Real Time Control for Autonomous Racing based on Viability Theory

Alex Liniger

29 September, 2016

Automatic Control Laboratory, ETH Zurich

www.control.ee.ethz.ch

Autonomous cars

Autonomous driving

- Google:
  - 1.600.000 km (2015)
- Every major car company

Autonomous racing

- Driving at handling limit
- ROBORACE in Formula E
- 2016 - 2017 session

Autonomous cars

Autonomous driving

- Google:
  - 1.600.000 km (2015)
- Every major car company

Autonomous racing

- Driving at handling limit
- ROBORACE in Formula E
- 2016 - 2017 session

Experimental set-up

IR Camera System

Ethernet

Controller Linux PC

Bluetooth

1:43 miniature RC race cars

Experimental set-up

IR Camera System

Ethernet

Controller Linux PC

Bluetooth

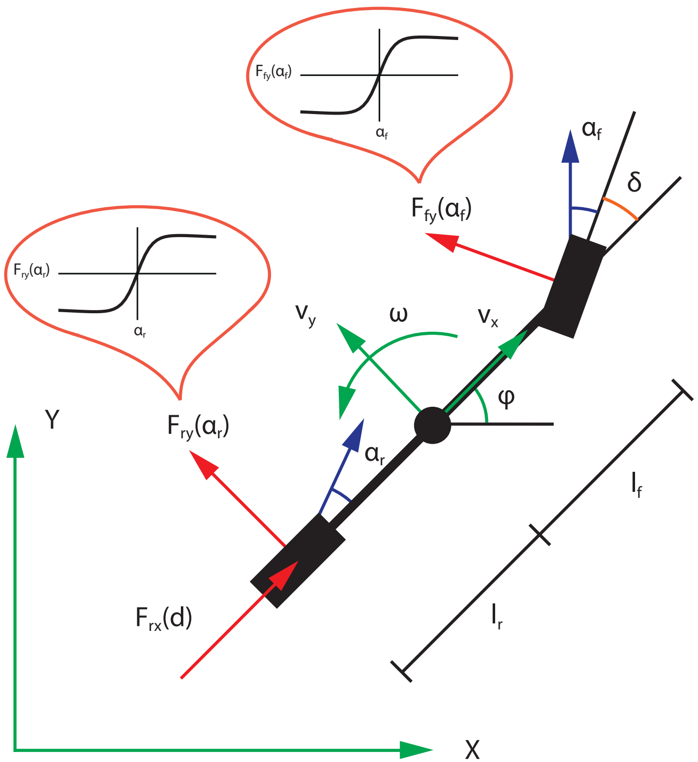
1:43 miniature RC race cars

Layout

- Introduction
- Hierarchical controller
- Viability kernel
- Error due to space discretization
- Viability kernel in autonomous racing
- Simulation and Experimental results
- Conclusion

Car Model

- Bicycle model, with nonlinear lateral tire forces (Pacejka)

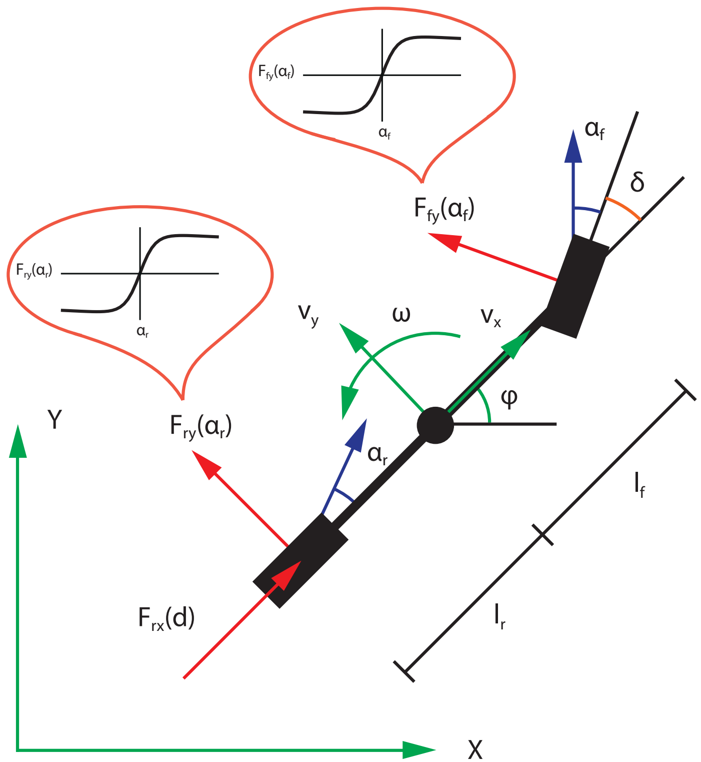


$$\dot{X} = v_x \cos(\varphi) - v_y \sin(\varphi)$$
$$\dot{Y} = v_x \sin(\varphi) + v_y \cos(\varphi)$$
$$\dot{\varphi} = \omega$$
$$\dot{v}_x = \frac{1}{m}(F_{r,x} - F_{f,y} \sin\delta + m v_y \omega)$$
$$\dot{v}_y = \frac{1}{m}(F_{r,y} + F_{f,y} \cos\delta - m v_x \omega)$$
$$\dot{\omega} = \frac{1}{I_z}(F_{f,y}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

- Highly nonlinear 6 dimensional system

Car Model

- Bicycle model, with nonlinear lateral tire forces (Pacejka)

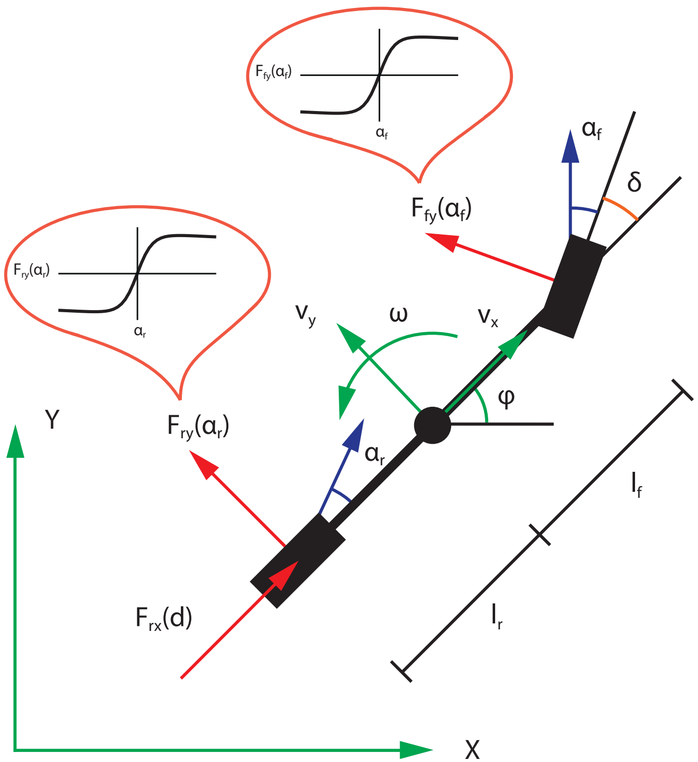


$$\dot{X} = v_x \cos(\varphi) - v_y \sin(\varphi)$$
$$\dot{Y} = v_x \sin(\varphi) + v_y \cos(\varphi)$$
$$\dot{\varphi} = \omega$$
$$\dot{v}_x = \frac{1}{m}(F_{r,x} - F_{f,y} \sin\delta + m v_y \omega)$$
$$\dot{v}_y = \frac{1}{m}(F_{r,y} + F_{f,y} \cos\delta - m v_x \omega)$$
$$\dot{\omega} = \frac{1}{I_z}(F_{f,y}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

- Highly nonlinear 6 dimensional system
- Separation is slow and fast dynamics

Car Model

- Bicycle model, with nonlinear lateral tire forces (Pacejka)



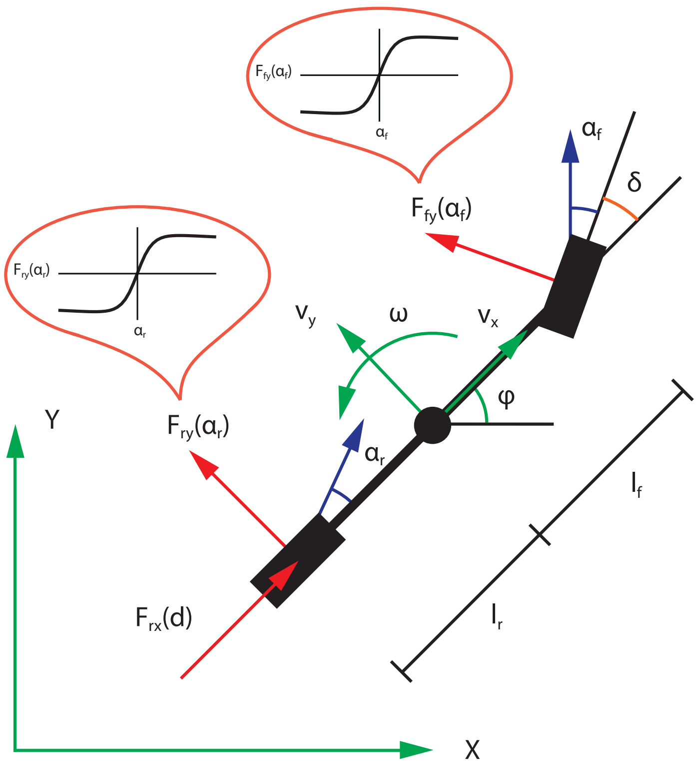
$$\dot{X} = v_x \cos(\varphi) - v_y \sin(\varphi)$$
$$\dot{Y} = v_x \sin(\varphi) + v_y \cos(\varphi)$$
$$\dot{\varphi} = \omega$$

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

- Highly nonlinear 6 dimensional system
- Separation is slow and fast dynamics

Car Model

- Bicycle model, with nonlinear lateral tire forces (Pacejka)



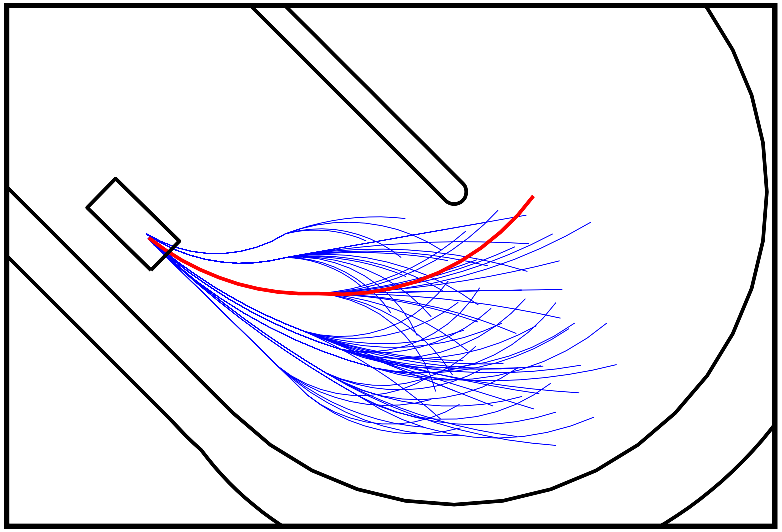
$$\dot{X} = v_x \cos(\varphi) - v_y \sin(\varphi)$$
$$\dot{Y} = v_x \sin(\varphi) + v_y \cos(\varphi)$$
$$\dot{\varphi} = \omega$$

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

- Highly nonlinear 6 dimensional system
- Separation is slow and fast dynamics

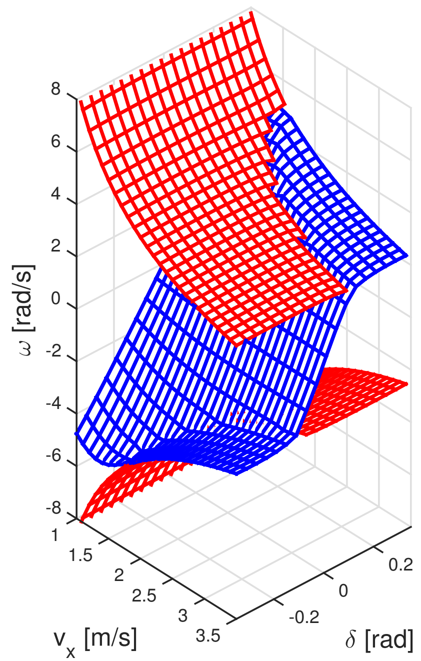
Hierarchical Control Structure

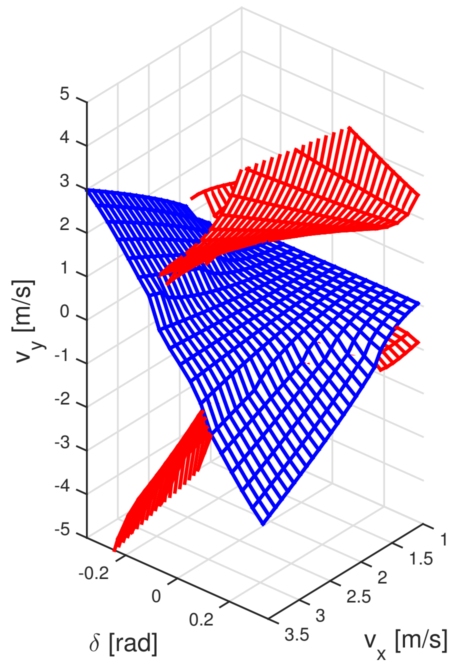
- “Path planning” based on constant velocity model
  - Plan for the slow dynamics
  - reduced dimension
  - “slow” actuation times
- Tracking planned path using MPC
  - Following path given the full dynamics
  - path planner gives linearization points



Time Scale Separation - Constant Velocities

- Velocities (v_x, v_y, ω) "always" at steady state
- find points where (v_x, v_y, ω) are constant

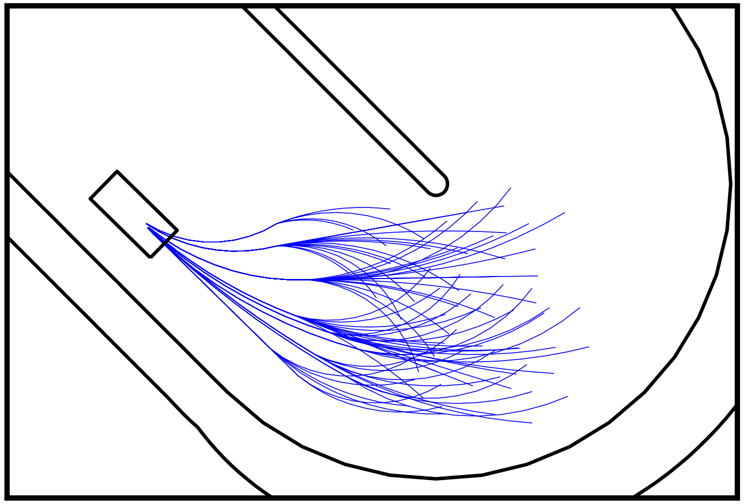




- Gridding stationary velocity points
- Library of possible movements (Motion Primitives)
- Low dimensional grid (~100) can capture the whole system

Path Planning

- Library of constant velocity “primitives”
- Assumptions:
  - new constant velocity can be reached immediately
  - stay at the constant velocity for a fix time period T_{pp}

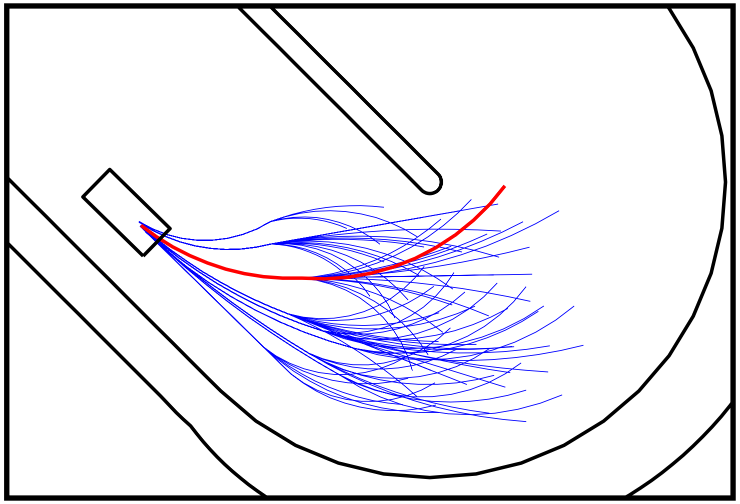


$$\dot{X} = \bar{v}_x(q)\cos(\varphi) - \bar{v}_y(q)\sin(\varphi)$$
$$\dot{Y} = \bar{v}_x(q)\sin(\varphi) + \bar{v}_y(q)\cos(\varphi)$$
$$\dot{\varphi} = \bar{\omega}(q)$$
$$\bar{v}(q) = [\bar{v}_x(q), \bar{v}_y(q), \bar{\omega}(q)]$$

- Transition between constant velocity are restricted:
  - only transitions that are feasible for the dynamics are considered **->** adds a discrete mode

Path Planning

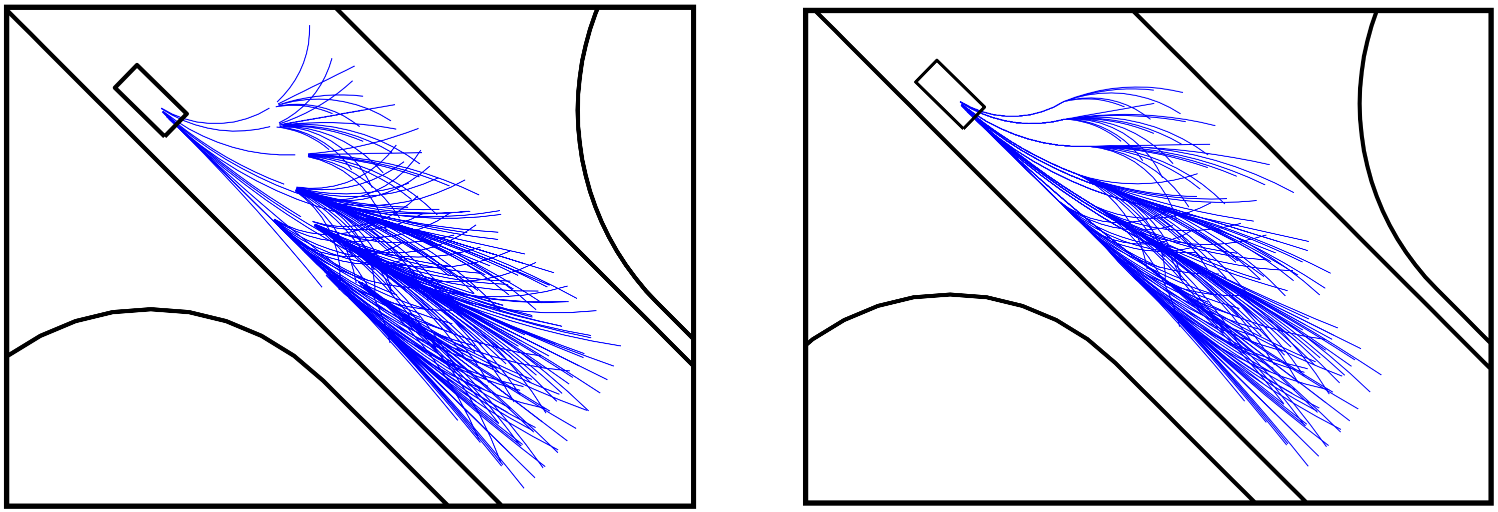
- Tree of possible trajectories
  - exclude all trajectories that leave the track
  - Find the best trajectories with the largest progress



- Tree grows exponentially in the horizon
- Time to check track constraints is the bottle neck
- Optimal trajectory often not recursive feasible/viable

Path Planning

- Only generate recursive feasible/viable trajectories
  - ~~exclude all trajectories that leave the track~~
  - Find the best trajectories with the largest progress



- Tree grows exponentially in the horizon
- check track constraints is not necessary
- All trajectories are recursive feasible/viable

Difference Inclusion

- We look at controlled discrete time system of the form

$$x_{k+1} = f(x_k, u_k)$$

- Assumptions:

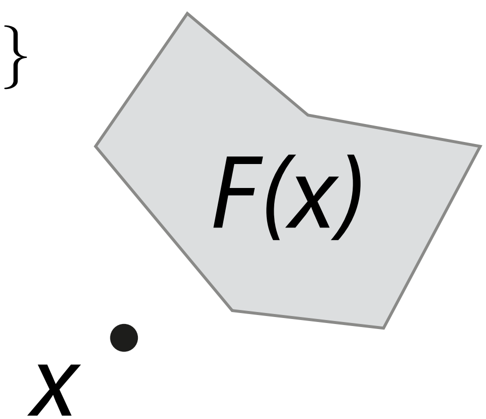
$$x \in \mathbb{R}^n \quad u \in U \subset \mathbb{R}^m$$

$$f : \mathbb{R}^n \times U \to \mathbb{R}^n \quad \text{is continuous}$$

$$f \text{ is } L \text{ Lipschitz w.r.t. x}$$

- The system can be reformulated as a difference inclusion

$$x_{k+1} \in F(x_k) = \{f(x_k, u_k) \mid u_k \in U\}$$



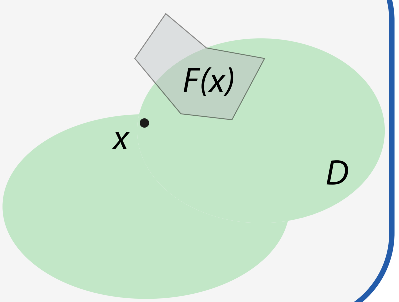
Viability Theory

- Given:
  - a difference inclusion $x_{k+1} \in F(x_k)$
  - $K \subset \mathbb{R}^n$ is a compact set
- a solution is viable if: $\begin{cases} x_{k+1} \in F(x_k) & \forall k \geq 0 \\ x_0 = x \in K & \\ x_k \in K & \forall k \geq 0 \end{cases}$

Definition 1: *[Saint-Pierre 94]*

Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be a set valued map. A closed subset $D \subset \mathbb{R}^n$ is a **viability domain** of *F* if;

$$\forall x \in D, \quad F(x) \cap D \neq \emptyset$$



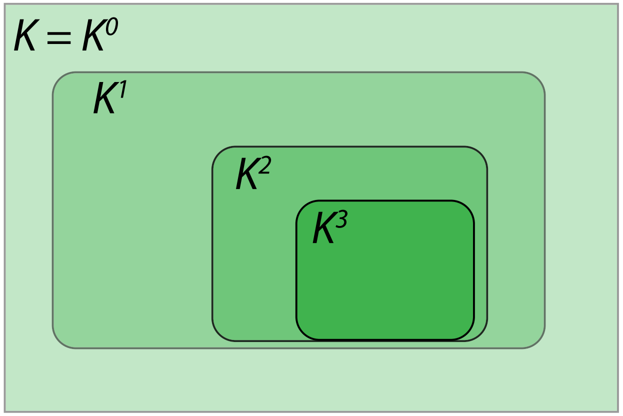
- The **viability kernel** $Viab_F(K)$, is the largest closed viability domain contained in K

Viability Kernel Algorithm

- Given:
  - a discrete difference inclusion $x_{k+1} \in F(x_k)$
  - $K \subset \mathbb{R}^n$ is a compact set
- Construction of $Viab_F(K)$:
  - Sequence of nested subsets

$$K^0 = K$$

$$K^{n+1} = \{x \in K^n | F(x) \cap K^n \neq \emptyset\}$$



Proposition 1: *[Saint-Pierre 94]*

Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be a upper-semicontinuous set-valued map with closed values and let K be a compact subset of $Dom(F)$

$$Viab_F(K) = \bigcap_{n=0}^{\infty} K^n$$

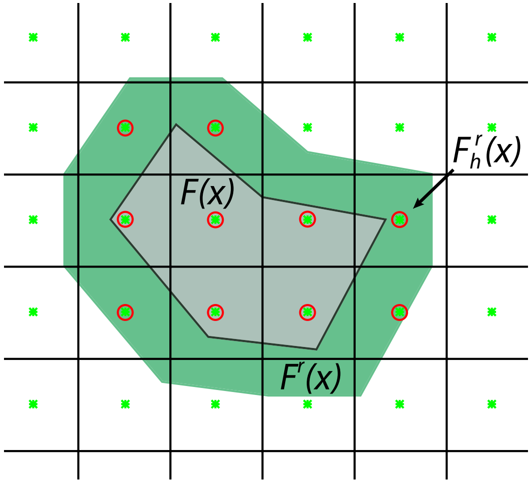
Finite Viability Kernel Algorithm

- Discretization by introducing a countable subset X_h

$$\forall x \in \mathbb{R}^n \quad \exists x_h \in X_h \; : \quad \|x - x_h\|_\infty \leq r$$

- Discretization of the set-valued may be empty
- Extended finite difference inclusion

$$x_{h,k+1} \in F_h^r(x_{h,k}) = (F(x_{h,k}) + rB) \cap X_h$$



- Viability kernel does not change

$$Viab_{F_h^r}(K_h) = \bigcap_{n=0}^{\infty} K_h^{r,n}$$

- and converges in a finite number of steps

Finite Viability Kernel Algorithm

- With $x_{h,k+1} \in F_h^r(x_{h,k}) = (F(x_{h,k}) + rB) \cap X_h$ and mild conditions on *F* and *K*, we have:

[Saint-Pierre 94]

The finite viability kernel does converge with the true kernel

$$\bigcap_{r>0} Viab_{F_h^r}(K_h) = Viab_F(K)$$

[Saint-Pierre 94]

The finite kernel inner approximates the "true" viability kernel in the following way

$$Viab_{F_h^r}(K_h) \subset (Viab_{F^r}(K) \cap X_h)$$

Finite Viability Kernel Algorithm

- With $x_{h,k+1} \in F_h^r(x_{h,k}) = (F(x_{h,k}) + rB) \cap X_h$ and mild conditions on *F* and *K*, we have:

[Saint-Pierre 94]

The finite viability kernel does converge with the true kernel

$$\bigcap_{r>0} Viab_{F_h^r}(K_h) = Viab_F(K)$$

[Saint-Pierre 94]

The finite kernel inner approximates the "true" viability kernel in the following way

$$Viab_{F_h^r}(K_h) \subset (\boxed{Viab_{F^r}(K)} \cap X_h)$$

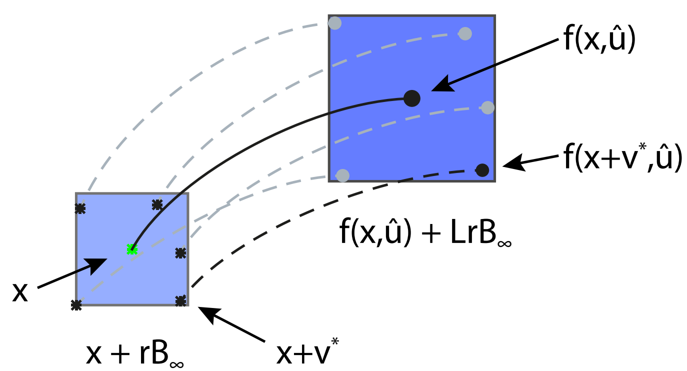
Error due to Space Discretization

- Given $x_{k+1} = f(x_k, u_k)$
  - current state only known to lie within a box of radius r
  - we can bound the error, using Lipschitz continuity:

$$x_{k+1} \in f(x_k, u_k) + LrB_\infty$$

- Idea: Robustify viability kernel against this uncertainty
  - Formulate as an additive disturbance

$$\boxed{x_{k+1} = f(x_k, u_k) + v_k, \qquad \forall v_k \in LrB_\infty}$$



Robust Viability Kernel - Qualitative Game

- Starting point - discrete time system with 2-inputs

$$x_{k+1} = g(x_k, u_k, v_k)$$

- Assumptions:

$$x \in \mathbb{R}^n \quad u \in U \subset \mathbb{R}^m \quad v \in V \subset \mathbb{R}^p$$

$$f : \mathbb{R}^n \times U \times V \to \mathbb{R}^n \quad \text{is continuous}$$

$$f \text{ is } L \text{ Lipschitz w.r.t. x}$$

- Dynamic game between the control and the disturbance
  - disturbance input tries to reach the open set $\mathbb{R}^n \setminus K$
  - control input tries to prevent this event
- Victory domain, can be computed using a slightly adapted viability kernel algorithm
- Feedback policy depends on **state and disturbance**

Discriminating Kernel Algorithm

- Reformulate difference equation as difference inclusion

$$x_{k+1} \in G(x_k, v_k) = \{g(x,u,v) | u \in U\}$$

Definition 2: *[Cardaliaguet 99]*

A closed subset $Q \subset \mathbb{R}^n$ is a **discriminating domain** of G if;

$$\forall x \in Q, \quad G(x,v) \cap Q \neq \emptyset \quad \forall v \in V$$

The largest discriminating domain of G contained in K is called the **discriminating kernel**, denoted by $Disc_G(K)$

- Algorithm to calculate the discriminating kernel

$$K^0 = K$$

$$K^{n+1} = \{x \in K^n | \, \forall v \in V, \; G(x,v) \cap K^n \neq \emptyset\}$$

- Algorithm converges under mild assumptions to $Disc_G(K)$

Space Discretisation Robust Viability Kernel

- Using the following difference inclusion

$$x_{k+1} \in G(x_k, v_k) = F(x_k) + v_k \quad \forall v_k \in LrB_\infty$$

- The following properties hold for $Disc_G(K)$

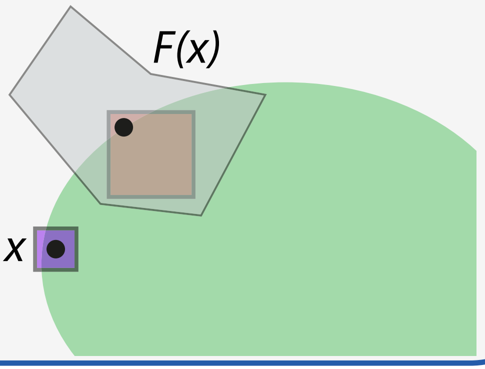
Proposition:

1.) $\bigcap_{r>0} Disc_G(K) = Viab_F(K)$.

2.) $Disc_G(K)$ is a viability domain of F.

3.) $\forall x \in Disc_G(K)$ and $\forall \hat{x} \in x + rB_\infty$,

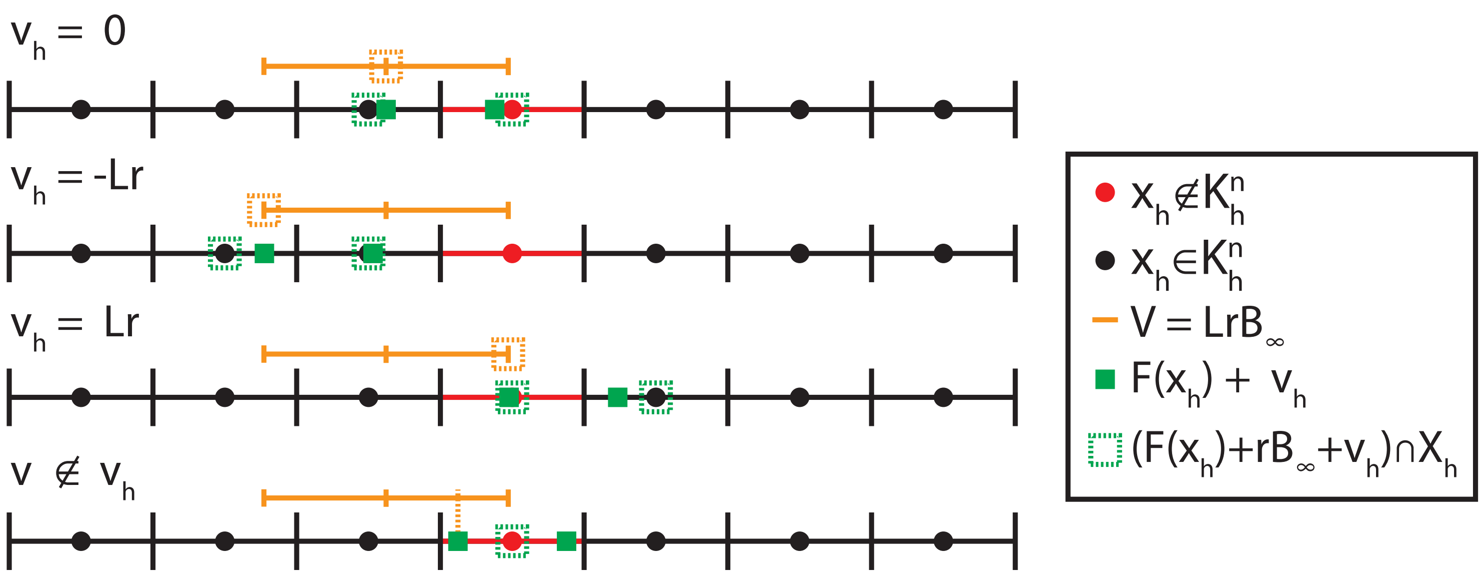
$$\exists u \in U : \; f(\hat{x}, u) \in Disc_G(K) \, .$$



- Motivated by space discretization:
  - **BUT** results hold for continuous space

Finite Inner Approximation

- space discretisation leads to an inner approximation
- disturbance space discretization causes problems
  - not all possible disturbances are considered!

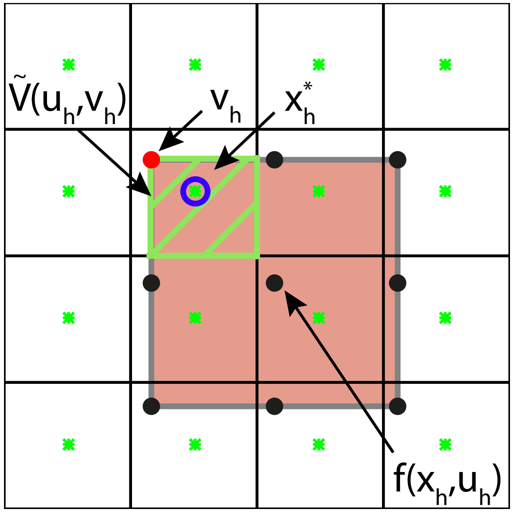


- Simple 1-D example with a 2-input set valued map

Finite Disturbance Space

- Find relationship between disturbance grid point and continuous disturbances
- Assumptions:
  - discrete input space (we can always discretize)
  - regular state and disturbance grids
- For every **disturbance grid point** v_h and **control** u_h, there exists a subset $\tilde{V}(u_h, v_h)$ that maps to the same grid point

$$
\begin{aligned}
x_h^* &= (f(x_h, u_h) + v_h + rB_\infty) \cap X_h \\
\tilde{V}(u_h, v_h) &= \{v \in V | \\
&\qquad \|x_h^* - (f(x_h, u_h) + v)\|_\infty \leq r\}
\end{aligned}
$$



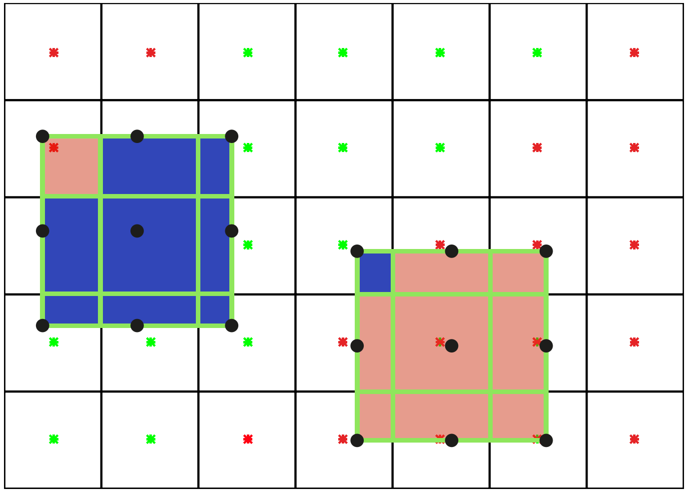
Finite Disturbance Space

- Use link between discrete and continuous disturbance

$$G_h^r(x_h, v) \cap K_h^n \neq \emptyset \quad \forall v \in V$$

$$G_h^r(x_h, v_h) \cap K_h^n \neq \emptyset \quad \forall v_h \in V_h$$

- 2-D example with 2 discrete inputs



- Union of the blue sets should be equal to V
- We can efficiently compute the sets $\tilde{V}(u_h, v_h)$ and conservatively verify that the union is equal to V

Finite Inner Approximation

- Using the propose algorithm the finite discriminating kernel is an inner approximation
  - First two properties of Proposition still hold
  - Third point changes

Proposition:

If the proposed algorithm is used and *r* is identical to the regular grid spacing:

$$\forall x_h \in Disc_{G_h^r}(K_h) \text{ and } \forall \hat{x} \in x_h + rB_\infty$$
$$\exists u_h \in U_h : \; f(\hat{x}, u_h) \in (Disc_{G_h^r}(K) + rB_\infty)$$

Reconstructing Viable Controls

- Exploiting this guarantee using predictive controller

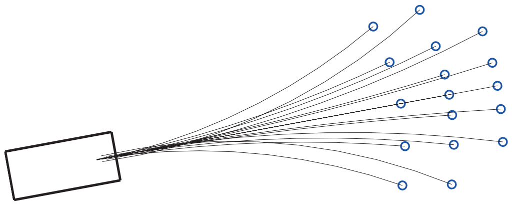
$$
\begin{aligned}
\min_{x, u_h} \; & \sum_{k=0}^{N_S} J(x_k, u_k) \\
\text{s.t.} \quad & x_0 = x \\
& x_{k+1} = f(x_k, u_{h,k}) \,, \quad u_{k,h} \in U_h \\
& f(x_k, u_{k,h}) \in Disc_{G_h^r}(K_h) + rB_\infty
\end{aligned}
$$

- When using viability kernel we can even pre-compute all viable inputs

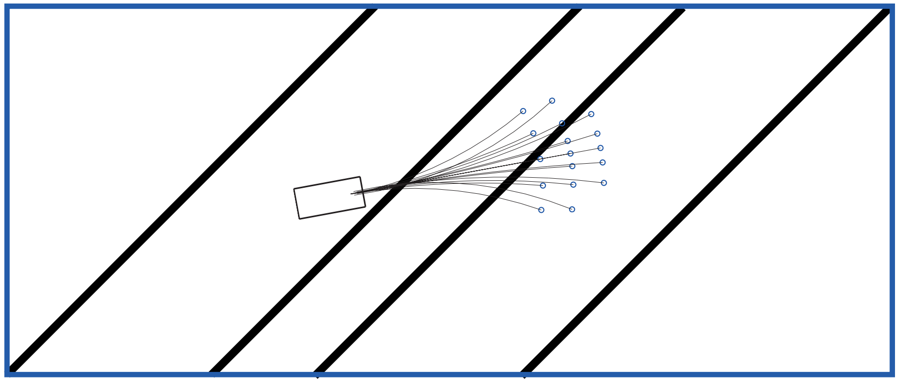
$$
U_{\mathrm{V}}(x_h) = \begin{cases} \left\{ \begin{array}{l} u_h \in U_h \mid \\ f(x_h, u_h) + rB_\infty \\ \cap Viab_{F_h^r}(K_h) \neq \emptyset \end{array} \right\} & \begin{array}{l} \text{if } x_h \in \\ Viab_{F_h^r}(K_h) \end{array} \\ \emptyset & \text{otherwise} \end{cases} .
$$

Autonomous Racing - Path Planning Model

- Path planning model is a data-sampled system with ZOH



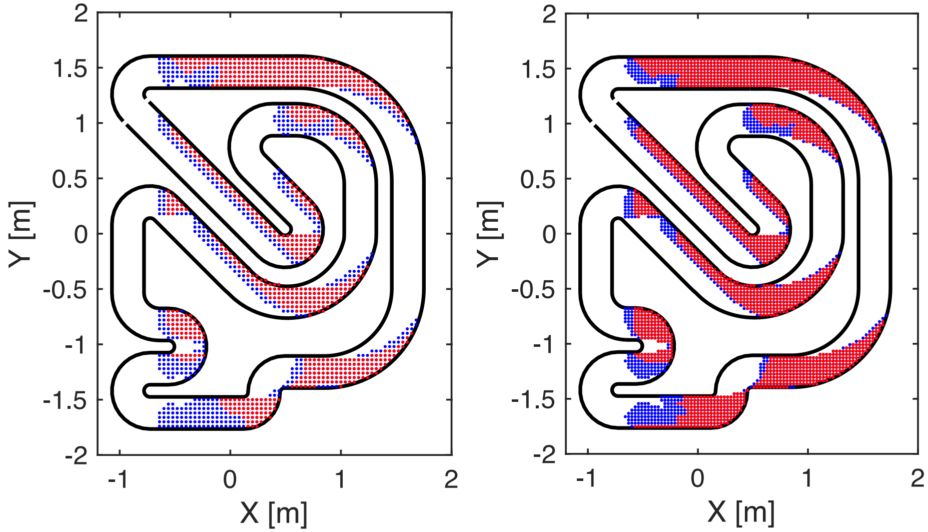
- Idea:
  - formulate as a discrete time system
  - difference inclusion reformulation
  - deal with continuous evolution in a pre-processing step



- Exclude all inputs which leave track from the set-valued map

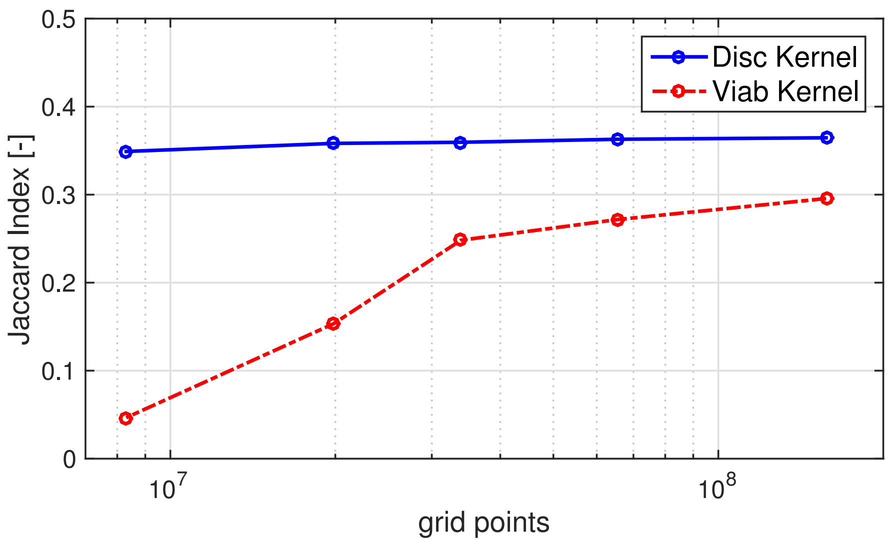
Viability and Discriminating Kernel

- Constraint set $K := \begin{cases} (X,Y) \in \mathcal{X}_{\text{Track}} \, , \\ \varphi \in [0, 2\pi] \, , \\ q \in Q \, . \end{cases}$
- Gridding:
  - *(X,Y)* -> 4/3 cm between grid points
  - ϕ -> 0.04/0.03 rad between grid points
  - q -> 129 modes



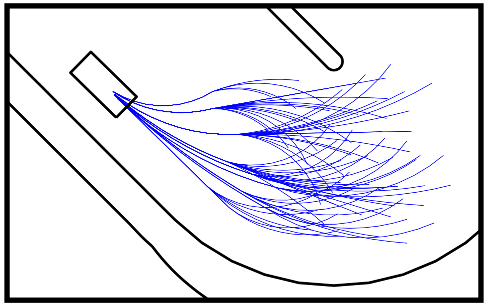
Viability and Discriminating Kernel

- Constraint set $K := \begin{cases} (X, Y) \in \mathcal{X}_{\text{Track}} \,, \\ \varphi \in [0, 2\pi] \,, \\ q \in Q \,. \end{cases}$
- Gridding:
  - *(X,Y)* -> 4/3 cm between grid points
  - ϕ -> 0.04/0.03 rad between grid points
  - q -> 129 modes



Simulation Results

- Every 20 ms redo path planning and MPC step
- Simulation using full non-linear model
- Based on sensitivity study we determined
  - $T_{pp} = 0.16$ s
  - $N_S = 3$
  - $N_M = 129$

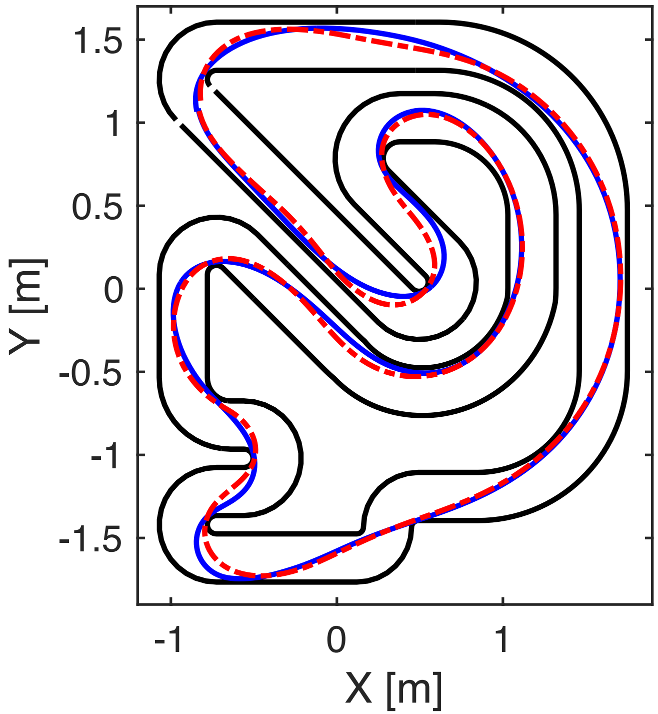


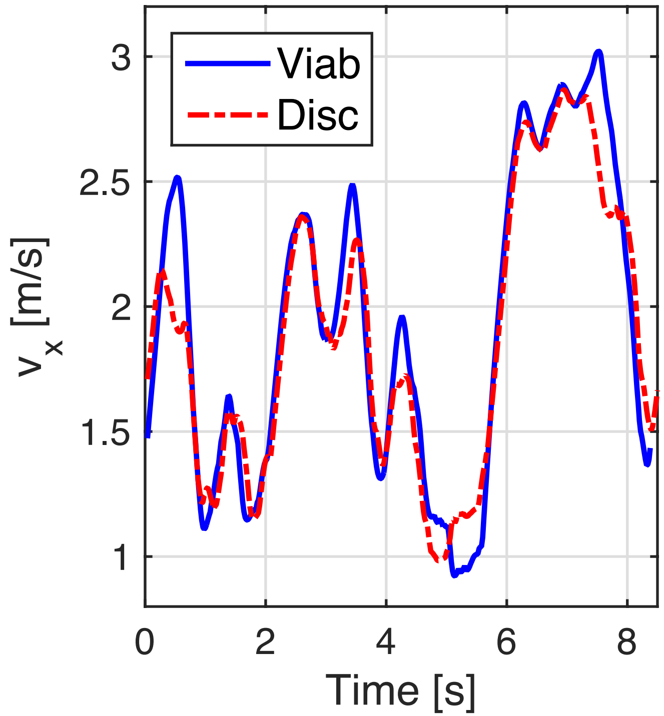
- Comparing: Viability vs Discrimination vs no kernel

Kernel	mean lap time [s]	# constr. violations	median comp. time [ms]	max comp. time [ms]
No	8.76	4	32.26	247.7
Viab	8.57	0	0.904	7.968
Disc	8.60	1	0.870	7.533

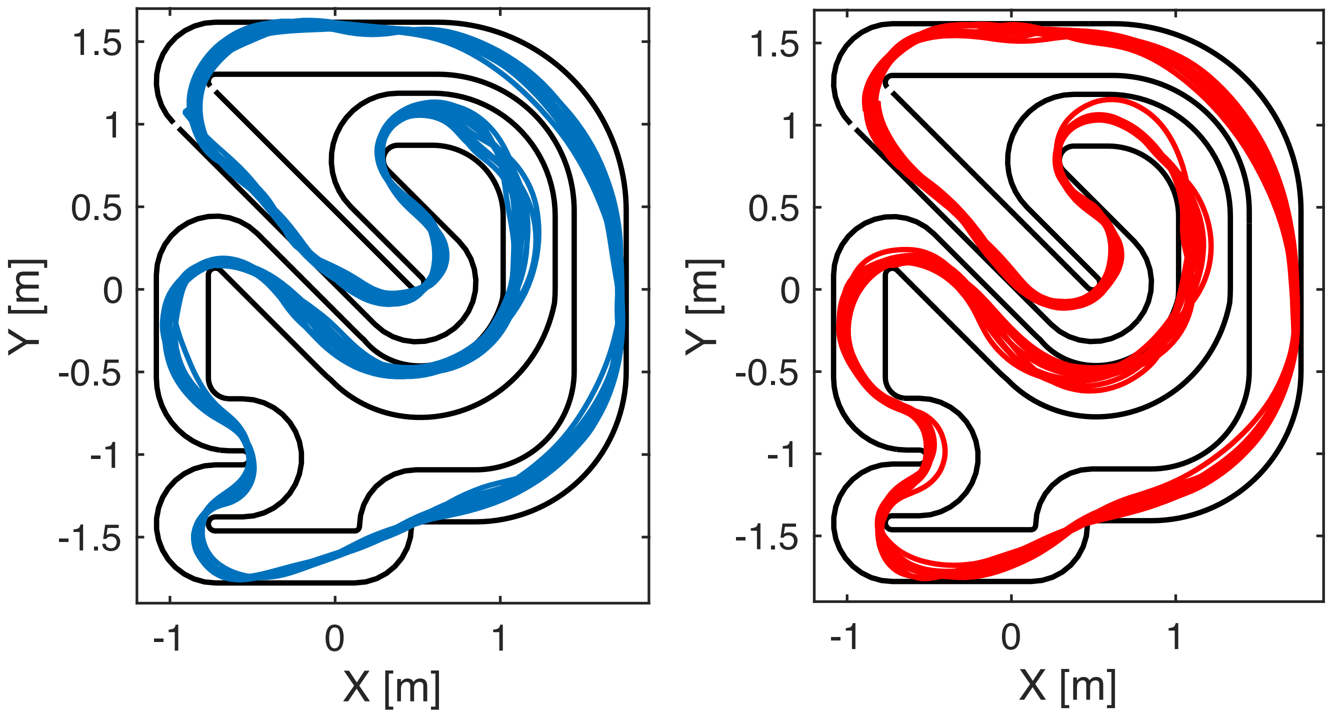
Simulation Results - Viab vs Disc

- Most of the time similar driving
- Disc based controller breaks earlier thereby achieving higher cornering speeds
- Two effects compensate each other leading to the practically the same mean lap time





Experimental Results



Kernel	mean lap time [s]	constr. violations prob. [%]	median comp. time [ms]	max comp. time [ms]
Viab	8.85	0.834	1.124	10.164
Disc	8.996	0.244	1.169	12.839

Conclusion

- We showed:
  - a control approach for real time autonomous racing
  - how viability theory can help to speed up computation while improving the performance
  - how viability and terminal set constraint can help in predictive controllers
- We introduce a new numerical scheme to compute the viability kernel, which incorporates the uncertainty introduce by gridding the state space

Outlook

- Pruning based on upper bound on the cost
- Improving MPC (e.g., NLP solver, including uncertainty)
- Terminal state constraints in MPC, recursive feasibility for the whole system
- Use the viability based controller to implement non-cooperative racing games