- Singular region at points $[s, n]^\top$, with $1 - n\kappa(s) = 0$
- Luckily usually no problem.

Can use the free choice of the reference in racing scenarios to our advantage[2]

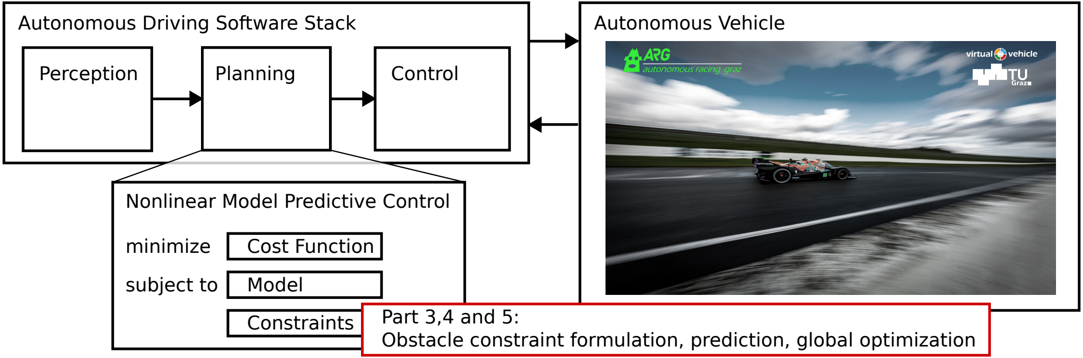
[2] Reiter and Diehl, "Parameterization Approach of the Frenet Transformation for Model Predictive Control of Autonomous Vehicles".

Solving a priori an optimization problem to obtain $\gamma(\sigma)$ that pushes the evolute outside and increases other favorable numercial properties for NMPC.[3]



[3] Reiter and Diehl, "Parameterization Approach of the Frenet Transformation for Model Predictive Control of Autonomous Vehicles".

Part 3: Obstacle constraint formulation[4]



[4] 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. DOI: https://doi.org/10.1016/j.ejcon.2023.100847. URL: https://www.sciencedirect.com/science/article/pii/S0947358023000766.

Part 3: Obstacle constraint formulation

Problem Statement

- Task: Obstacle formulation for the Frenet Coordinate Frame

- Basic approach: use optimization-based control: (Cartesian) NMPC
- Problem: nonconvexities and nonlinearities

- Variation: transform model into curvilinear coordinate frame (Frenet Frame)
- Problem: new coordinate frame makes part of problem more non-smooth

- Our idea: Use redundantly two coordinate frames
- Questions: How to formulate it? Speedup? Other advantages?

Part 3: Obstacle constraint formulation

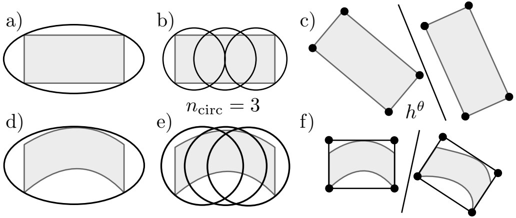
Outline

1. Obstacle avoidance
2. Ways to combine both models
3. NMPC Algorithm
4. Results

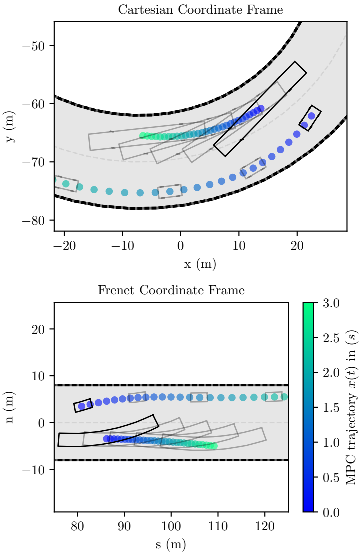
Obstacle avoidance

Comparison of several different obstacle avoidance formulations

1. Ellipse - circle
2. Covering circles
3. Separating hyper-planes



Remember: Frenet Coordinate Frame vs. Cartesian Coordinate Frame



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

Ways to combine both models

Goal:

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

Possible formulations of NMPC:

- Use only one CF, approximate and simplify non-smooth constraints
- Model dynamics in *one* CF, use Frenet transformation $\mathcal{F}_\gamma$ or inverse Frenet transformation ${\mathcal{F}_\gamma}^{-1}$ to obtain *other* states
- Model dynamics redundantly in *both* CFs

Ways to combine both models

Conventional

Possible formulations of NMPC:

- Use only one CF, approximate and simplify non-smooth constraints
  - Main frame CCF: approximate $\mathcal{F}_\gamma$ with artificial path state (MPCC) *(Not reviewed here)*
  - Main frame FCF: 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
  - Main frame CCF ✗: $\mathcal{F}_\gamma$ is an nonlinear optimization problem by itself
  - Main frame FCF ✓: $\mathcal{F}_\gamma{}^{-1}$ can be obtained efficiently $\rightarrow$ direct elimination
- Model dynamics redundantly in *both* CFs

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

NMPC Problem

Direct elimination

$$\min_{\substack{x_0^{\mathrm{F}},\dots,x_N^{\mathrm{F}},\\ u_0,\dots,u_{N-1}\\ \theta_1,\dots,\theta_{n_{\mathrm{opp}}}}} \quad \sum_{k=0}^{N-1} \|u_k\|_R^2 + \left\|x_k^{\mathrm{F}} - x_{\mathrm{ref},k}^{\mathrm{F}}\right\|_Q^2 + \left\|x_N^{\mathrm{F}} - x_{\mathrm{ref},N}^{\mathrm{F}}\right\|_{Q_N}^2$$

$$\begin{aligned}
\text{s.t.} \quad & x_0^{\mathrm{F}} = \hat{x}_0^{\mathrm{F}}, && \\
& x_{i+1}^{\mathrm{F}} = \Phi^{\mathrm{F}}(x_i^{\mathrm{F}}, u_i, \Delta t), && i = 0,\dots,N-1, \\
& \underline{u} \leq u_i \leq \overline{u}, && i = 0,\dots,N-1, \\
& \underline{x}^{\mathrm{F}} \leq x_i^{\mathrm{F}} \leq \overline{x}^{\mathrm{F}}, && i = 0,\dots,N, \\
& \underline{x}^{\mathrm{c,C}} \leq \mathcal{F}_\gamma^{-1}(x^{\mathrm{c,F}}) \leq \overline{x}^{\mathrm{c,C}}, && i = 0,\dots,N, \\
& \underline{a}^{\mathrm{lat}} \leq a_{\mathrm{lat}}^{\mathrm{F}}(x_i) \leq \overline{a}^{\mathrm{lat}}, && i = 0,\dots,N, \\
& v_N \leq \overline{v}_N, && \\
& \mathcal{F}_\gamma^{-1}(x^{\mathrm{c,F}}) \in \mathcal{P}(x_i^{\mathrm{c,opp,j}}, \theta_j), && i = 0,\dots,N-1, \\
& && j = 1,\dots,n_{\mathrm{opp}}.
\end{aligned} \tag{6}$$

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

NMPC Problem

Lifted

$$
\begin{aligned}
\min_{\substack{x_0^{\mathrm{d}},\dots,x_N^{\mathrm{d}},\\ u_0,\dots,u_{N-1}\\ \theta_1,\dots,\theta_{n_{\mathrm{opp}}}}} \quad & \sum_{k=0}^{N-1} \|u_k\|_R^2 + \left\|x_k^{\mathrm{F}} - x_{\mathrm{ref},k}^{\mathrm{F}}\right\|_Q^2 + \left\|x_N^{\mathrm{F}} - x_{\mathrm{ref},N}^{\mathrm{F}}\right\|_{Q_N}^2 \\
\text{s.t.} \quad & x_0^{\mathrm{d}} = \hat{x}_0^{\mathrm{d}}, \\
& x_{i+1}^{\mathrm{d}} = \Phi^{\mathrm{d}}(x_i^{\mathrm{d}}, u_i, \Delta t), i = 0, \dots, N-1, \\
& \underline{u} \leq u_i \leq \overline{u}, \qquad i = 0, \dots, N-1, \\
& \underline{x}^{\mathrm{d}} \leq x_i^{\mathrm{d}} \leq \overline{x}^{\mathrm{d}}, \qquad i = 0, \dots, N, \\
& \underline{a}^{\mathrm{lat}} \leq a_{\mathrm{lat}}(x_i^{\mathrm{d}}) \leq \overline{a}^{\mathrm{lat}}, i = 0, \dots, N, \\
& v_N \leq \overline{v}_N, \\
& x_i^{\mathrm{c,C}} \in \mathcal{P}(x_i^{\mathrm{c,opp,j}}, \theta_j), \; i = 0, \dots, N-1, \\
& \qquad j = 1, \dots, n_{\mathrm{opp}}.
\end{aligned} \tag{7}
$$

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

Results

Setup:

- Simulation on randomized scenarios with three obstacles to overtake
- acados, 6s horizon length, 50 discr. points
- Two scenarios:
  - Truck-sized obstacles
  - Car-sized obstacles
- 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

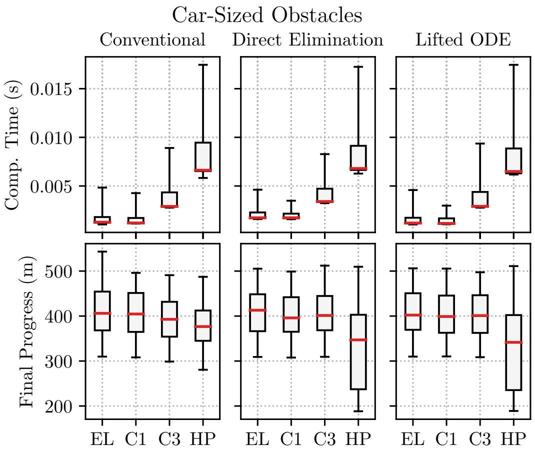


Figure: Box-plot comparison of the NMPC solution timings for each real-time iteration and the final progress after 20 seconds for different obstacle formulations for car-sized vehicles.

Results

truck-sized

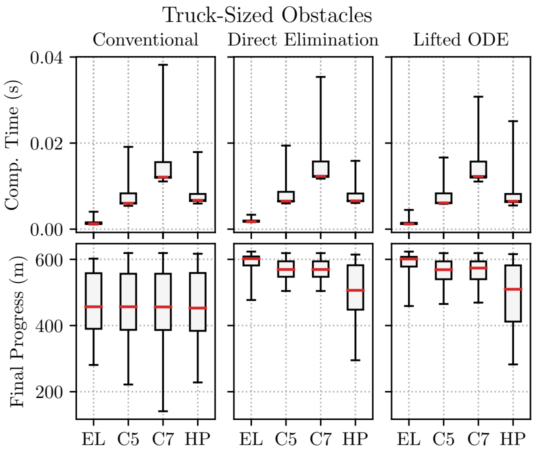
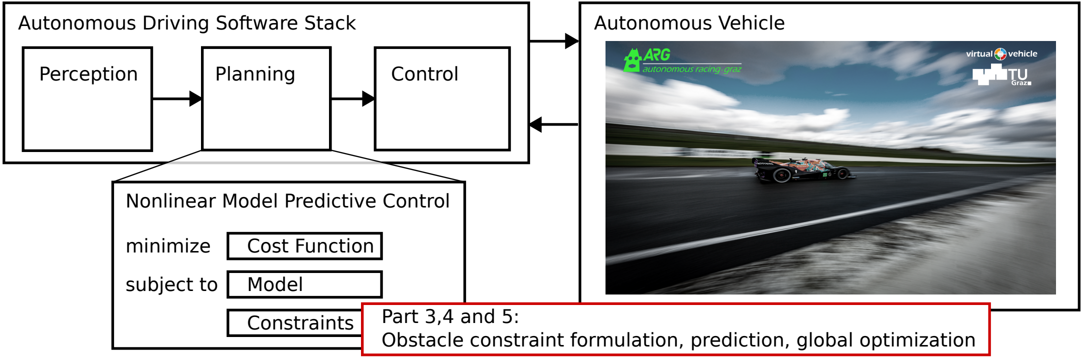


Figure: Box-plot comparison of the NMPC solution timings for each real-time iteration and the final progress after 20 seconds for different obstacle formulations for truck-sized vehicles.

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.

Part 4: Obstacle prediction[5]



[5] 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.

Part 4: Obstacle prediction

General Goal: Prediction of Opponents

- In AD, a core challenge is the prediction of other agents
- Algorithms differ related to the availability of data

For autonomous racing

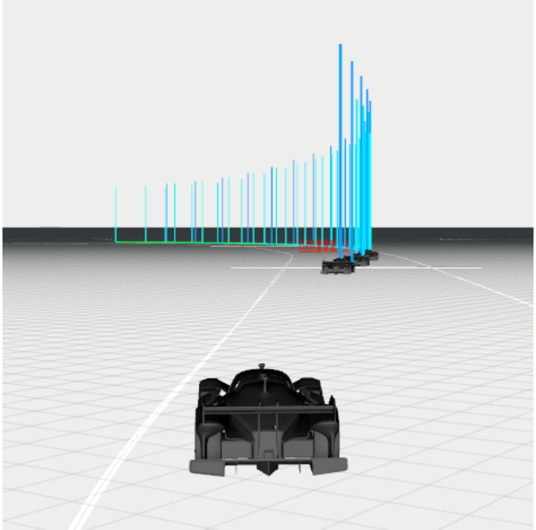
- Lack of huge data sets
- Some prior knowledge available: coarse models, racing objective
- An extensive online system identification is impossible

Our goal

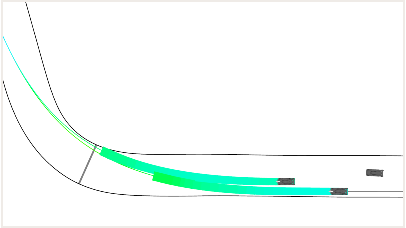
- Fast prediction within Milliseconds and adaption to observed data

Part 4: Obstacle prediction

General Goal: Prediction of Opponents



- 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)

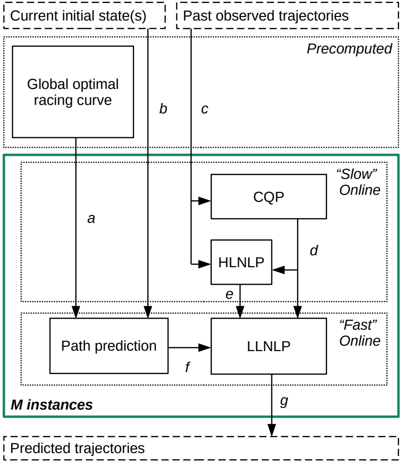


Advantages:

- A physically explainable prediction
- A good prediction even without any data
- Adaptive algorithm that improves with amount of data
- Fast improvement

Disadvantages:

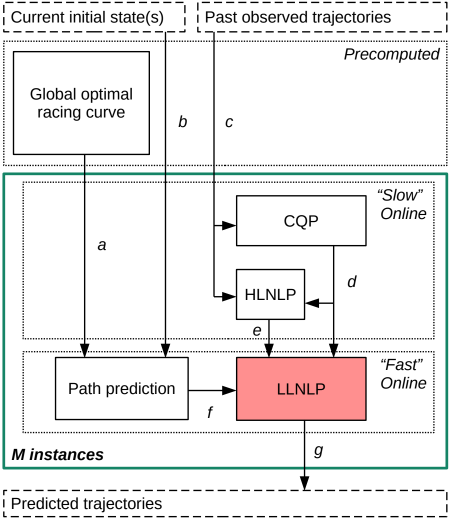
- We ignore interactive behavior of any kind
- Structural bias even with an infinite amount of data



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)



The Prediction Algorithm

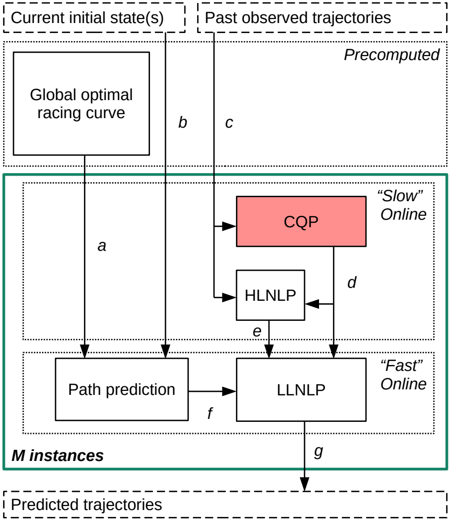
Low-Level Program for Trajectory Prediction (LLNLP)

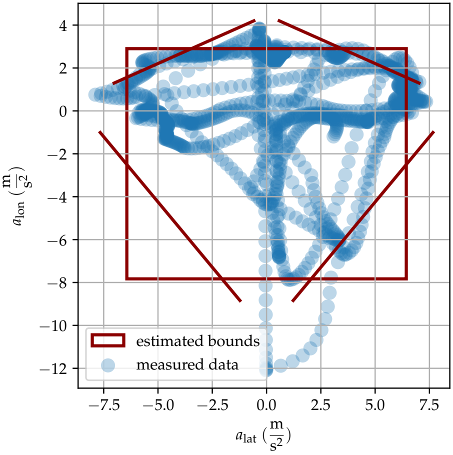
- Nonlinear program to maximize progress (x_N) along given path
- Weights Q, R, q_N estimated by HLNLP
- Acceleration constraints $h_a(x_k, \bar{\kappa}, a_{\max})$ estimated by CQP

$$
\begin{aligned}
\min_{\substack{x_0,\ldots,x_N,\\ U_0,\ldots,U_{N-1}\\ s_0,\ldots,s_N}} \quad & \sum_{k=0}^{N-1} \|x_k - x_k^{\mathrm{r}}\|_{2,Q}^2 + \|U_k - U_k^{\mathrm{r}}\|_{2,R}^2 + {q_N}^\top x_N + \sum_{k=0}^{N} \alpha_1 \mathbf{1}^\top s_{\mathrm{LL},k} + \alpha_2 \|s_{\mathrm{LL},k}\|_2^2 \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& x_{k+1} = F(x_k, U_k, \Delta t), \qquad k = 0,\ldots,N-1 \\
& \underline{x} \preccurlyeq x_k \preccurlyeq \overline{x} \\
& 0 \preccurlyeq h_a(x_k, \bar{\kappa}, a_{\max}) + s_{\mathrm{LL},k} \\
& 0 \preccurlyeq s_{\mathrm{LL},k}, \qquad k = 0,\ldots,N,
\end{aligned}
\tag{8}
$$

The Prediction Algorithm

Quadratic Program for Constraint Estimation

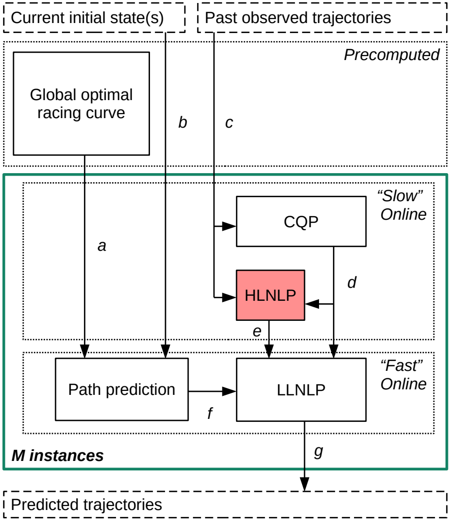




- Constraints are estimated separatly from the weights
- Symmetric polytope with 8 bounds (5 independent) fitted to data
- Iterative QP, with previously estimated value as "arrival term" (moving horizon estimation)

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

$$\begin{aligned} \min_{X,U,w} \quad & \sum_{k=1}^{N_T-1} \|x_k - \bar{x}_k\|^2_{2,Q_k} + \|w - \hat{w}\|^2_{2,P^{-1}} \\ \text{s.t.} \quad & X, U \in \operatorname{argmin} P_{\mathrm{LL}}(w, \bar{x}_0, \bar{\kappa}, a_{\max}) \\ & w \succcurlyeq 0 \end{aligned} \tag{9}$$

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

Results
Setup

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:

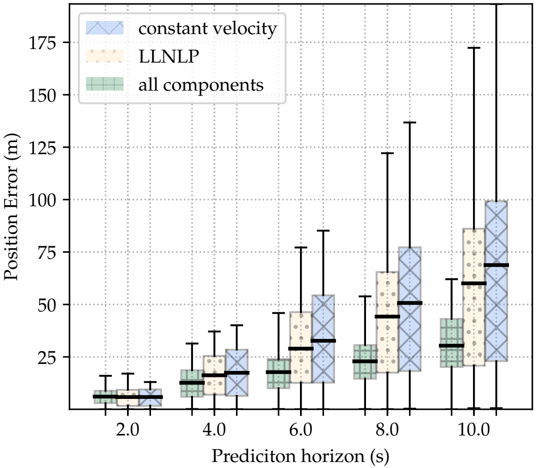
- Hardware: HP Elitebook, Intel Core i7-8550 CPU (1.8 GHz) and Nvidia Drive PX2
- The used frequency for the synchronous LLNLP was 10 Hz
- The HLNLP and CQP ran asynchronously
- 200 seconds until HLNLP converged

Table: Solver and timing statistics

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(QP)	2748	91	0.2

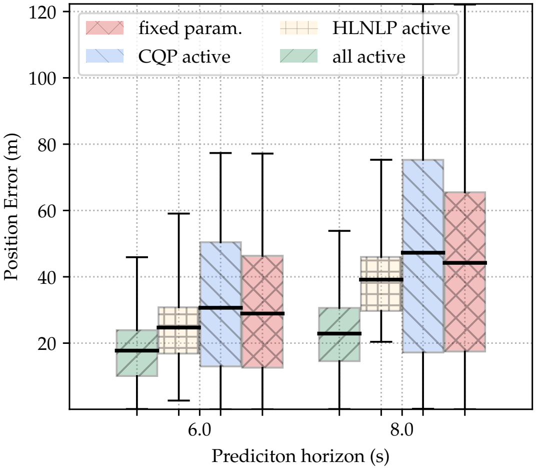
Results

Final Prediction Errors by Prediction Horizon (converged)

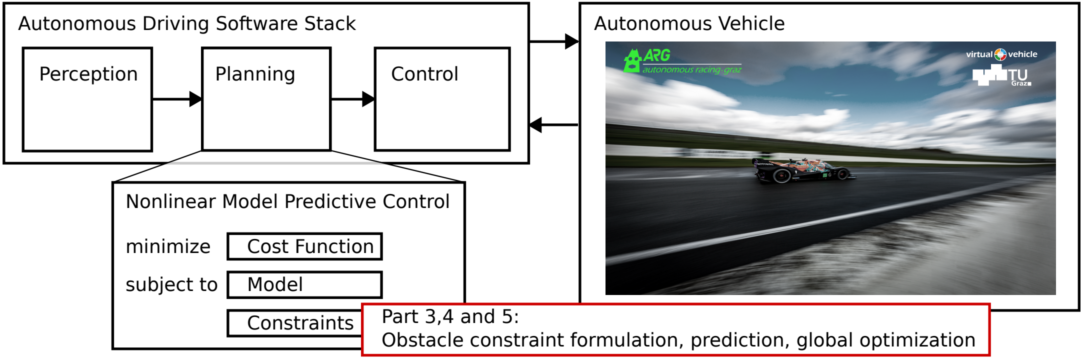


Results

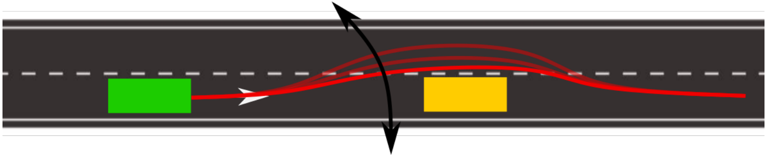
Final Prediction Errors by Used Components (converged)



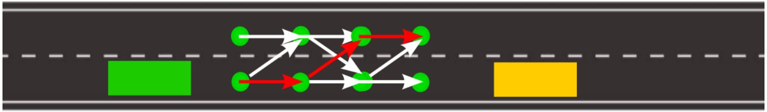
Part 5: Global optimization for obstacle avoidance



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



- Alternative 1: search in a discrete space

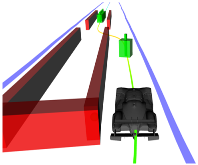


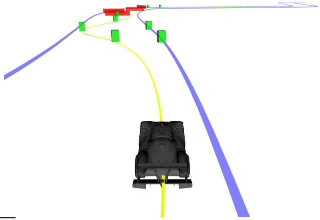
- Alternative 2 : search in a mixed continuous-discrete space **mixed integer optimization**

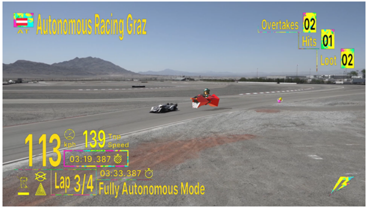
Global optimization for obstacle avoidance

Overview

- Mixed-integer optimization in racing with static obstacles and rewards[6]: Solving simplified problem first → shifting road boundaries accordingly
- Learning-based mixed-integer optimization for multi-lane traffic Expert MIQP formulation that solves problems offline. Learning the binary variables. Predicting the binary variables and solving the remaining QP online *(submitted)*
- Efficient formulation to obtain small MIQP that can be solved online within Milliseconds *(soon submitted)*

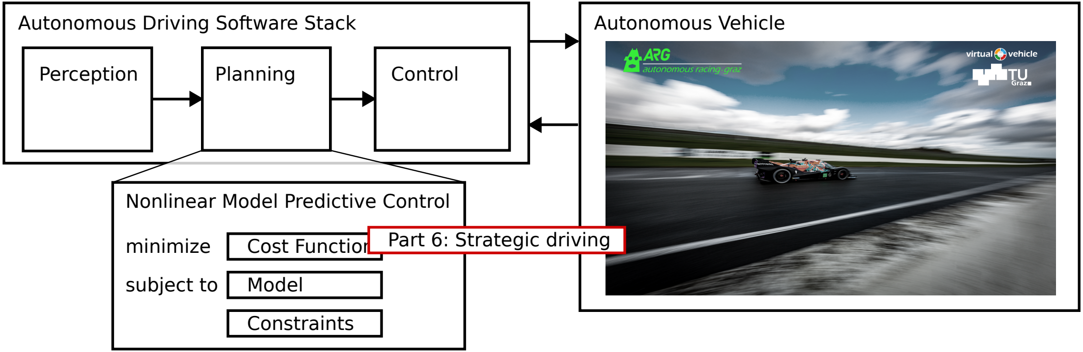






[6] Rudolf Reiter et al. "Mixed-integer optimization-based planning for autonomous racing with obstacles and rewards". In: *IFAC-PapersOnLine* 54.6 (2021). 7th IFAC Conference on Nonlinear Model Predictive Control NMPC 2021, pp. 99–106. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2021.08.530. URL: https://www.sciencedirect.com/science/article/pii/S2405896321013057.

Part 6: Strategic motion planning[7]



[7] 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.

- Task: Strategic planning and control of autonomous race cars $\rightarrow$ blocking of other agents, efficient overtaking

- Idea 1: use **only** optimization-based control (NMPC)
- Problem: hard to define strategic decisions (bi-level problem)

- Idea 2: use **only** reinforcement learning
- Problem: can hardly account for safety, many data needed for simple maneuvers

- Idea: Combine reinforcement learning and NMPC hierarchically
- Questions: Improved performance over pure RL? Faster learning? Meaningful learning? Guaranteed safety?

Part 6: Strategic motion planning

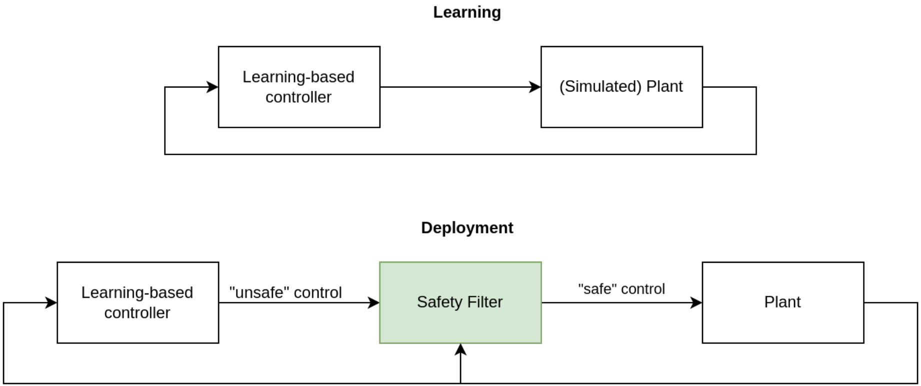
Outline

1. Relation to the safety filter
2. Proposed architecture
3. NMPC Formulation
4. RL Formulation
5. HILEPP Algorithm
6. Evaluation
7. Conclusion and Discussion

Relation to the safety filter[8]

Original

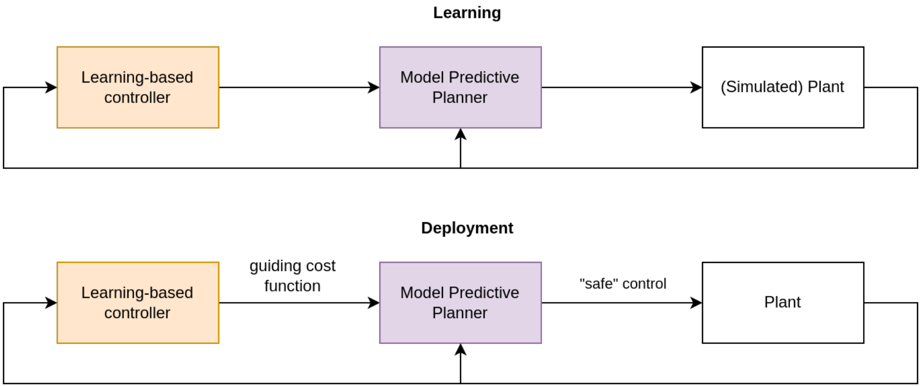
The safety filter uses an NLP to project controls onto safe sets



[8] 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[9]

Our approach



[9]Wabersich and Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems".

Relation to the safety filter[10]

Our approach

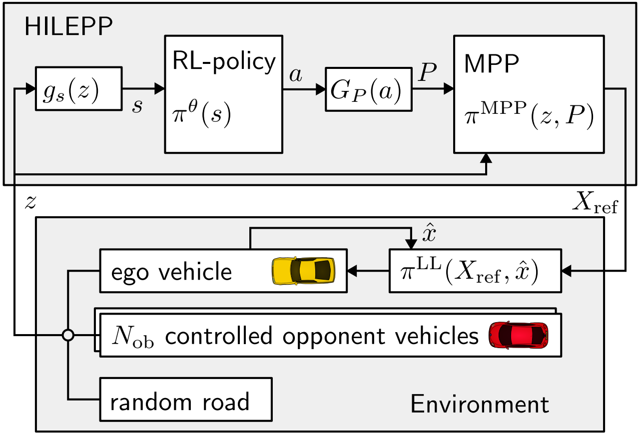
Safety Filter

$$\begin{aligned} \min_{X,U} \quad & \|u_0 - \bar{a}\|_R^2 \\ \text{s.t.} \quad & x_0 = \bar{x}_0, \quad x_N \in \mathcal{S}^{\mathrm{t}} \\ & x_{i+1} = F(x_i, u_i), \quad i = 0, \dots, N-1 \\ & x_i \in \mathcal{X}, u_i \in \mathcal{U}, \quad i = 0, \dots, N-1 \end{aligned} \tag{10}$$

HILEPP (ours)

$$\begin{aligned} \min_{X,U} \quad & L(X, U, a) \\ \text{s.t.} \quad & x_0 = \hat{x}_0, \quad x_N \in \mathcal{S}^{\mathrm{t}} \\ & x_{i+1} = F(x_i, u_i), \quad i = 0, \dots, N-1 \\ & x_i \in \mathcal{X}, u_i \in \mathcal{U}, \quad i = 0, \dots, N-1, \end{aligned} \tag{11}$$

[10] Wabersich and Zeilinger, "A predictive safety filter for learning-based control of constrained nonlinear dynamical systems".



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. States $x^\top = [\zeta, n, \alpha, v, \delta]$
- Obstacle avoidance with ellipses - circles[11]
- 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:

$$G_P(a) : a \rightarrow \Big(\xi_{\text{ref},0}(a), \ldots, \xi_{\text{ref},N}(a), Q_{\text{w}}(a)\Big) \tag{12}$$

$$\xi_{\text{ref},k}(a) = [0 \quad n \quad 0 \quad v_x \quad 0]^\top \in \mathbb{R}^{n_x} \tag{13}$$

$$Q_{\text{w}}(a) = \text{diag}([0 \quad w_n \quad 0 \quad w_v \quad 0]) \tag{14}$$

[11] Rudolf Reiter et al. *Frenet-Cartesian Model Representations for Automotive Obstacle Avoidance within Nonlinear MPC*. 2023. arXiv: 2212.13115 [eess.SY].

Cost parameterization through RL actions:

$$G_P(a) : a \rightarrow \Big(\xi_{\mathrm{ref},0}(a), \ldots, \xi_{\mathrm{ref},N}(a), Q_{\mathrm{w}}(a)\Big) \tag{15}$$

$$\xi_{\mathrm{ref},k}(a) = [0 \quad n \quad 0 \quad v_x \quad 0]^\top \in \mathcal{R}^{n_x} \tag{16}$$

$$Q_{\mathrm{w}}(a) = \mathrm{diag}([0 \quad w_n \quad 0 \quad w_v \quad 0]) \tag{17}$$

NMPC (MPP) parameterized cost:

$$\begin{aligned} L(X, U, a, \Xi) = \sum_{k=0}^{N-1} \|x_k - \xi_{\mathrm{ref},k}(a)\|^2_{Q_{\mathrm{w}}(a)} + \|u_k\|^2_R \\ + \|x_N - \xi_{\mathrm{ref},N}(a)\|^2_{Q^t} + \sum_{k=0}^{N} \|\sigma_k\|^2_{Q_{\sigma,2}} + |q_{\sigma,1}^\top \sigma_k|. \end{aligned} \tag{18}$$

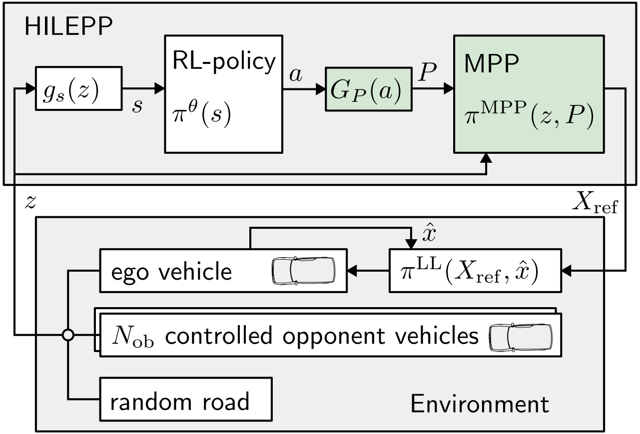
We compare two action vectors (with or without setting weights):

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

The NLP that is solved for each MPP iteration, can be written as:

$$
\begin{aligned}
\min_{X,U,\Xi} \quad & L(X,U,a,\Xi) \\
\text{s.t.} \quad & x_0 = \hat{x}, \quad \Xi \geq 0, \quad x_N \in \mathcal{S}^{\mathrm{t}} \\
& x_{i+1} = F(x_i, u_i) && i = 0,\dots,N-1 \\
& U_i \in B_u, && i = 0,\dots,N-1 \\
& x_i \in B_x(\sigma_k) \cap B_{\mathrm{lat}}(\sigma_k) && i = 0,\dots,N \\
& x_i \in B_{\mathrm{ob}}(p_i^{\mathrm{ob},j}, \Sigma_i^{\mathrm{ob},j}, \sigma_k) && i = 0,\dots,N \\
& && j = 0,\dots,N_{\mathrm{ob}},
\end{aligned}
\tag{19}
$$

using states X, controls U, slacks Ξ, dynamic integration function $F(\cdot)$, state and acceleration constraints $B_x(\cdot), B_{\mathrm{lat}}(\cdot)$ and obstacle constraints $B_{\mathrm{ob}}(p_i^{\mathrm{ob},j}, \Sigma_i^{\mathrm{ob},j}, \sigma_k)$, depending on prediction $p_i^{\mathrm{ob},j}, \Sigma_i^{\mathrm{ob},j}$ for each obstacle.


HILEPP
RL-policy
$\pi^{\theta}(s)$
$g_s(z)$
s
a
$G_P(a)$
P
MPP
$\pi^{\mathrm{MPP}}(z, P)$
z
X_{ref}
$\hat{x}$
ego vehicle
$\pi^{\mathrm{LL}}(X_{\mathrm{ref}}, \hat{x})$
N_{ob} controlled opponent vehicles
random road
Environment

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 *actor critic policy gradient* algorithm[12] 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 $s_{\mathrm{ob}_i} = [\zeta_{\mathrm{ob}_i} - \zeta, n_{\mathrm{ob}_i}, v_{\mathrm{ob}_i}, \alpha_{\mathrm{ob}_i}]^\top$

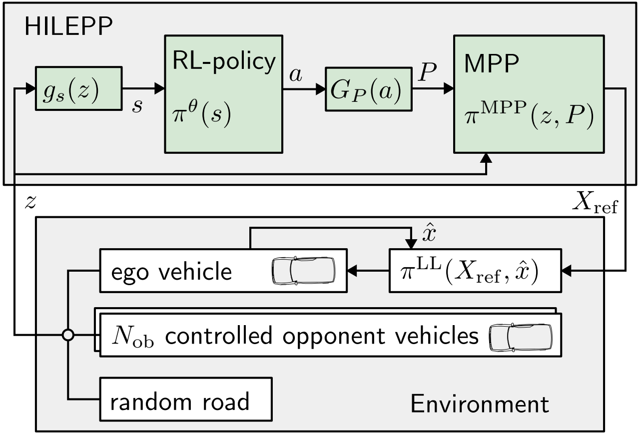
$$s_k = g_s(z_k) = [\kappa(\zeta + d_i), \ldots, \kappa(\zeta + d_N), s^\top, s_{\mathrm{ob}_1}^\top, \ldots, s_{\mathrm{ob}_N}^\top]^\top \tag{20}$$

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

$$R(s, a) = \frac{\dot{s}}{200} + \sum_{i=1}^{N_{\mathrm{ob}}} 1_{\zeta_k > \zeta_k^{\mathrm{ob}_i}} \tag{21}$$

[12] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

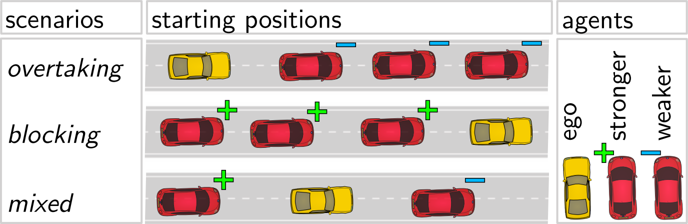
Overview
Architecture



Evaluation

Setup

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

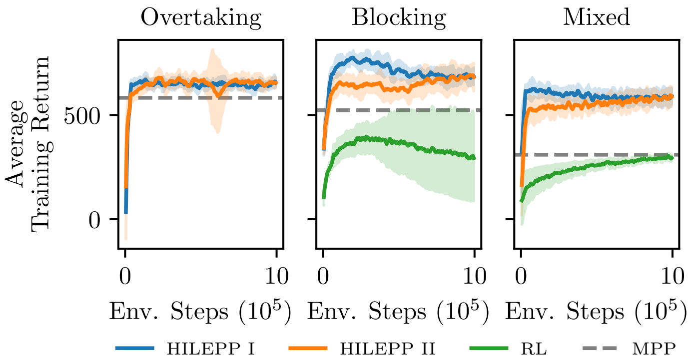


- Comparison of
  - MPP
  - RL
  - HILEPP-I (only reference states)
  - HILEPP-II (reference states and weights)

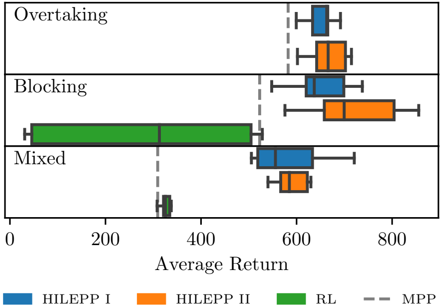
Evaluation

Training

- pure RL learns slow
- HILEPP very sample efficient
- HILEPP-I learns quicker than HILEPP-II



- pure RL *struggled* to keep up even with MPP
- overtaking does not require much strategy → MPP compared to HILEPP smaller
- 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.

Evaluation

Examples

- →Play-scenario-blocking
- →Play-scenario-mixed
- →Play-scenario-overtake

Conclusion and discussion

- Nonlinear model predictive control is a powerful framework for motion planning in autonomous driving
- Additional performance obtained by
  - Mixed-integer optimization
  - Inverse optimal control
  - Reinforcement learning
- Orthogonal approaches exist
  - Discrete search space $\rightarrow$ graph search, tree search
  - End-to-end learning

Pros

- Using existing powerful NLP solvers
- Easy separation and specification of task (cost, model, constraints)
- Optimal solutions
- Interpretability
- Safety certificate
- Extendability
- Adaptability

Cons

- No bound on computation time
- No guarantees for global optimum
- No guarantees to even converge to a stationary point
- Interpretability

Thanks for the help of all supervisors, colleagues and friends!

Thank you for your attention!