Autonomous Racing by Model Predictive Contouring Control

Alexander Liniger

Alexander Domahidi, John Lygeros, Manfred Morari

Berkeley April 2015

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Automatik
Automatic Control Laboratory

Autonomous Driving

- DARPA Grand & Urban Challenge
- Google: >1.1 Mio km driven as of 4/14
- Tesla: >90% autonomous in 2016
- Mercedes: autonomous E & S-class
- BMW: autonomous drift with 235i
- ... and many others.

Autonomous Racing

- Autonomous Racing Audi TTS (Stanford University):

No online path planning / obstacle avoidance / overtaking

ORCA Autonomous Racing

Goal: Autonomous racing with obstacle avoidance

Outline

- Introduction
- Hardware Setup
- Car Modeling
- Model Predictive Contouring Control
- Obstacle Avoidance
- Solution Approach
- Results
- Current/Future Work

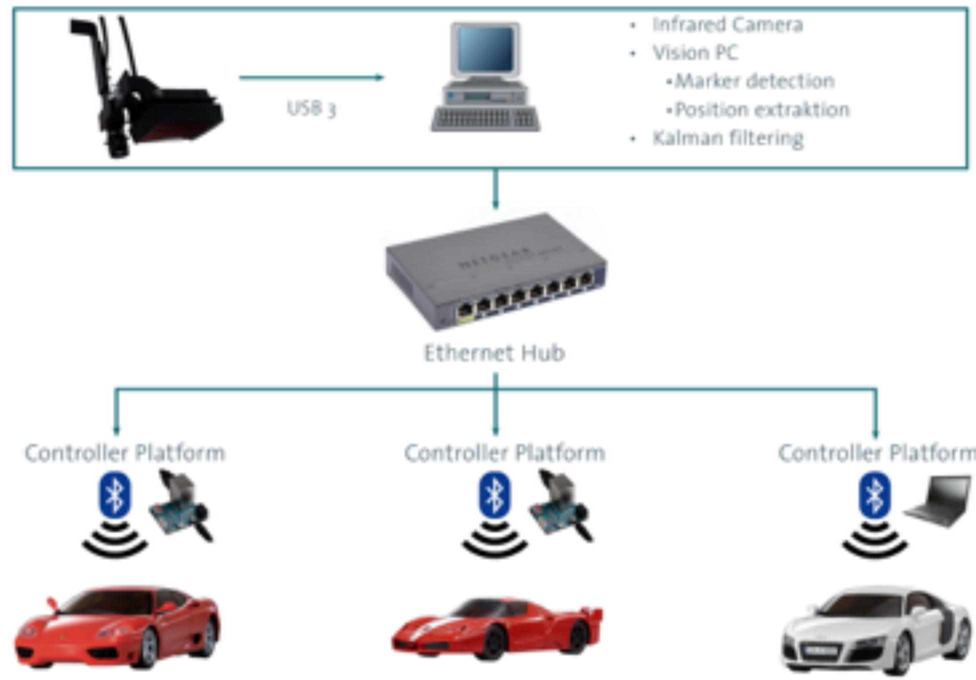
The ORCA Race Car Setup

- Built by ETH students from standard components using Kyosho dnano 1:43 RC cars

- Multiple control boards with RF connection to embedded car

- Embedded board inside the car:
  - ARM M4 microcontroller
  - Gyro and accelerometer
  - H-bridges for motors
  - Voltage & current measurement
  - Comms (Bluetooth)

http://control.ee.ethz.ch/~racing/



Camera System

- Infrared based vision system
- PointGrey Flea3
  - 100 fps
  - resolution ~3.5mm

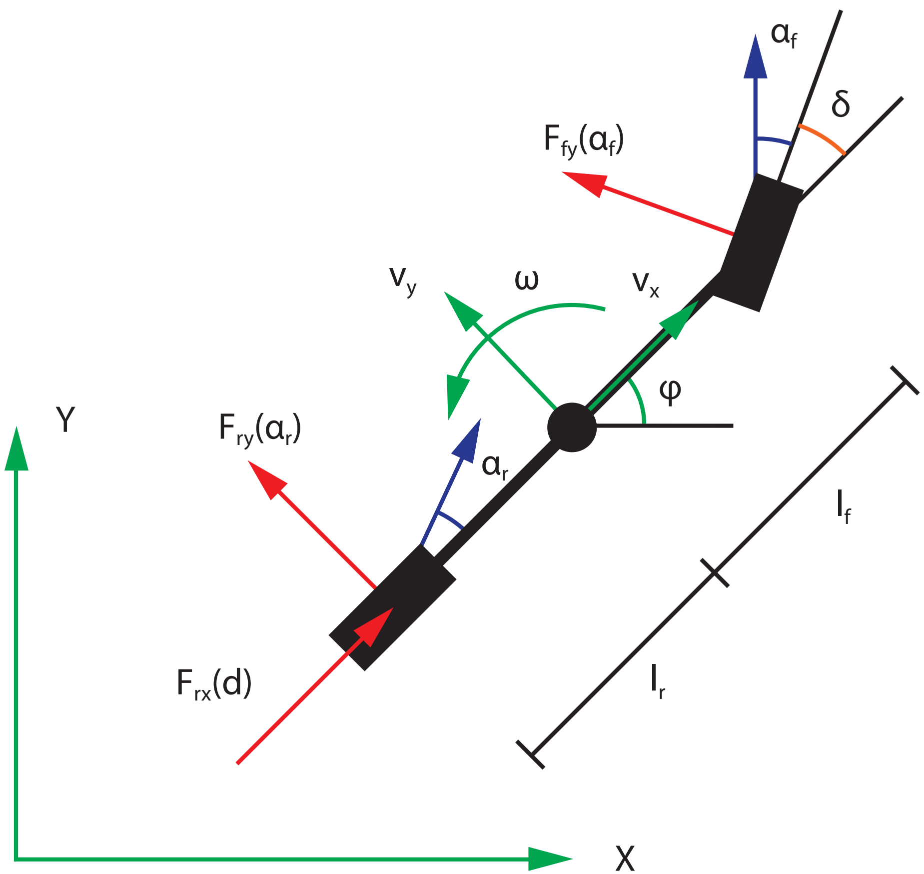
- 7 unique marker patterns

Outline

- Introduction
- Hardware Setup
- Car Modeling
- Model Predictive Contouring Control
- Obstacle Avoidance
- Solution Approach
- Results
- Current/Future Work

Car Model

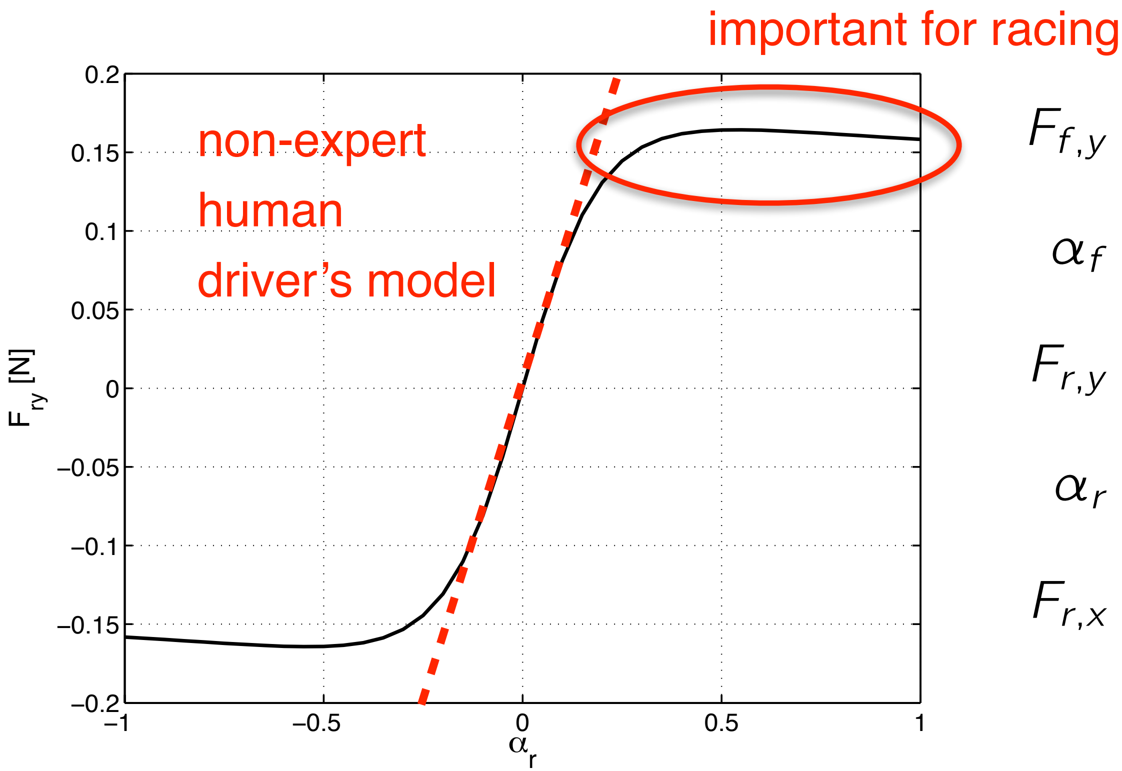
- Bicycle model with generic tire forces



$$\dot{X} = v_x \cos(\varphi) - v_y \sin(\varphi)$$
$$\dot{Y} = v_x \sin(\varphi) + v_y \cos(\varphi)$$
$$\dot{\varphi} = \omega$$
$$\dot{v}_x = \frac{1}{m}(F_{r,x} - F_{f,y} \sin\delta + m v_y \omega)$$
$$\dot{v}_y = \frac{1}{m}(F_{r,y} + F_{f,y} \cos\delta - m v_x \omega)$$
$$\dot{\omega} = \frac{1}{I_z}(F_{f,y} l_f \cos\delta - F_{r,y} l_r)$$

Tire Force Model

- Pacejka's "magic formula"



$$F_{f,y} = D_f \sin(C_f \arctan(B_f \alpha_f))$$

$$\alpha_f = -\arctan\left(\frac{\omega l_f + v_y}{v_x}\right) + \delta$$

$$F_{r,y} = D_r \sin(C_r \arctan(B_r \alpha_r))$$

$$\alpha_r = \arctan\left(\frac{\omega l_r - v_y}{v_x}\right)$$

$$F_{r,x} = (C_{m1} - C_{m2} v_x) d - C_r - C_d v_x^2$$

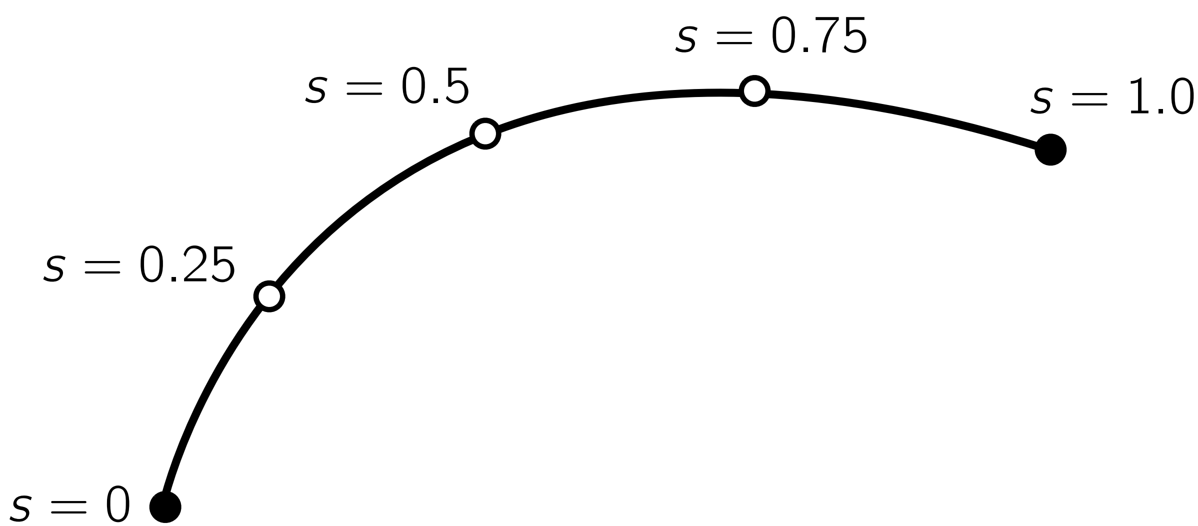
- Car inputs: duty cycle for DC motor d, steering angle δ
- Coefficients identified in various experiments

Outline

- Introduction
- Hardware Setup
- Car Modeling
- Model Predictive Contouring Control
- Obstacle Avoidance
- Solution Approach
- Results
- Current/Future Work

Path Parameterization

- Introduce variable $s \in [0, 1]$ denoting the normalized arc length of path:



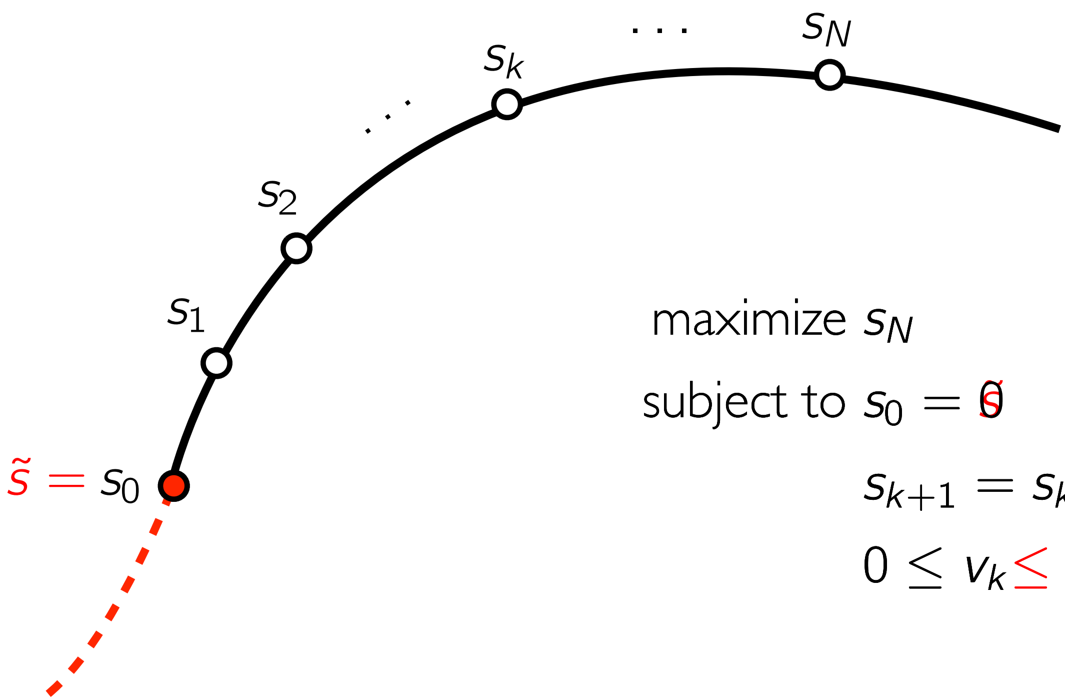
- Write discrete-time dynamics for arc length:

$$s_{k+1} = s_k + v_k \ , \quad v_k \geq 0$$

- v_k is (constrained) input

Minimum Time Objective

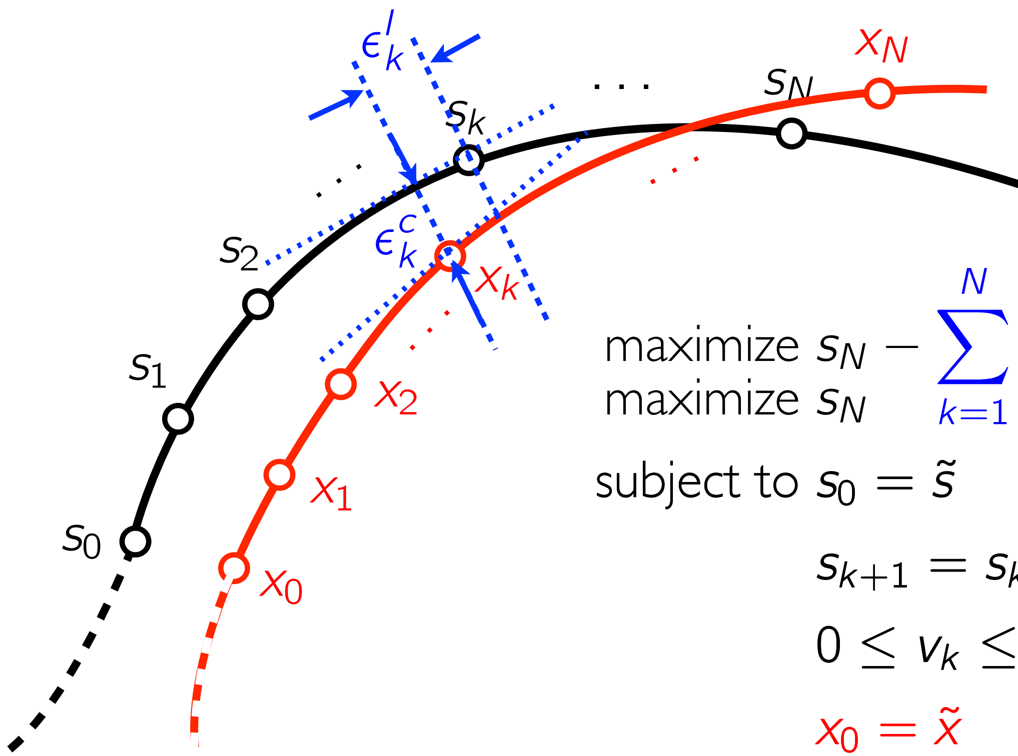
- maximize travelled arc in N time steps $\Leftrightarrow$ minimize time to travel s_N (assuming $s_N < 1$)



$$
\begin{aligned}
&\text{maximize } s_N \\
&\text{subject to } s_0 = \cancel{0}\,\tilde{s} \\
&\qquad s_{k+1} = s_k + v_k\,,\ k = 0, \dots, N-1 \\
&\qquad 0 \leq v_k \leq \bar{v}
\end{aligned}
$$

Linking the Physical System via Path Errors

- Adding physical dynamics:



$$\text{maximize } s_N - \sum_{k=1}^{N} \gamma_c \|\epsilon_k^c\|^2 + \gamma_l \|\epsilon_k^l\|^2$$

$$\text{maximize } s_N$$

$$\text{subject to } s_0 = \tilde{s}$$

$$s_{k+1} = s_k + v_k \,, k = 0, \ldots, N-1$$

$$0 \le v_k \le \bar{v}$$

$$x_0 = \tilde{x}$$

$$x_{k+1} = f(x_k, u_k) \,,\ k = 0, \ldots, N-1$$

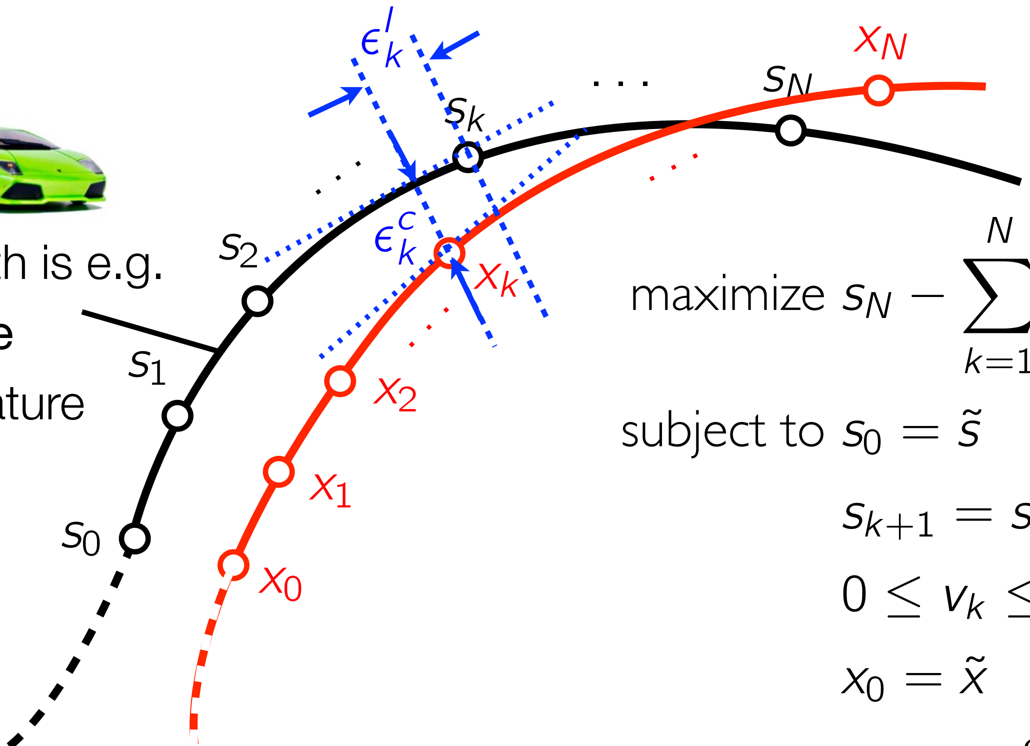
$$x_k \in \mathbb{X}_k, u_k \in \mathbb{U}_k$$

$$\epsilon_k^c = g(x_k, s_k) \,,\ \epsilon_k^l = h(x_k, s_k)$$

- Minimization of contouring errors ϵ_k^c and lag errors ϵ_k^l couples car dynamics to minimum time problem

Contouring Control Adapted for Racing

- **Machine tools:** want to follow a given path accurately
  - Contouring error ϵ_k^c must be low - γ_c must be high



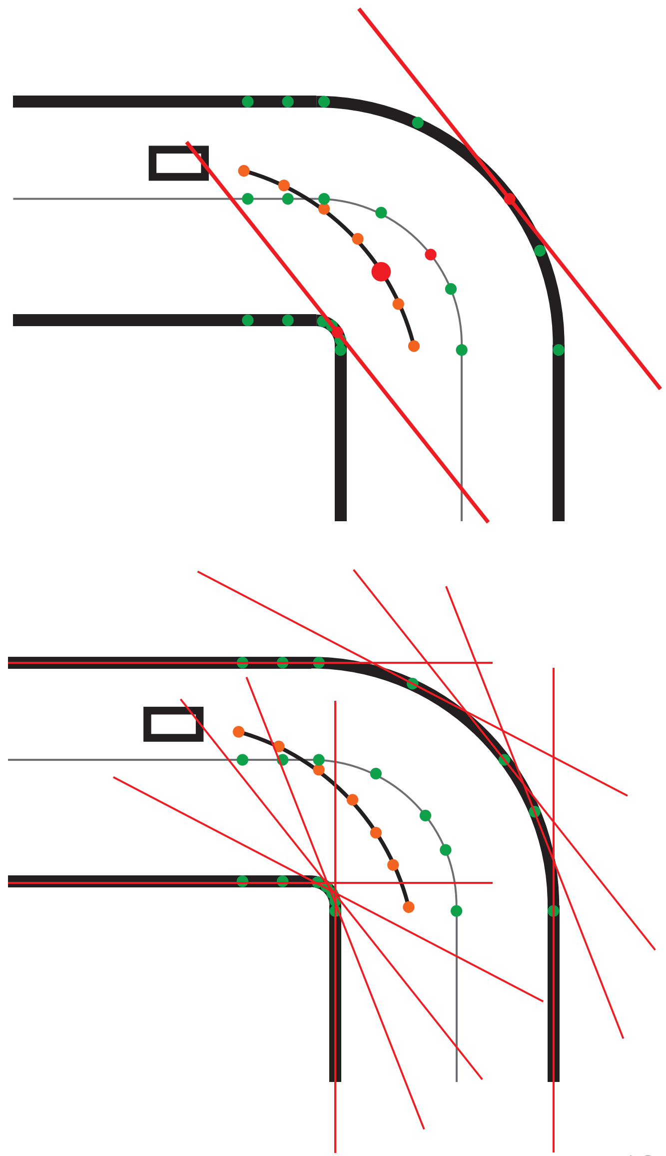
Racing: path is e.g.
- **center line**
- min. curvature path

$$
\begin{aligned}
\text{maximize } & s_N - \sum_{k=1}^{N} \gamma_c \|\epsilon_k^c\|^2 + \gamma_l \|\epsilon_k^l\|^2 \\
\text{subject to } & s_0 = \tilde{s} \\
& s_{k+1} = s_k + v_k \,,\ k = 0, \ldots, N-1 \\
& 0 \leq v_k \leq \bar{v} \\
& x_0 = \tilde{x} \\
& x_{k+1} = f(x_k, u_k)\,,\ \ k = 0, \ldots, N-1 \\
& x_k \in \mathbb{X}_k,\, u_k \in \mathbb{U}_k \\
& \epsilon_k^c = g(x_k, s_k)\,,\ \epsilon_k^l = h(x_k, s_k)
\end{aligned}
$$

- **Racing:** lateral deviation from path is OK - choose γ_c low

Race Track Constraints

- Track constraints in general non-convex
- Each point in the horizon is constrained within two half spaces (state constraints)
- Points of previous prediction are used to generate constraints (projection on track borders)

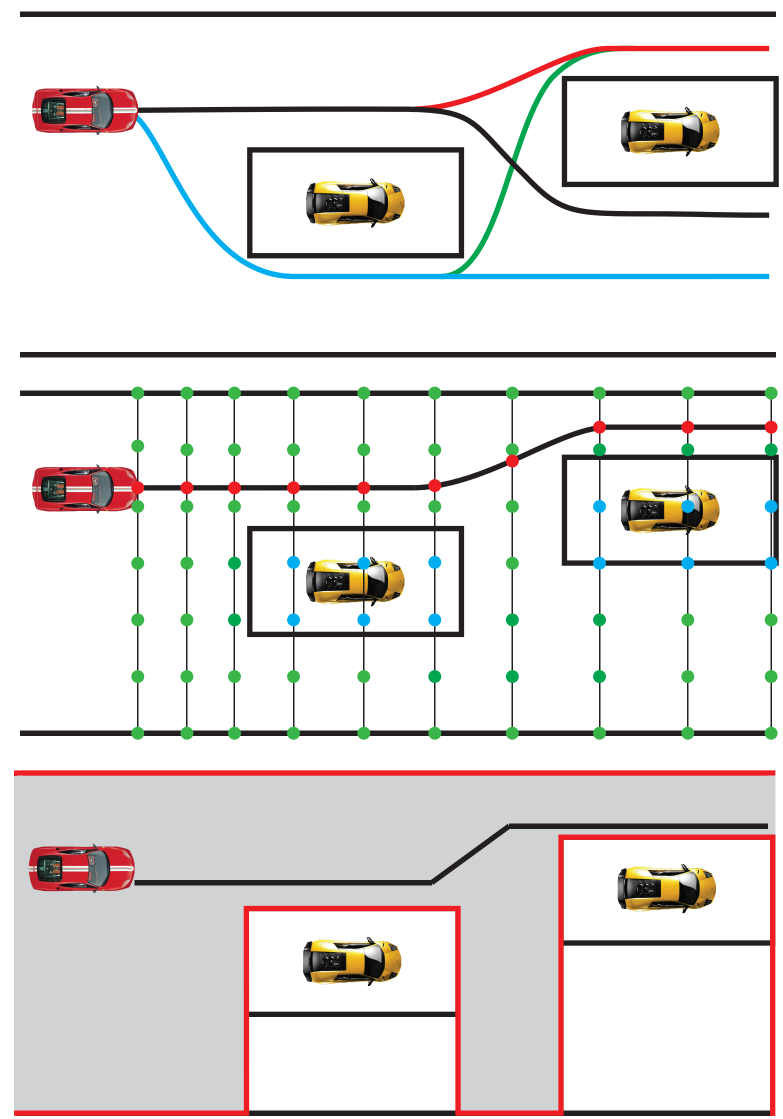


Outline

- Introduction
- Hardware Setup
- Car Modeling
- Model Predictive Contouring Control
- Obstacle Avoidance
- Solution Approach
- Results
- Current/Future Work

Obstacle Avoidance via Track Morphing

- Overtaking problem is non-convex
- 2-level solution approach:
  - high-level controller computes feasible "corridor" (convex set)
  - MPCC uses this corridor
- Dynamic programming computes optimal corridor avoiding opponents
  - Generate a spatial-temporal grid, based on the last MPCC iteration
  - minimize travelled distance and deviation from the last trajectory
- Morph feasible set for MPCC based on optimal corridor



Outline

- Introduction
- Hardware Setup
- Car Modeling
- Model Predictive Contouring Control
- Obstacle Avoidance
- Solution Approach
- Results
- Current/Future Work

Nonlinear MPC Implementation

- Real-time iteration/multiple shooting algorithm [Diehl 2002] with matrix exponential as integrator. In each time step,

1. Get current position $\tilde{x}$
2. Construct new state trajectory with $\tilde{x}$ and new $x_N \equiv x_{N-1}$
3. Generate new track constraints
4. **Linearize** continuous-time dynamics around trajectory
5. **Discretize** using matrix exponential
6. Solve **local convex approximation** (QP) $\longrightarrow$
7. Apply first input

$$
\begin{aligned}
\max\ & s_N - \sum_{k=1}^{N} \gamma_c \|\epsilon_k^c\|^2 + \gamma_l \|\epsilon_k^l\|^2 \\
\text{s.t. } & s_0 = \tilde{s}\,, \ \ x_0 = \tilde{x} \\
& s_{k+1} = s_k + v_k \\
& 0 \leq v_k \leq \bar{v}\,, \ \ x_k \in \mathbb{X}_k, u_k \in \mathbb{U}_k \\
& x_{k+1} = A_k x_k + B_k u_k + g_k \\
& \epsilon_k^c = E_k x_k + F_k s_k + f_k \\
& \epsilon_k^l = G_k x_k + H_k s_k + h_k
\end{aligned}
$$

- Can be shown to converge to local optimum
- Initialization: run iterations from above until convergence

Outline

- Introduction
- Hardware Setup
- Car Modeling
- Model Predictive Contouring Control
- Obstacle Avoidance
- Solution Approach
- Results
- Current/Future Work

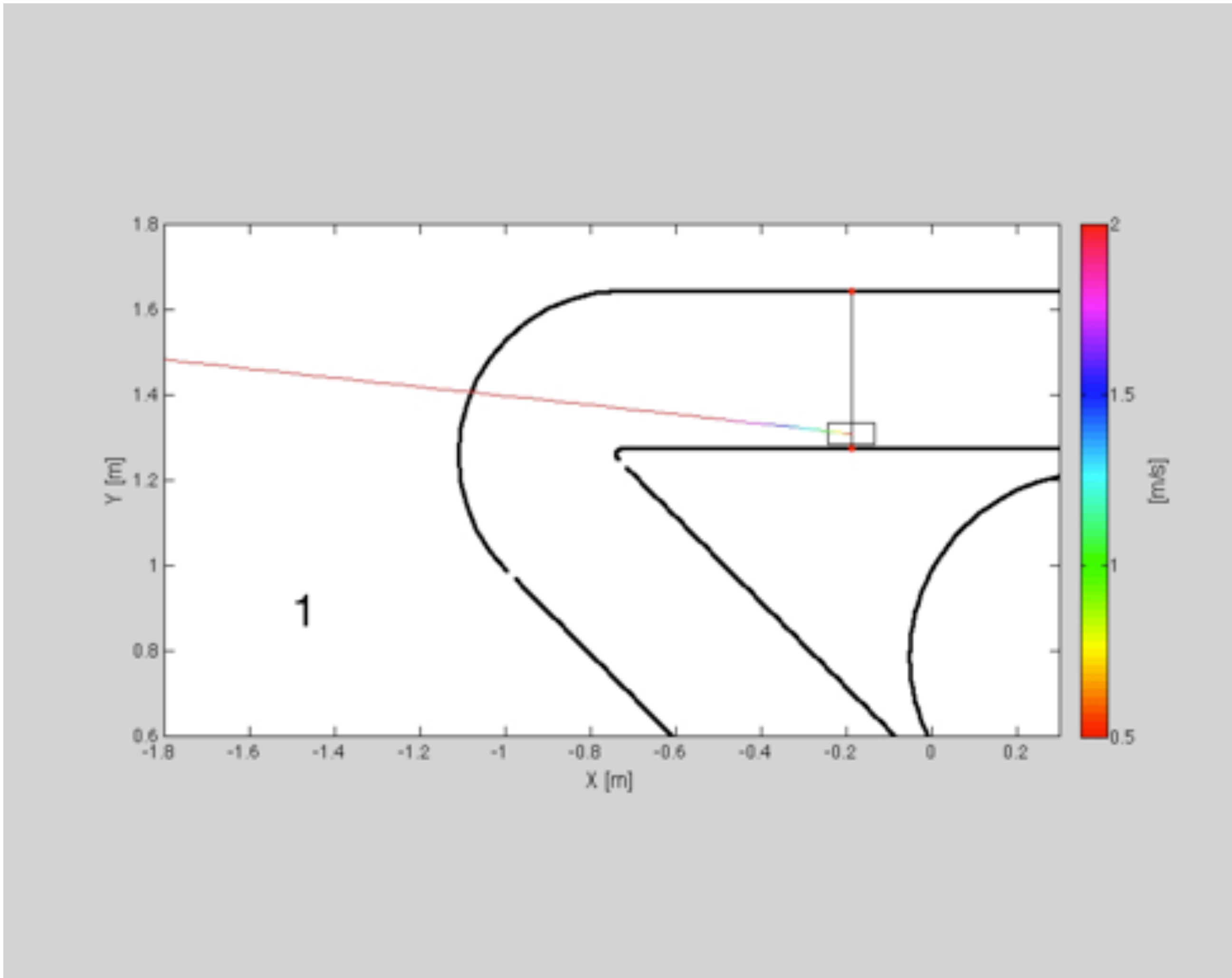
Computation Times

- Computation times in milliseconds on Exynos 5410 (ARM Cortex A15 @1.6 GHz)

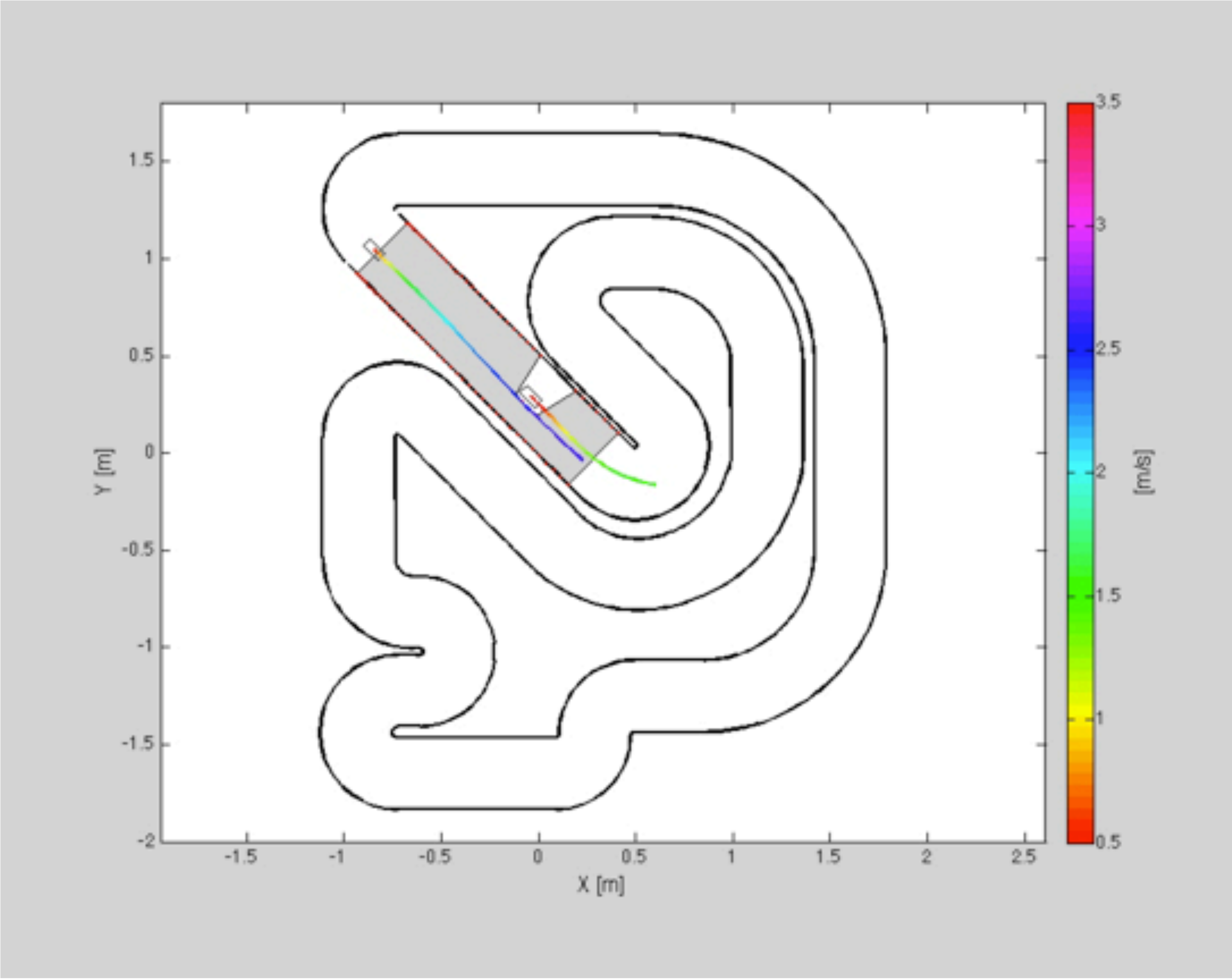
	Mean	Stdev	Max
Border Adjustment	0.7	0.24	1.05
QP Generation	2.34	0.37	2.79
QP with FORCES	14.43	1.40	21.46

- Sampling rate of 50 Hz possible (4.4% overtime)
- Problem dimensions:
  - Variables per stage: 13
  - Constraints per stage: 16
  - Horizon length: 40
- Total: QP with 520 variables, 640 constraints, LTV dynamics

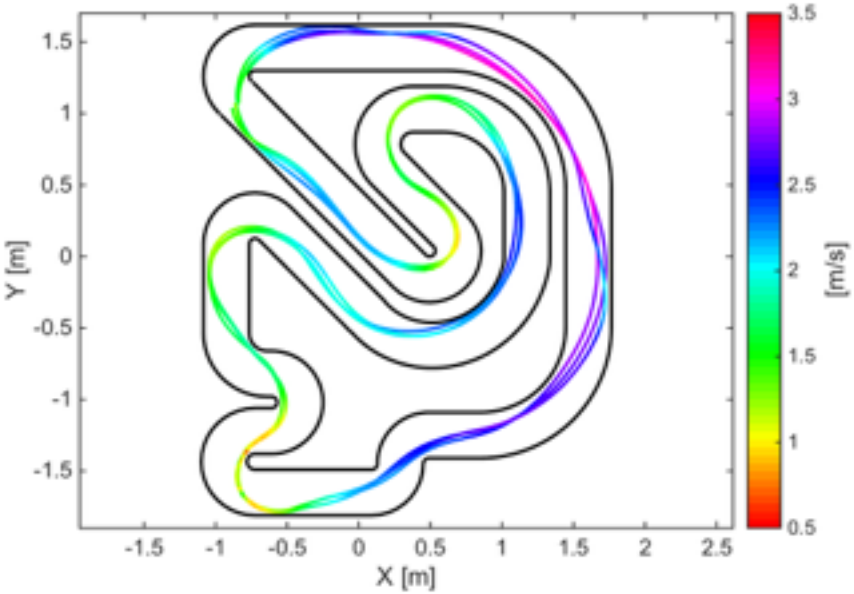
Initialization

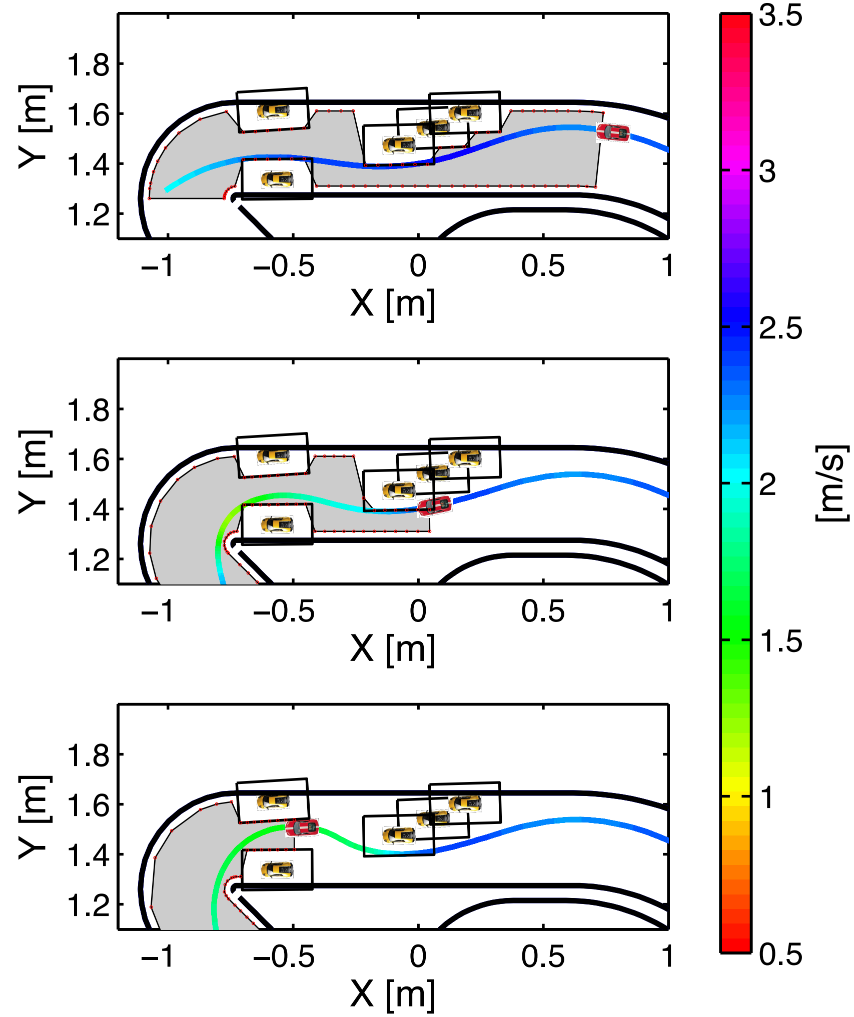


Closed-Loop Simulation

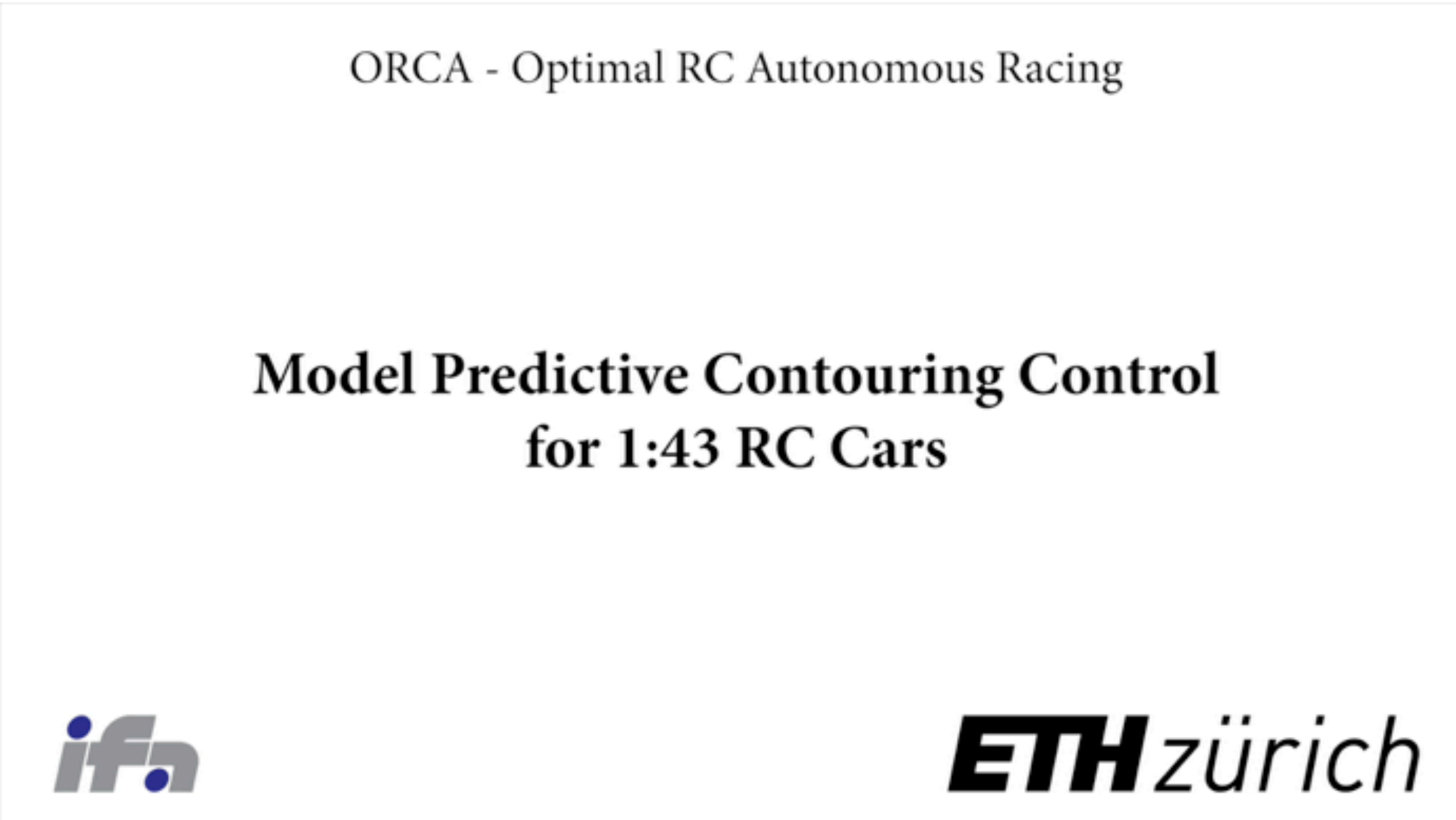


Closed Loop Results



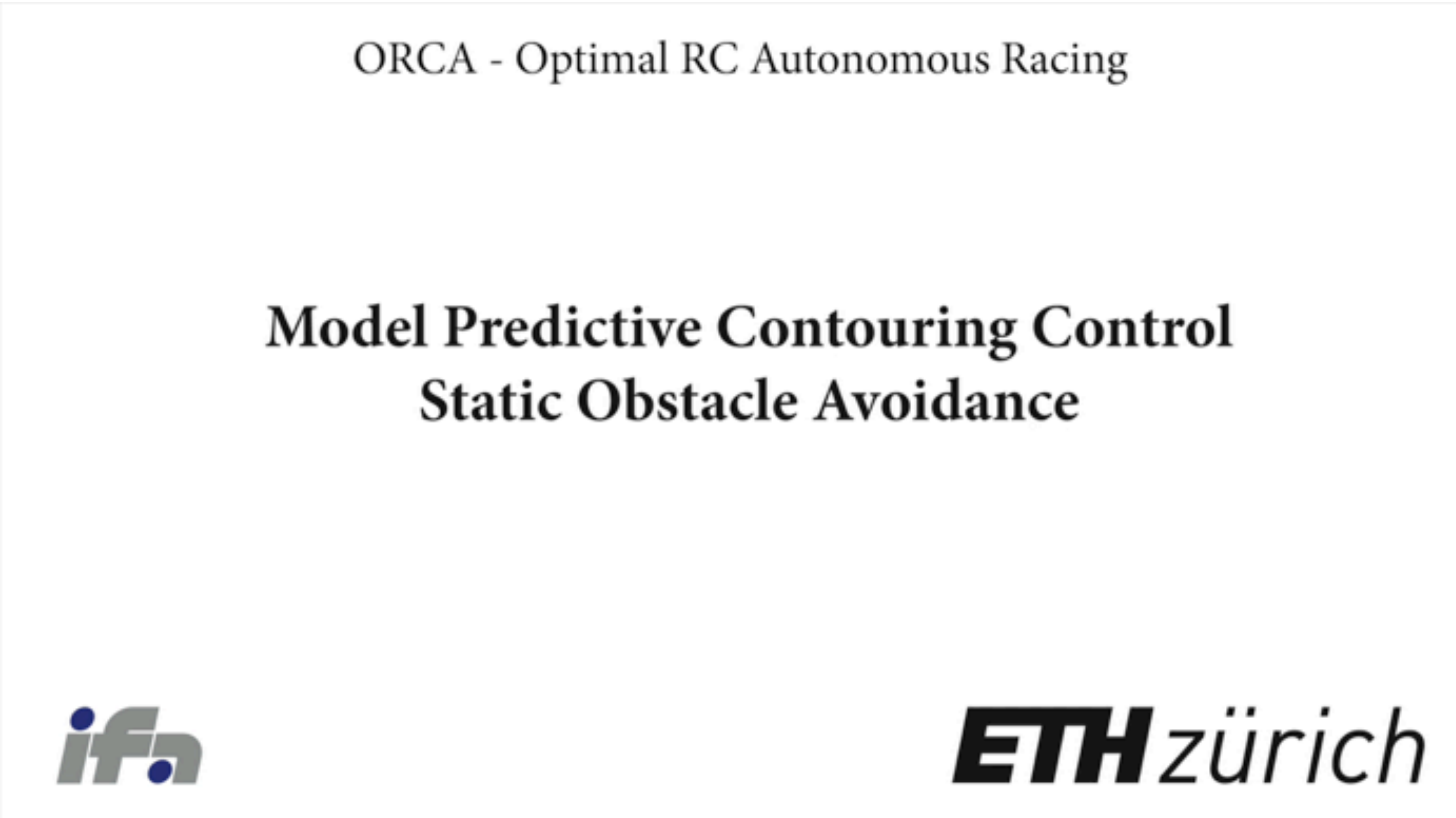


Autonomous RC Racing Using MPCC



https://www.youtube.com/watch?v=w46kkjKda9s

Obstacle Avoidance



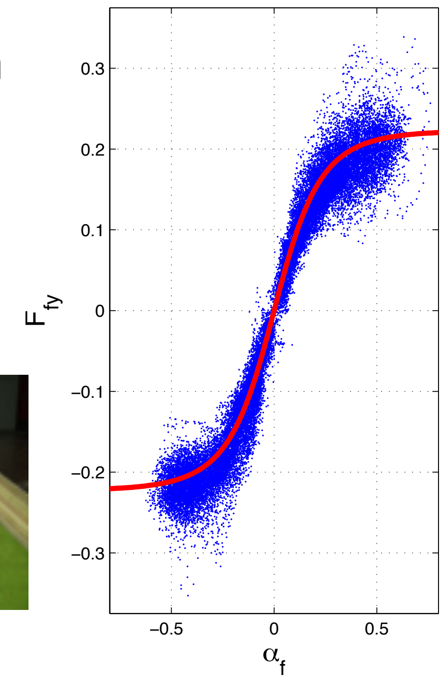
https://www.youtube.com/watch?v=JoHfJ6LEKVo

Outline

- Introduction
- Hardware Setup
- Car Modeling
- Model Predictive Contouring Control
- Obstacle Avoidance
- Solution Approach
- Results
- Current/Future Work

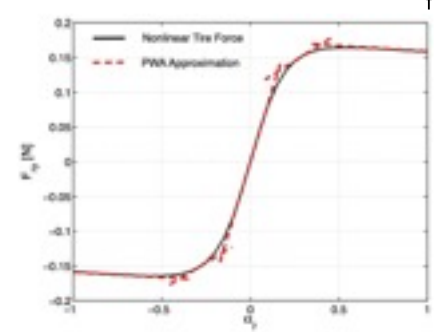
Additional Work using MPCC

- Scenario MPC
  - Additive error describing “any” kind of model mismatch
  - difference between prediction and measured state
  - “model-based” constraint tightening
  - open-loop control inputs are too conservative



- Driving Backwards
  - same model for driving backwards
  - a lot harder than expected

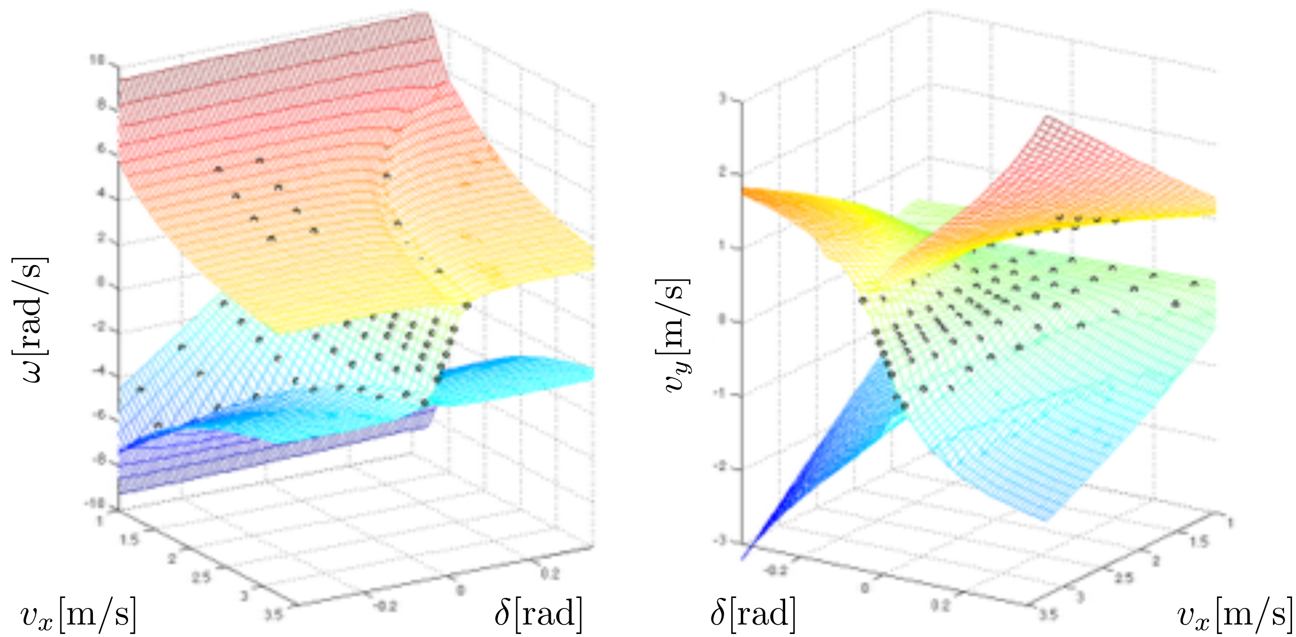
- Piecewise-affine tire forces (on going)
  - goal: using inverse optimization to solve problem efficiently



Hierarchical Control

1. **Path planning** based on motion primitives
2. **MPC** tracking optimal trajectory

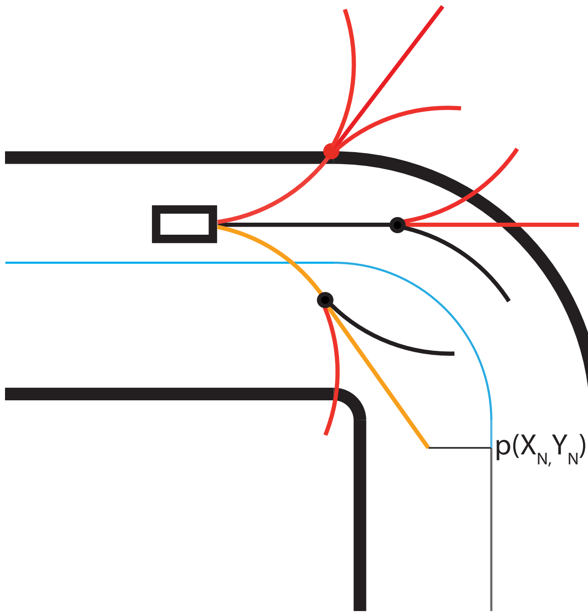
- Motion primitives / Constant velocity points:



- Directly linking trims if the transition is feasible for bicycle model

Hierarchical Control

- Splits at fixed sampling times



- Assumption:
  - new constant velocity can be reached immediately
- Model simplifies

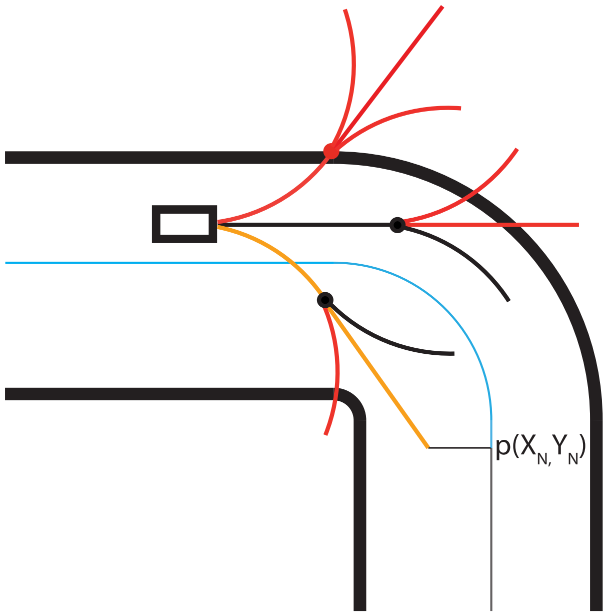
$$\dot{X} = \bar{v}_x(q)\cos(\varphi) - \bar{v}_y(q)\sin(\varphi)$$
$$\dot{Y} = \bar{v}_x(q)\sin(\varphi) + \bar{v}_y(q)\cos(\varphi)$$
$$\dot{\varphi} = \bar{\omega}(q)$$
$$\bar{v}(q) = [\bar{v}_x(q), \bar{v}_y(q), \bar{\omega}(q)]$$

$$p(X,Y) := \arg\min_{\theta}(X - X^{\text{cen}}(\theta))^2 + (Y - Y^{\text{cen}}(\theta))^2$$

- Can be modeled as hybrid system and analyzed formally

Hierarchical Control

- Splits at fixed sampling times



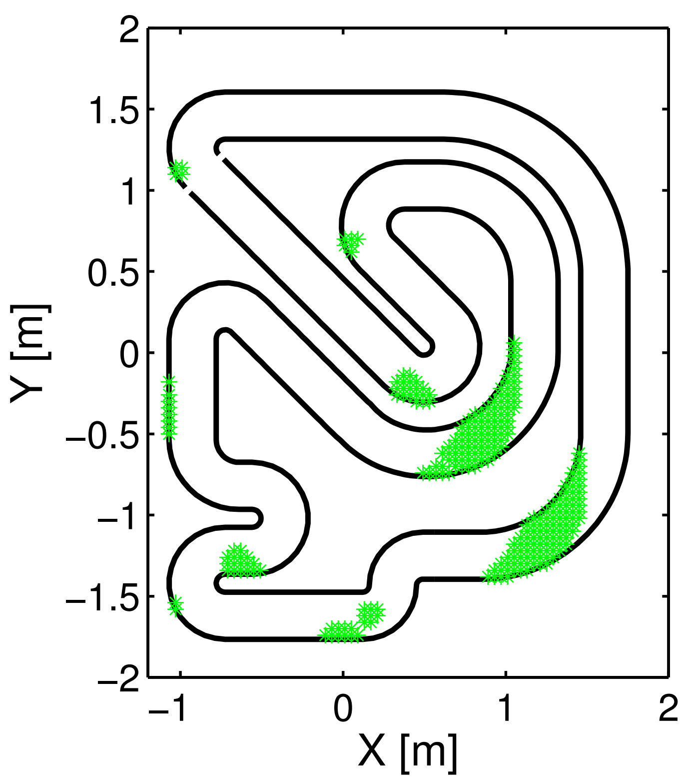
- Assumption:
  - new constant velocity can be reached immediately
- Find trajectory:
  - with maximal progress
  - which does not leave the track
- Tree grows exponentially

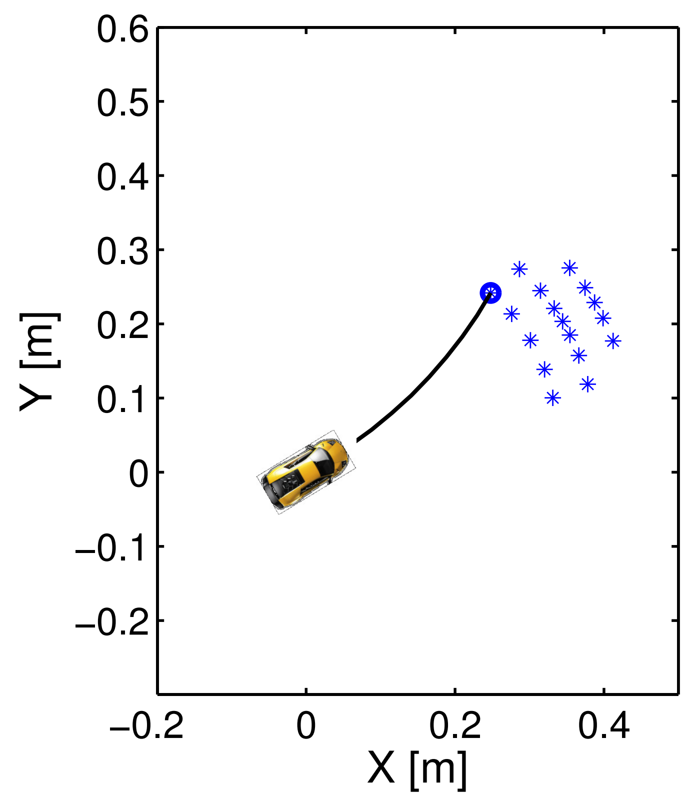
$$p(X,Y) := \arg\min_{\theta}(X - X^{\text{cen}}(\theta))^2 + (Y - Y^{\text{cen}}(\theta))^2$$

- Can be modeled as hybrid system and analyzed formally

Fast Viable Path Planning

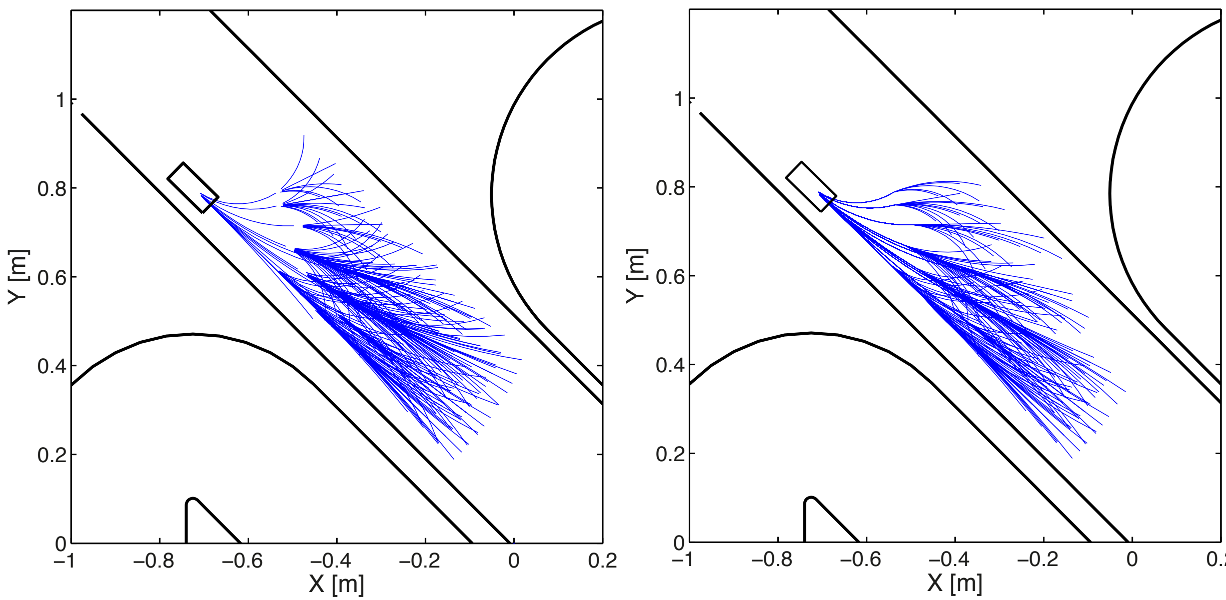
- Based on viability theory (invariant set theory)
  - reconstructing all “safe” motion primitives given the state
- Driving straight with 2m/s with a fix angle





Fast Viable Path Planning

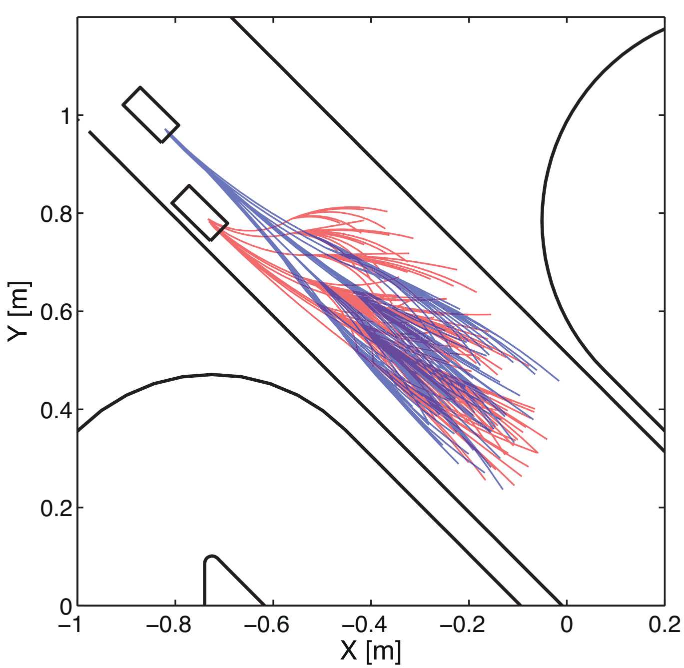
- Only generating trajectories which are recursive feasible
  - only generate 145 vs 341 trajectories
  - no constraint checks necessary

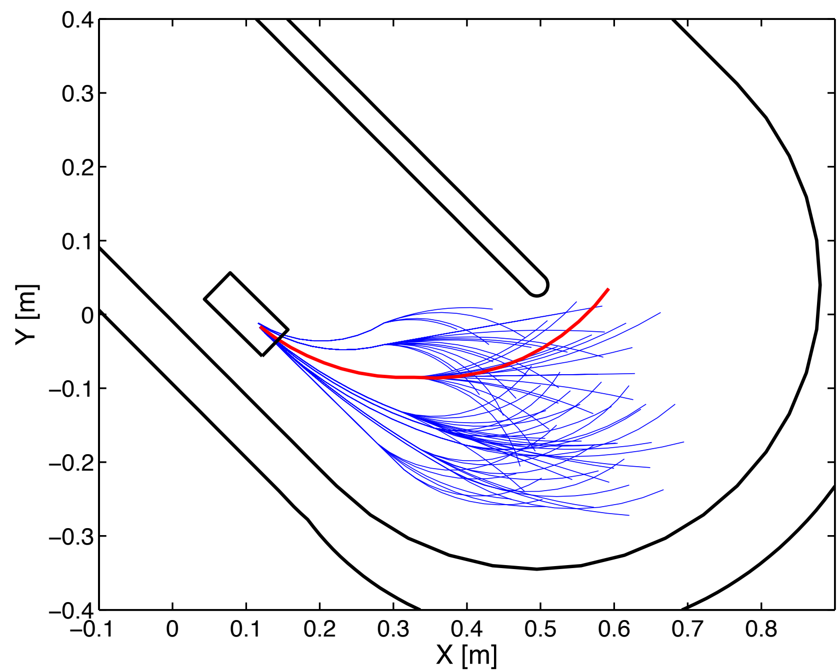


	Naive Path Planner	$\mathcal{V}_{\text{safe}}(x)$ Path Planner
mean [s]	0.219	0.0205
max [s]	0.530	0.0749
min [s]	0.034	0.003

Prediction of Opponent

- Efficient and fast calculation of possible movements
- Could be used for dynamic obstacle avoidance
  - robust or probabilistic collision avoidance
  - store one optimal trajectory (we showed that this is a Nash equilibrium)





Acknowledgments

- Students involved in MPCC:
  - Florian Perrodin (First MPCC Simulations)
  - Samuel Zhao (MPCC Implementation I)
  - Kenneth Kuchera (MPCC Implementation II)
  - Michael Janser (Dynamic Programming for obstacle avoidance)
  - Sandro Merkli (Embedded controllers & electronics)
  - Marcin Dymczyk (Embedded software stack & electronics)
  - Nils Wenzler (Control loop and communication framework)
  - Christian Stocker (Embedded hardware design)
  - Michael Dahinden (Embedded hardware design)
  - Simon Tanner (Vision System)

- Prof. Colin Jones & Prof. Manfred Morari

Closed Loop Results

- Lap time **8.5 - 9** s