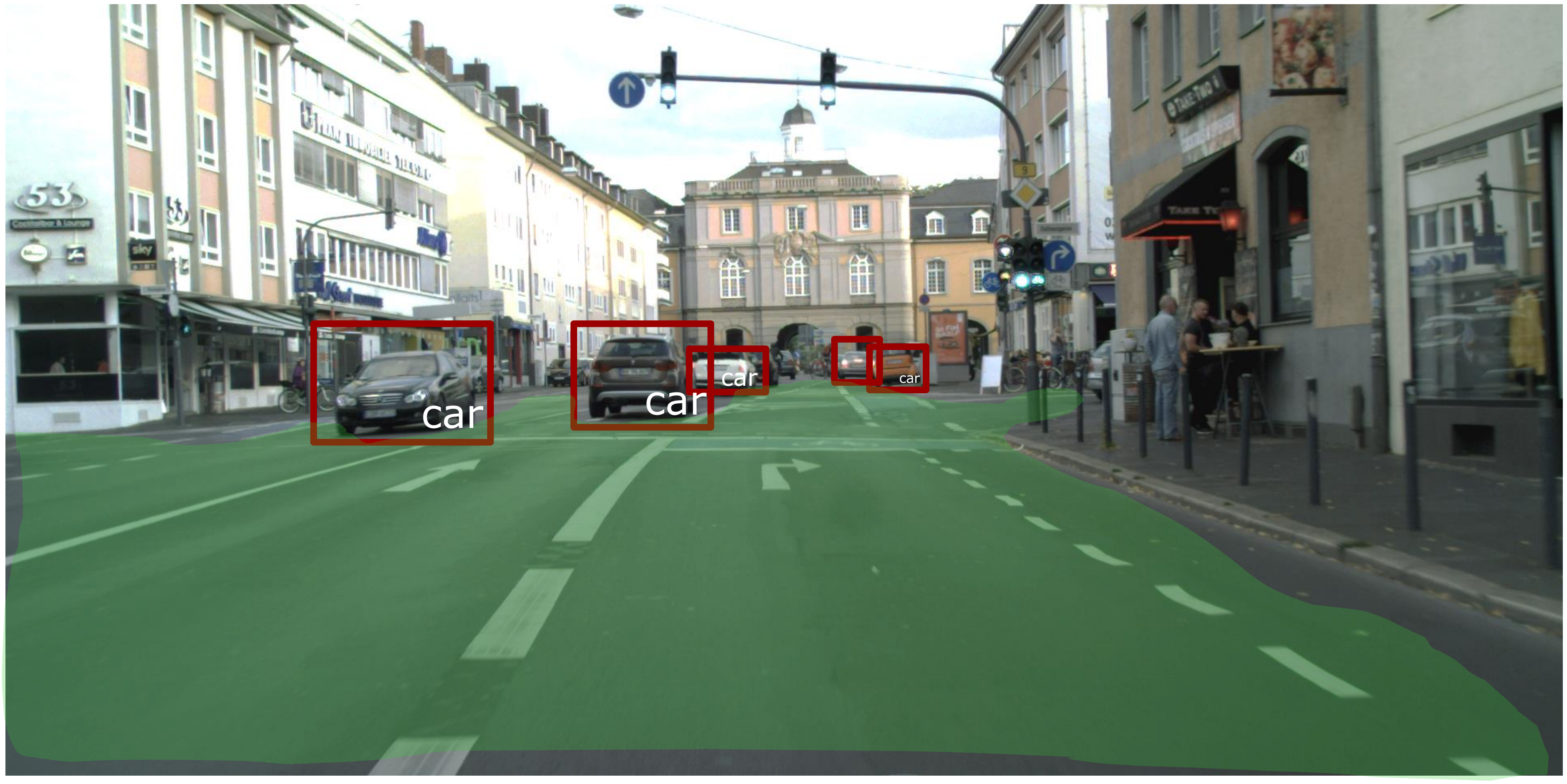
Photogrammetry & Robotics Lab

Introduction to
Model Predictive Control

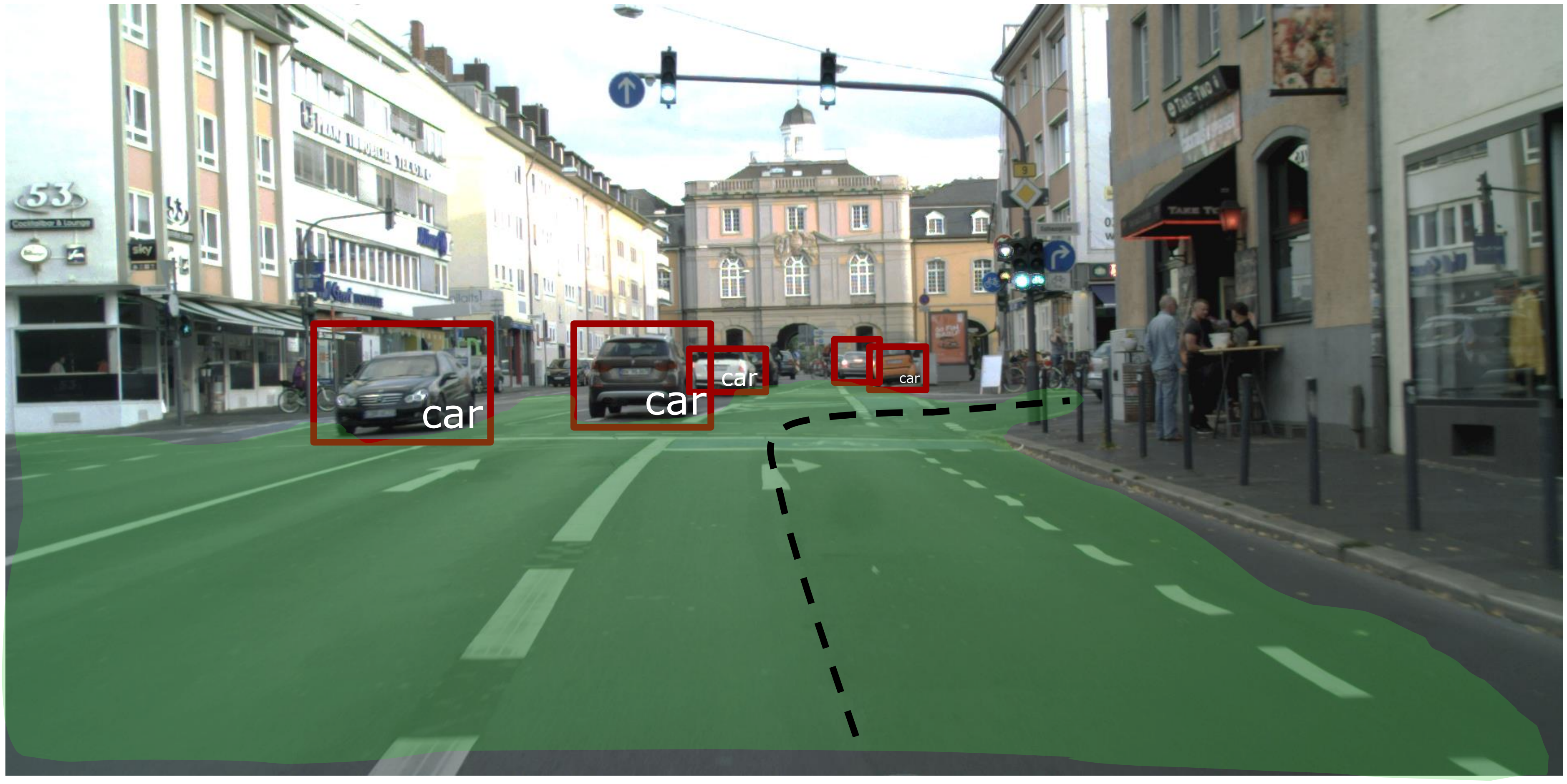
Lasse Peters

Autonomous Driving Scenario

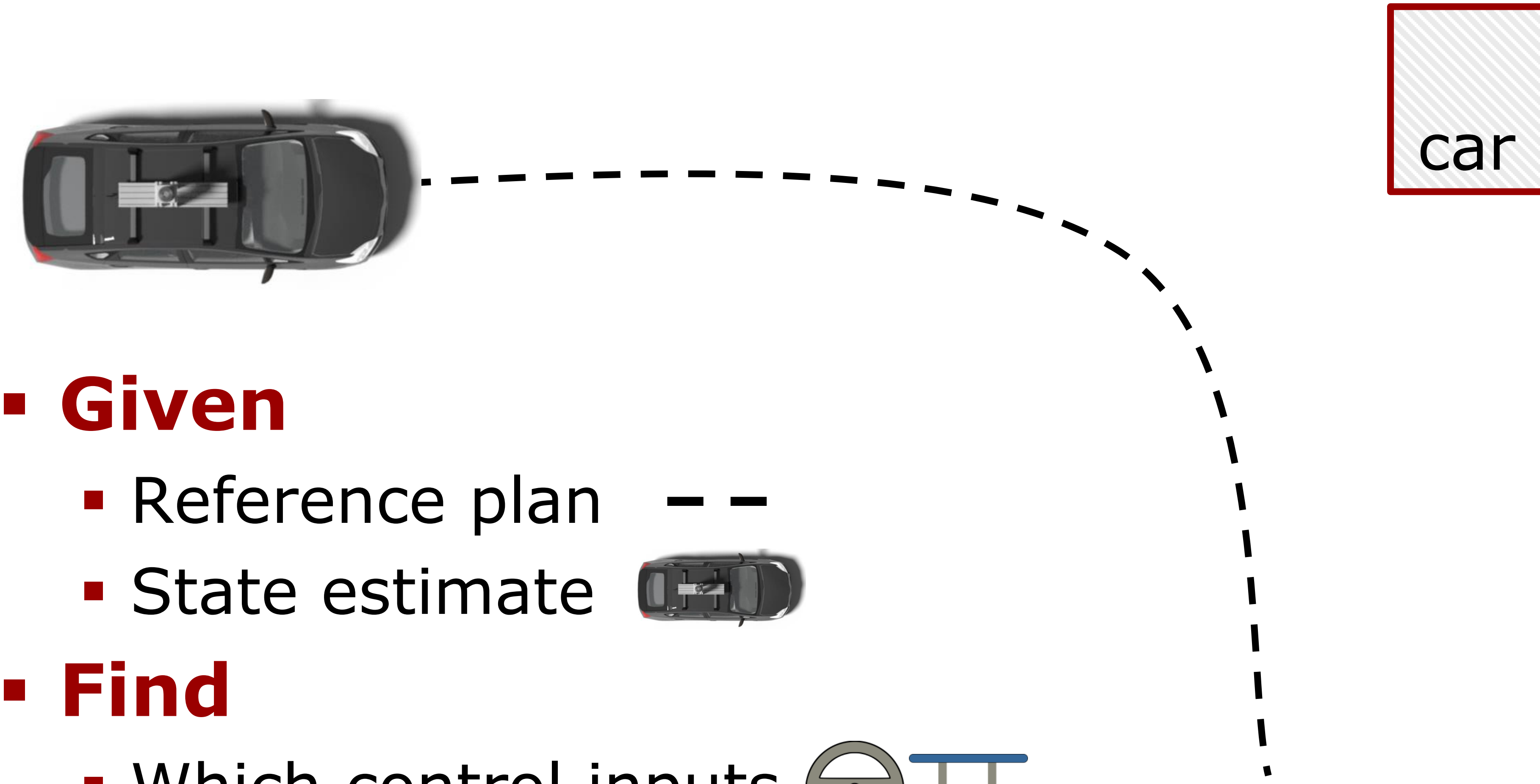
Autonomous Driving Scenario



Autonomous Driving Scenario



Introduction: The Control Task



- **Given**
  - Reference plan – –
  - State estimate
- **Find**
  - Which control inputs achieve the plan?

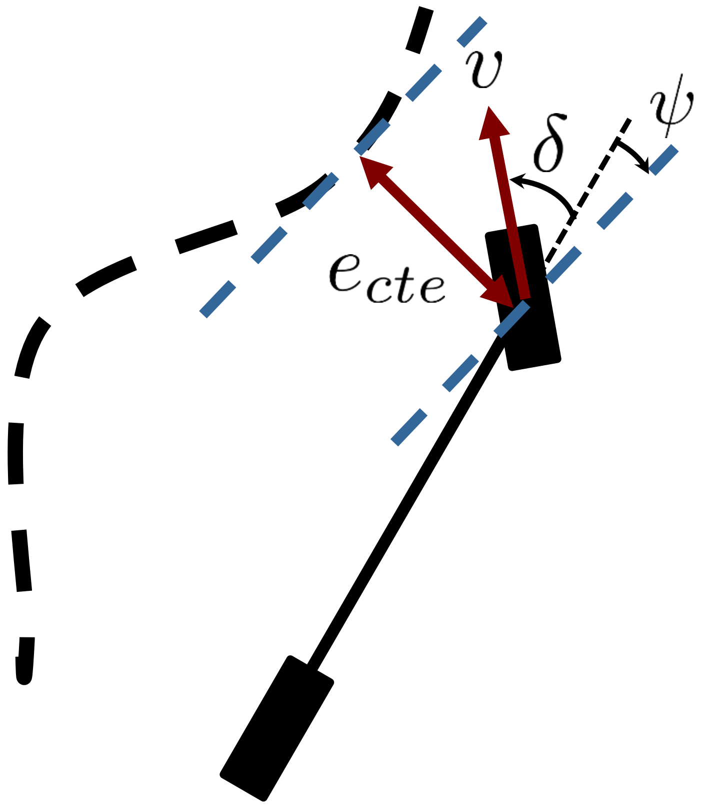
Reminder: Reactive Control

Typically, decomposition of the problem, e.g.

- **Longitudinal** control via PID

- **Lateral** control via Stanley

$$\delta = \psi + \tan^{-1}\left(\frac{k\, e_{cte}}{v}\right)$$



Limitations of Reactive Control

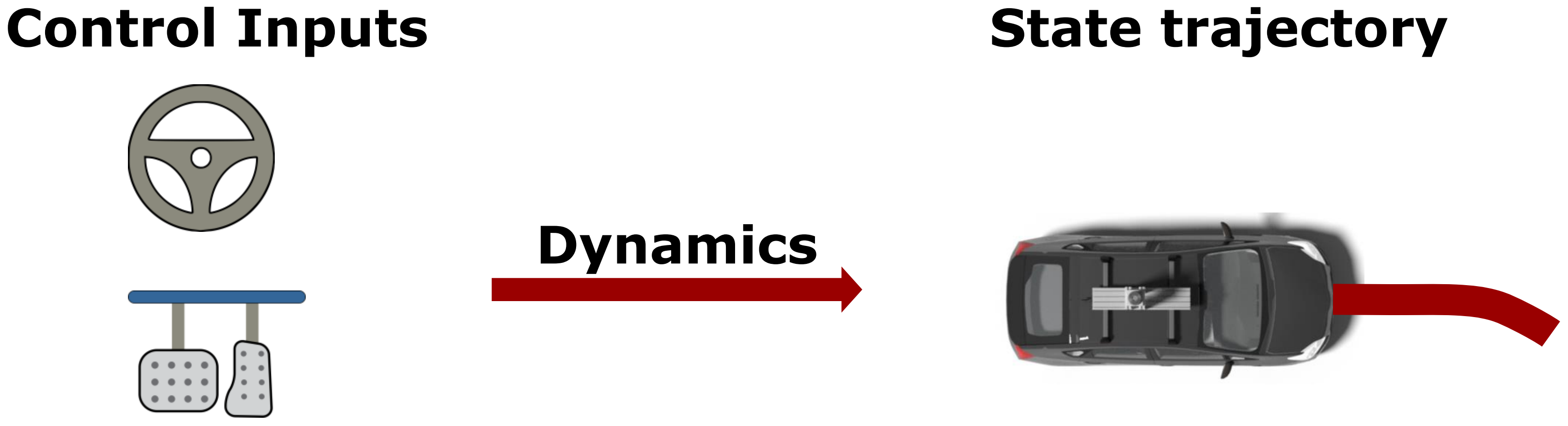
- Non-trivial for more complex systems
- Control gains must be tuned manually
- Separation into longitudinal and lateral controllers ignores coupling
- No handling of constraints such as obstacles
- Ignores future decisions

Optimal Control:
How to best control the system?

Optimal Control: Model

- **Model** of the system dynamics:
  - Predicts the evolution of the state for a given sequence of inputs.

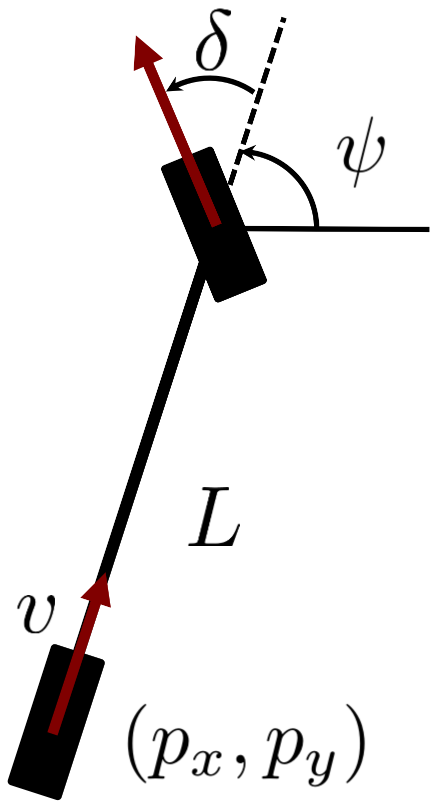
$$x_{t+1} = f(x_t, u_t)$$



Model Example: Discrete 2D Bicycle

$$x_{t+1} = \begin{cases} p_{x,t+1} & = p_{x,t} + \Delta t \ v_t \cos \psi_t \\ p_{y,t+1} & = p_{y,t} + \Delta t \ v_t \sin \psi_t \\ \psi_{t+1} & = \psi_t + \Delta t \ v/L \tan \delta \\ v_{t+1} & = v_t + \Delta t \ a_t \\ \delta_{t+1} & = \delta_t + \Delta t \ \omega_t \end{cases}$$

where $u_t = [a_t, \omega_t]^\top$



Optimal Control: Objective

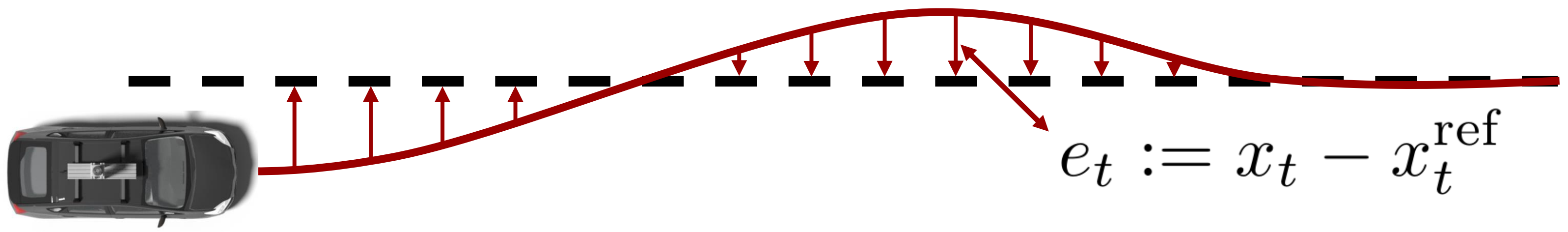
- **Objective**
  Assigns a cost to the trajectory

$$J(x_{1:T}, u_{1:T}) = \sum_{t \in [T]} g_t(x_t, u_t),$$

$$\text{where} \quad x_{1:T} := (x_1, \ldots, x_T), u_{1:T} := (u_1, \ldots, u_T)$$

- Example: deviation from a reference

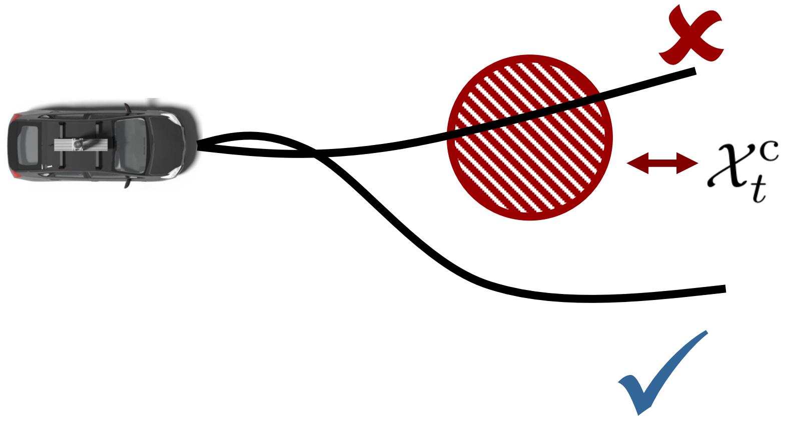
$$g_t(x_t, u_t) = e_t^\top Q_t e_t + u_t^\top R_t u_t$$



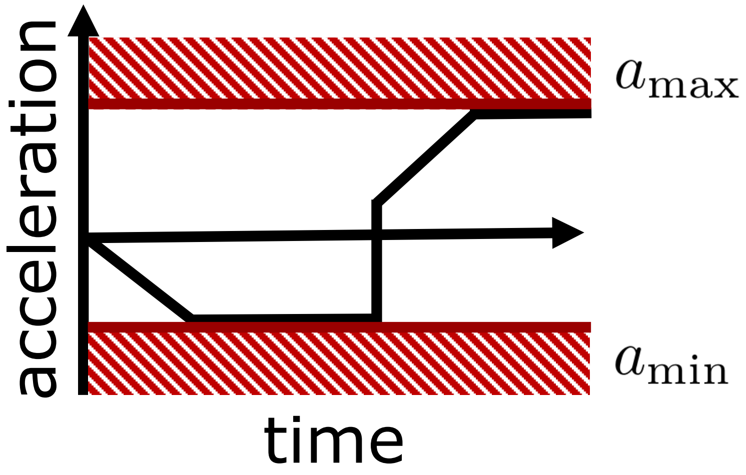
Optimal Control: Constraints

- **Constraints**
  Encode the domains of allowed states $x_t \in \mathcal{X}_t$ and inputs $u_t \in \mathcal{U}_t$.

$$\mathcal{X}_t = \{x \mid p_x^2 + p_y^2 \geq r^2\}$$



$$\mathcal{U}_t = \{u \mid a_{\min} \leq a \leq a_{\max}\}$$



Control as Optimization Problem

- In summary
  - **minimizes the cost** of a …
  - **dynamically feasible** trajectory that …
  - does **not violate the constraints**.

$$\begin{aligned}
\min_{x_{1:T}, u_{1:T}} \quad & J(x_{1:T}, u_{1:T}) \\
\text{subject to} \quad & x_{t+1} = f(x_t, u_t), && \forall t \in [T-1] \\
& u_t \in \mathcal{U}_t, && \forall t \in [T] \\
& x_t \in \mathcal{X}_t, && \forall t \in [T] \\
& x_1 = x_{\text{init}}
\end{aligned}$$

Solving the Optimization Problem

- Typically, no closed-form solution
- Tools for numerical solution:
  - CasADi (C++, Python, Matlab)
  - JuMP.jl (Julia)

$$\begin{aligned} \underset{x,y}{\text{minimize}} \quad & (y - x^2)^2 \\ \text{subject to} \quad & x^2 + y^2 = 1 \\ & x + y \geq 1, \end{aligned}$$

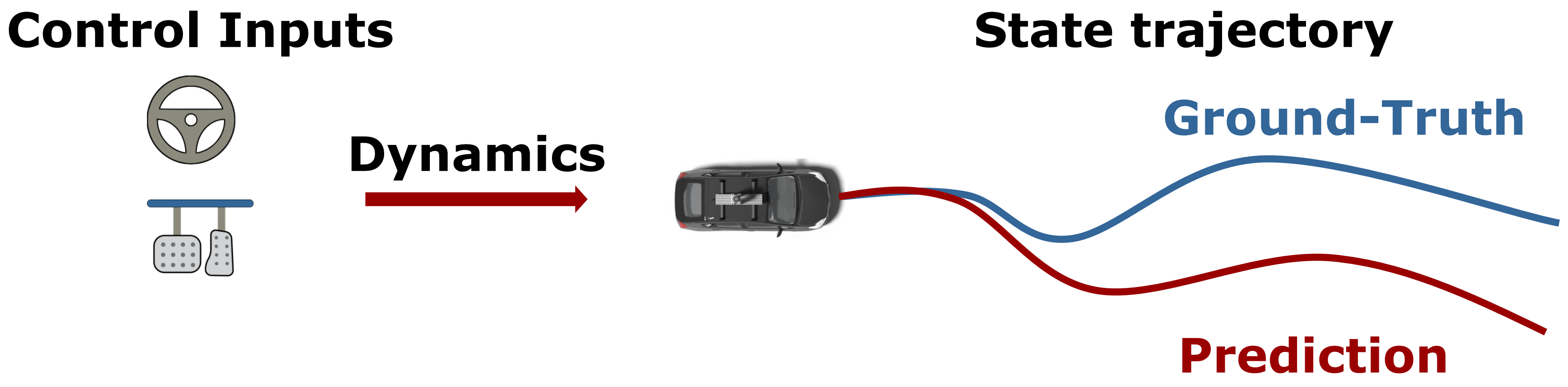
```
opti = casadi.Opti()
x, y = opti.variable(), opti.variable()

opti.minimize(  (y-x**2)**2   )
opti.subject_to( x**2+y**2==1 )
opti.subject_to(       x+y>=1 )

opti.solver('ipopt')
sol = opti.solve()
```

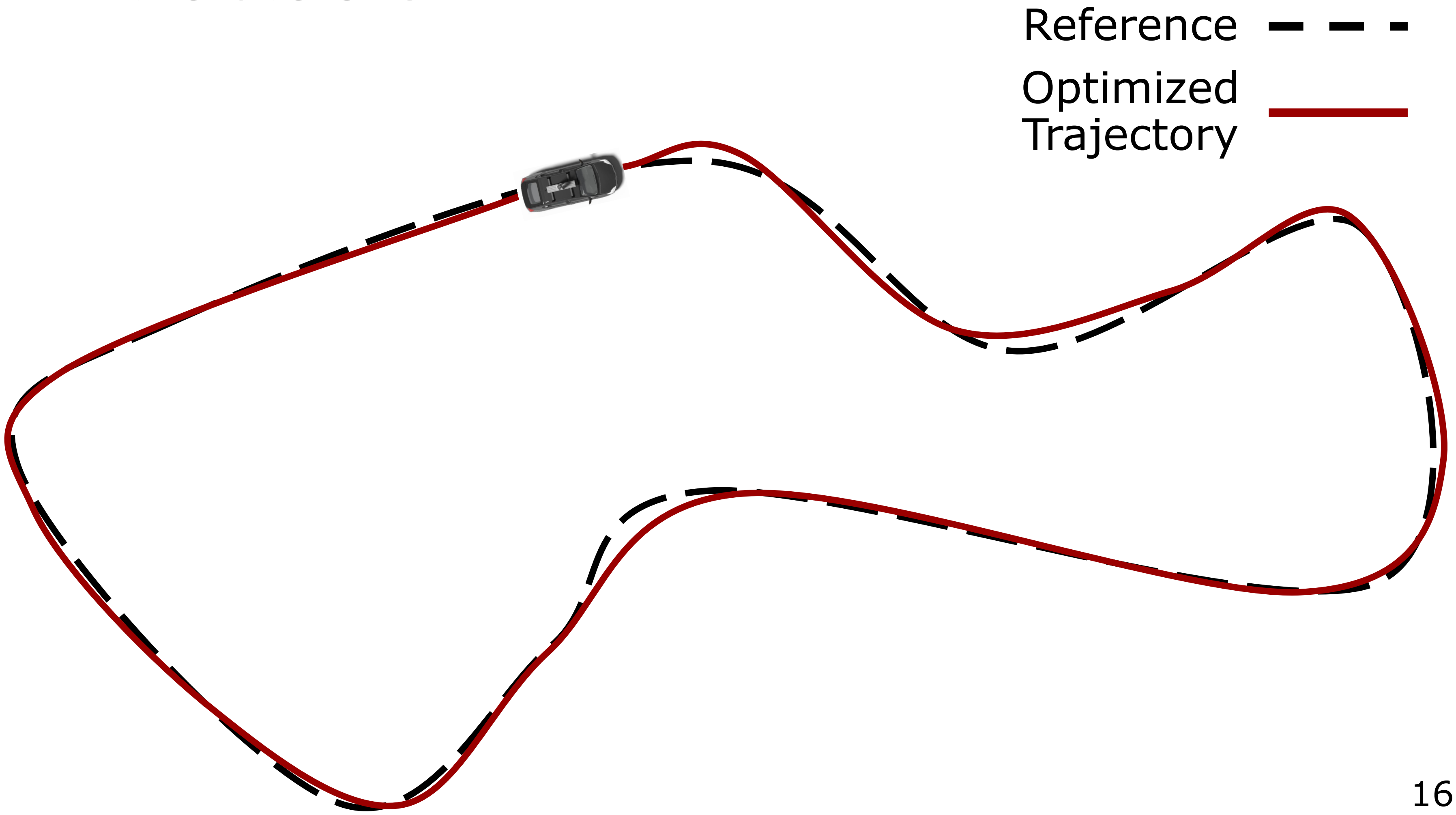
Open-Loop Control: Model Errors

- The prediction model will always be wrong to some extend
- Model errors accumulate over time and result in diverging predictions



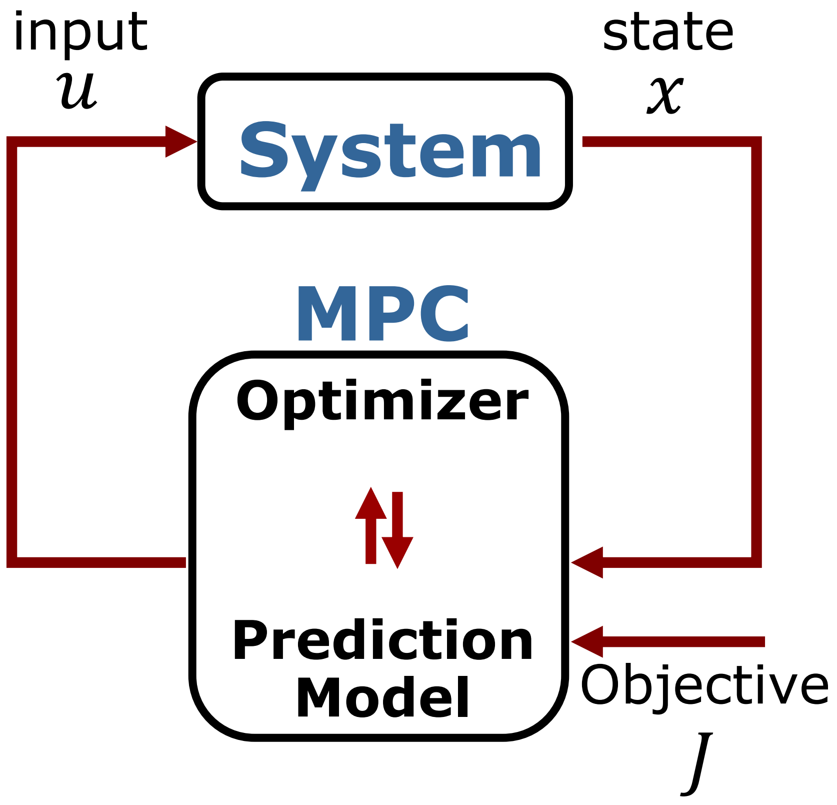
Open-Loop Control: Horizon

- Long task-horizons make the problem intractable



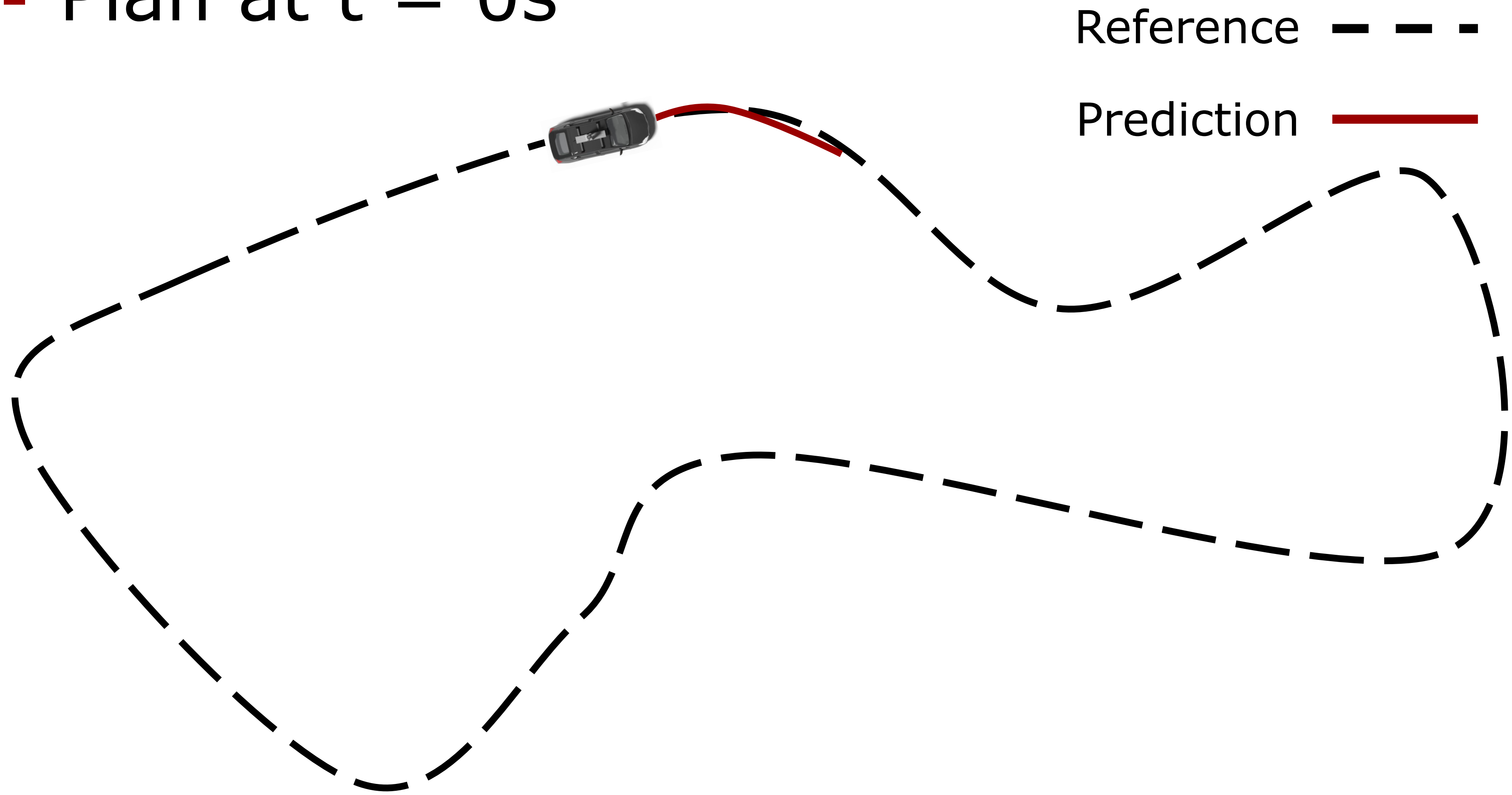
Model Predictive Control (MPC)

- Receding horizon control
  - Start from the **current state**
  - Find controls for a **limited preview** into the future
  - Apply only the first input, then **re-plan**



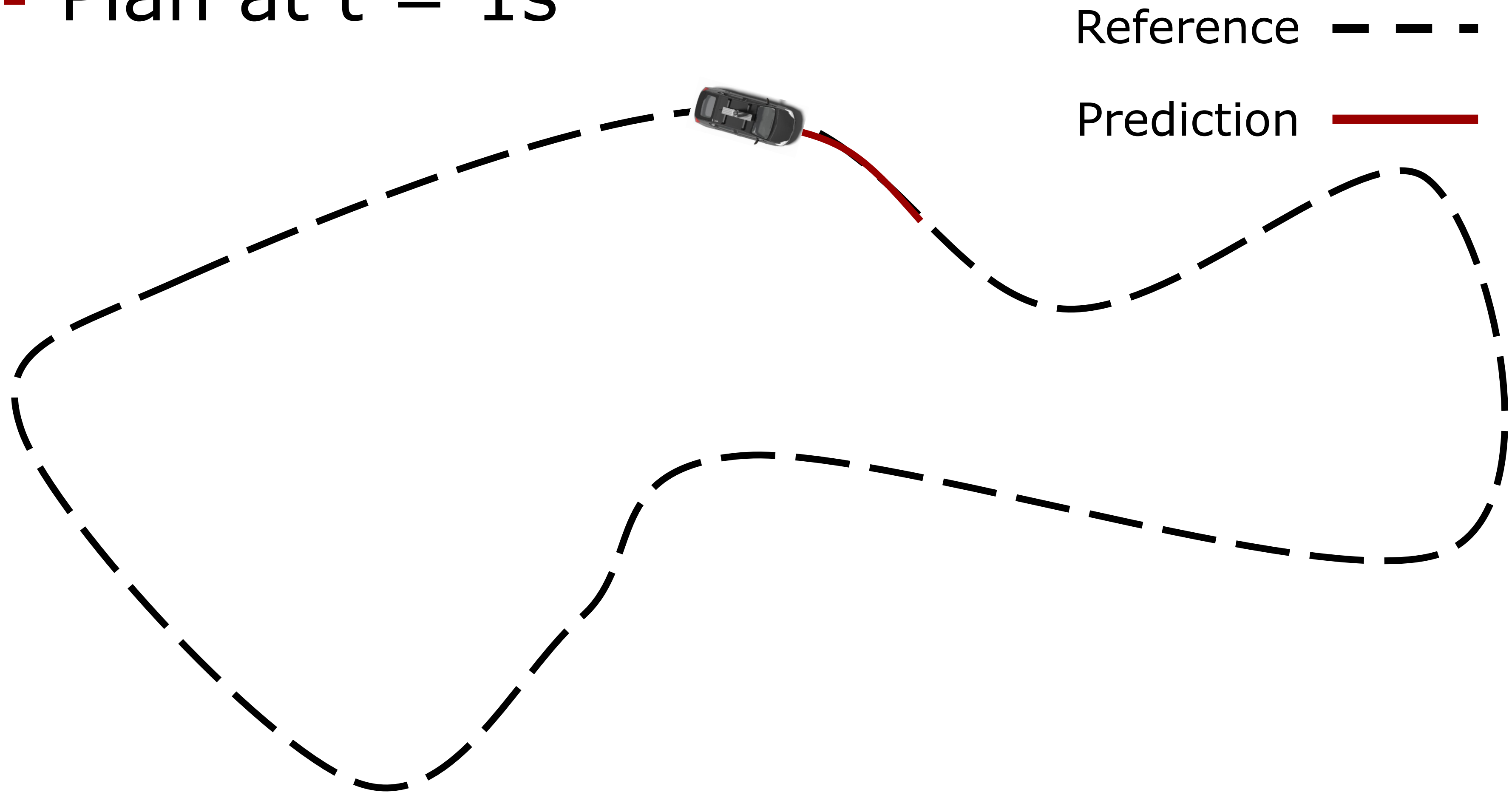
MPC: Schematic View

- Plan at t = 0s



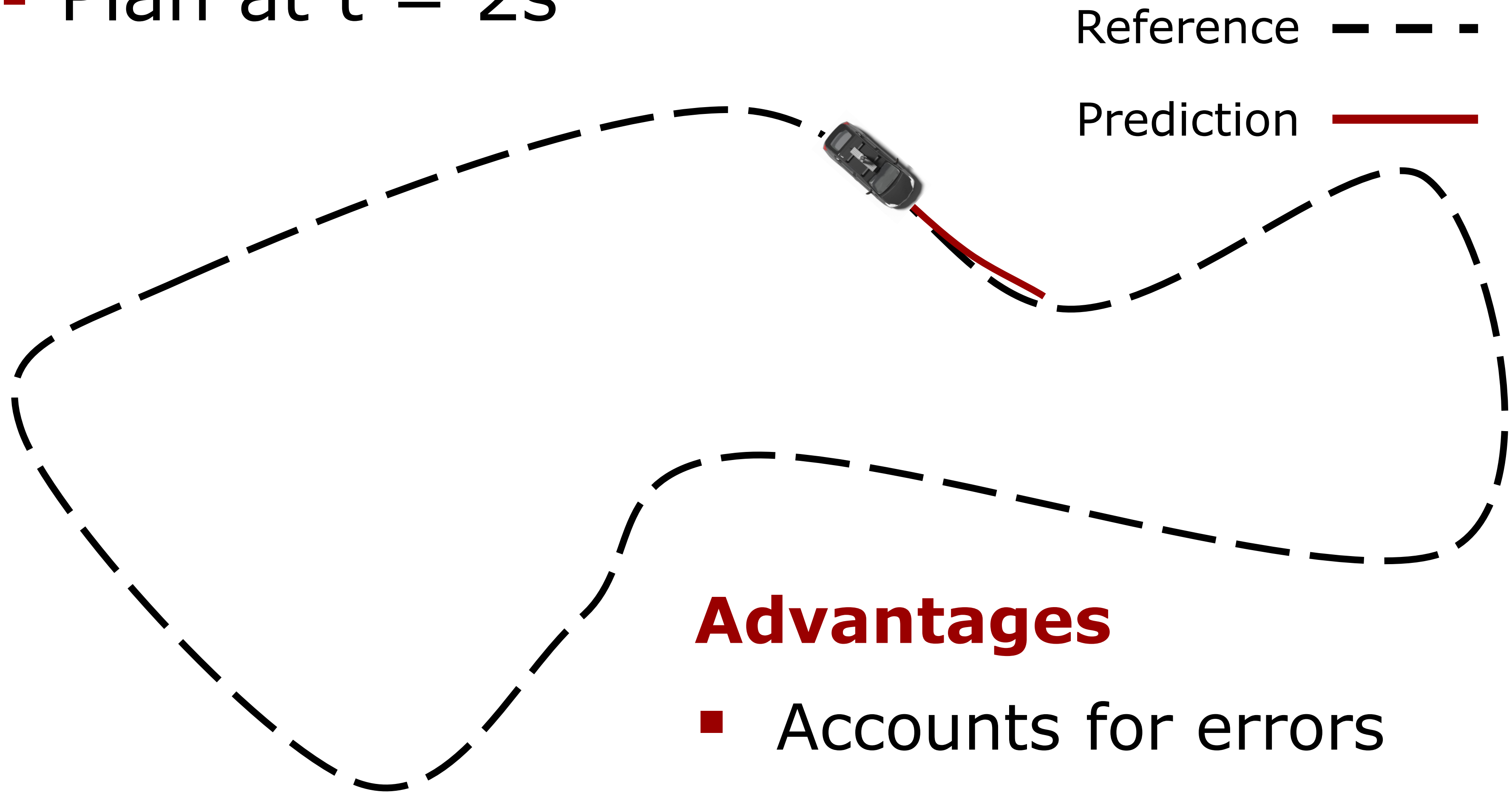
MPC: Schematic View

- Plan at t = 1s



MPC: Schematic View

- Plan at t = 2s



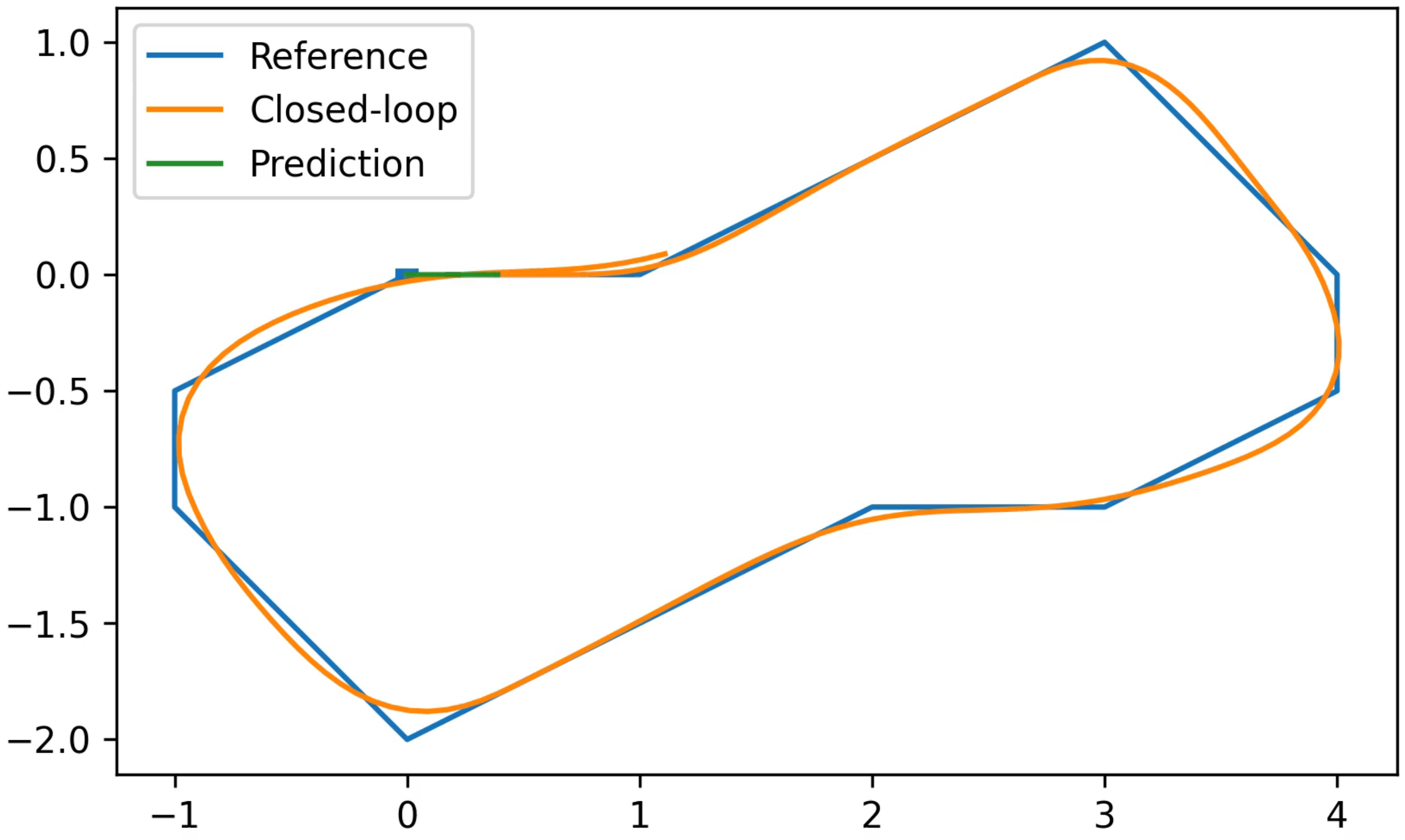
Advantages

- Accounts for errors
- Reduced problem size

MPC: Algorithm

```
Algorithm 1: Model Predictive Control (MPC)
  Input: Objective $J$, Dynamics model $f$, horizon $T$, initial guess $\hat{u}_{1:T}$.
1 $u_{1:T} \leftarrow \hat{u}_{1:T}$;
2 while task not completed do
3     $x_{\text{init}} \leftarrow$ GetCurrentState();
      // Solver is warm-started with the previous solution.
4     $u_{1:T} \leftarrow$ SolveOptimizationProblem($J, f, x_{\text{init}}, T, \underline{u_{1:T}}$);
5     $u \leftarrow$ First($u_{1:T}$);
6     ApplyInput($u$);
```

MPC: Toy Example



MPC Design

Design Parameters

- Prediction model
- Cost function
- Prediction horizon
- Terminal constraints
- ...

MPC Design: Prediction Model

- Trade-off in choice of **model family**: Model **accuracy** vs. **complexity**
- Data-driven approach
  - Collect data of the real system behavior

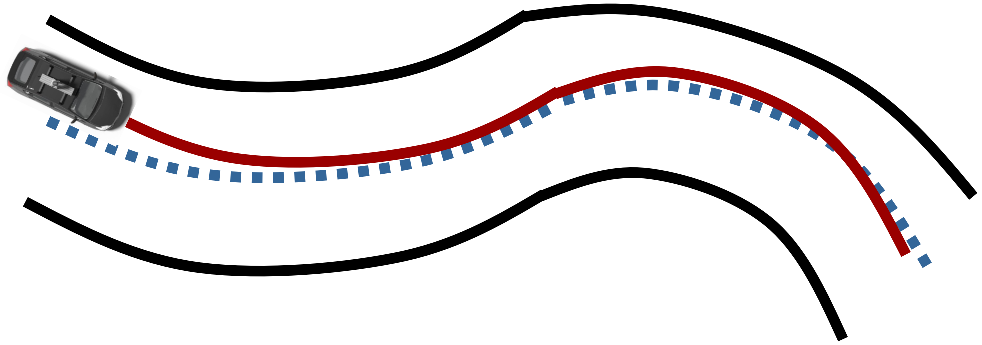
$$\mathcal{D} := \{(x'_d, x_d, u_d) \mid x'_d = f(x_d, u_d), d \in [N_\mathrm{d}]\}$$

  - Optimize the parameters of the model to match the behavior of the real system

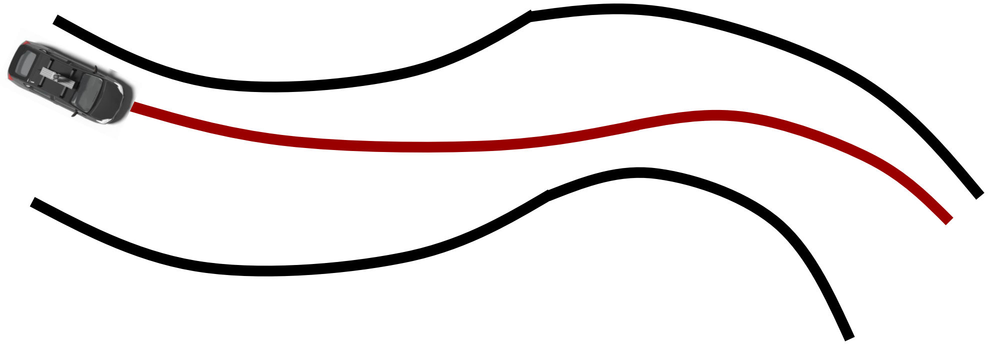
$$\min_\theta \mathcal{L}(\theta; \mathcal{D}) = \min_\theta \sum_{d \in [N_\mathrm{d}]} \|x'_d - f(x_d, u_d; \theta)\|_2^2$$

MPC Design: Cost Function

- Cost function induces optimal behavior
  - Penalize deviation from reference

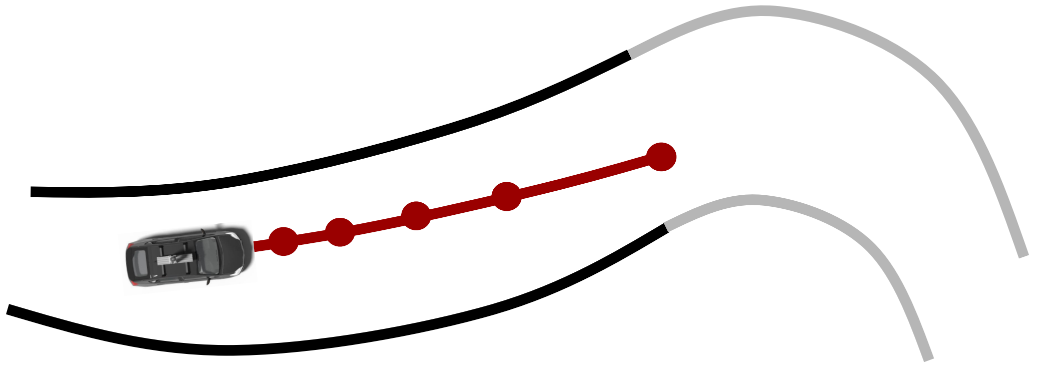


  - Maximizing progress inside track bounds



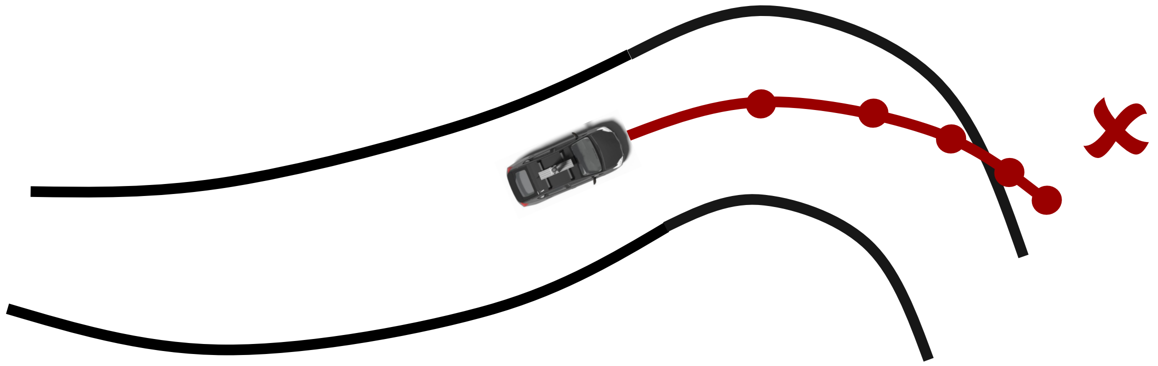
MPC Design: Prediction Horizon

- Short prediction horizon
  - **Pro:** Reduced computation
  - **Con:** Myopic behavior
    - Inefficient
    - potentially **unsafe**



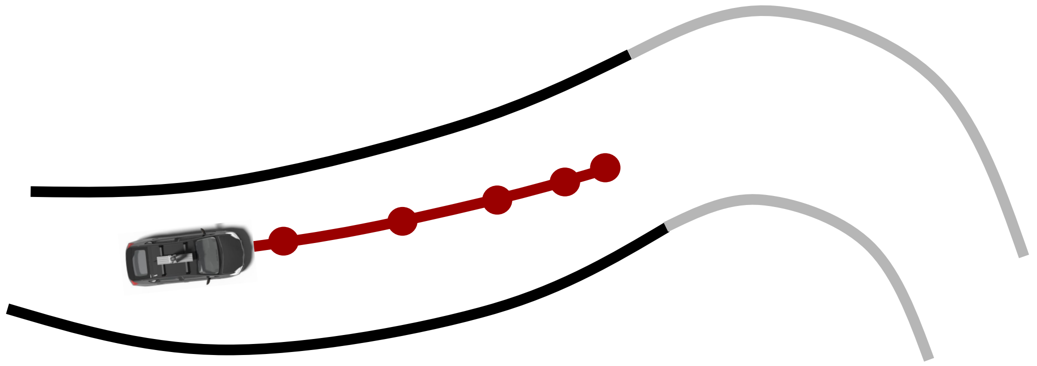
MPC Design: Prediction Horizon

- Short prediction horizon
  - **Pro:** Reduced computation
  - **Con:** Myopic behavior
    - Inefficient
    - potentially **unsafe**



MPC Design: Terminal Constraint

- Additional constraints at the end of the prediction horizon can ensure **recursive feasibility**.
  - Example: Zero-velocity constraint in a static environment.

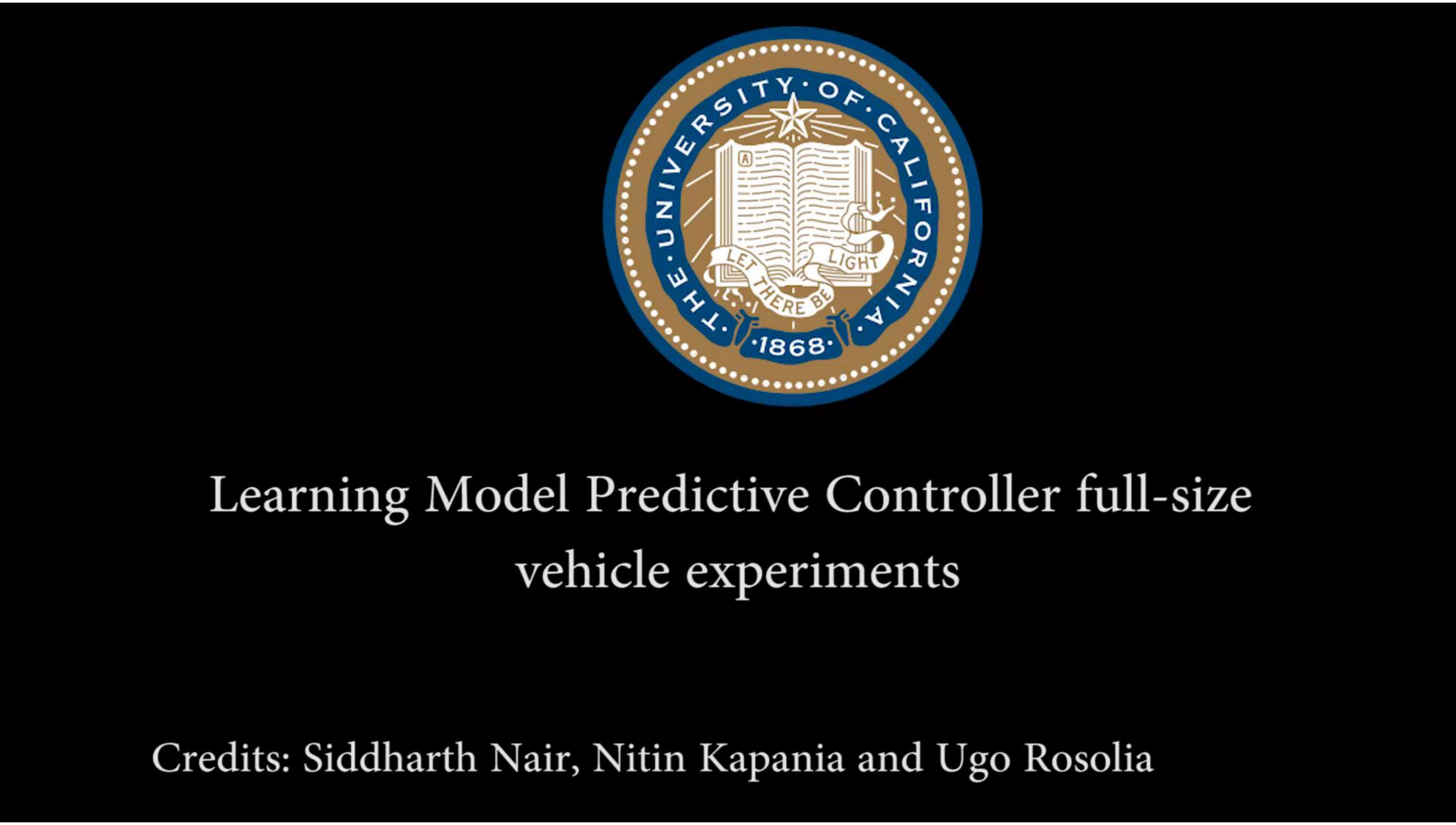


Pros and Cons of MPC

- **Pros**
  - Explicitly handles constraints
  - Preview accounts for future decisions
  - Systematic procedure to derive controllers even for complex systems

- **Cons**
  - Higher computational complexity and memory requirements than reactive controllers

Example: Learning MPC



Learning Model Predictive Controller full-size vehicle experiments

Credits: Siddharth Nair, Nitin Kapania and Ugo Rosolia

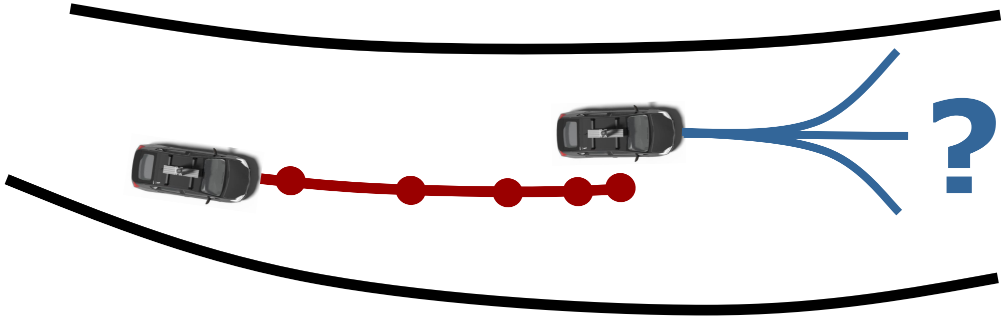
Courtesy: Ugo Rosolia

Example: Whole Body MPC

Courtesy: RSL – ETH Zürich Marco Hutter

Limitation: Interaction

- MPC only models other agents indirectly via the dynamics
  - Other agents treated as dynamic obstacles with constant velocity



- **Challenge:** Other agents also plan!

  ⇒**Decisions are coupled!**

Outlook: Dynamic Games

- Ingredients of a dynamic game
  - Joint dynamics

    $$x_{t+1} = f_t(x_t, u_t^1, \dots, u_t^N)$$

  - Individual costs

    $$J^i(x_{1:T}, u_{1:T}^i, u_{1:T}^{\neg i}) = \sum_{t \in [T]} g_t^i(x_t, u_t^1, \dots, u_t^N)$$

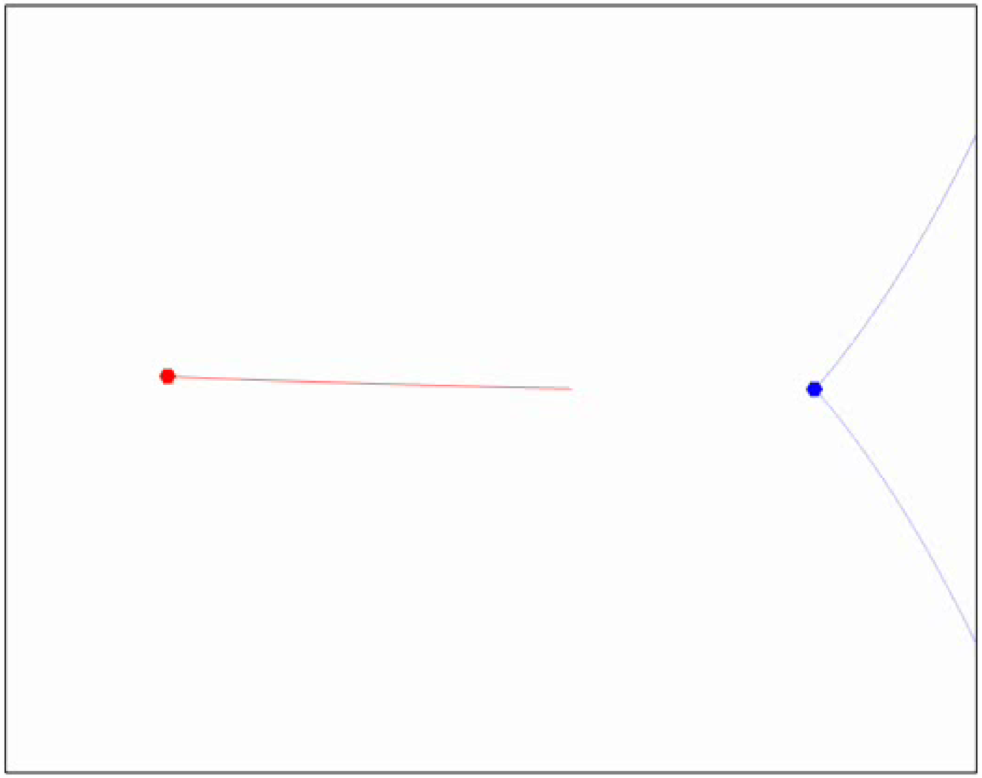
    $$\text{where } u_{1:T}^{\neg i} := (u_{1:T}^j)_{j \in [N] \setminus \{i\}}$$

- Solution: **Nash Equilibrium**
  - No player can unilaterally improve

$$\forall i \in [N] : J^i(x_{1:T}^*, u_{1:T}^*) \leq J^i(x_{1:T}, u_{1:T}^i, u_{1:T}^{\neg i *})$$

Dynamic Game Example: Tag

- 2D point-mass dynamics
- Objectives
  - **P1**:Minimize distance to P2
  - **P2**: Maximize distance to P1



Courtesy: Forrest Laine

Dynamic Game Example: Racing

Courtesy: IfA - ETH Zürich, Alexander Liniger

Summary

- Limitations of reactive control
- **Control as optimization** problem
- **Model-Predictive Control (MPC)** via receding-horizon optimization
- MPC **design parameters**
- Limitations of MPC in the presence of other agents

Resources

- “Predictive Control for Linear and Hybrid Systems” by Borrelli et al.
  Link: http://www.mpc.berkeley.edu/mpc-course-material
- “Numerical Optimization” by Nocedal & Wright
- YouTube channel of Steve Brunton

Thank you for your attention