SELF-DRIVING CARS
CONTROL

Photogrammetry & Robotics Lab

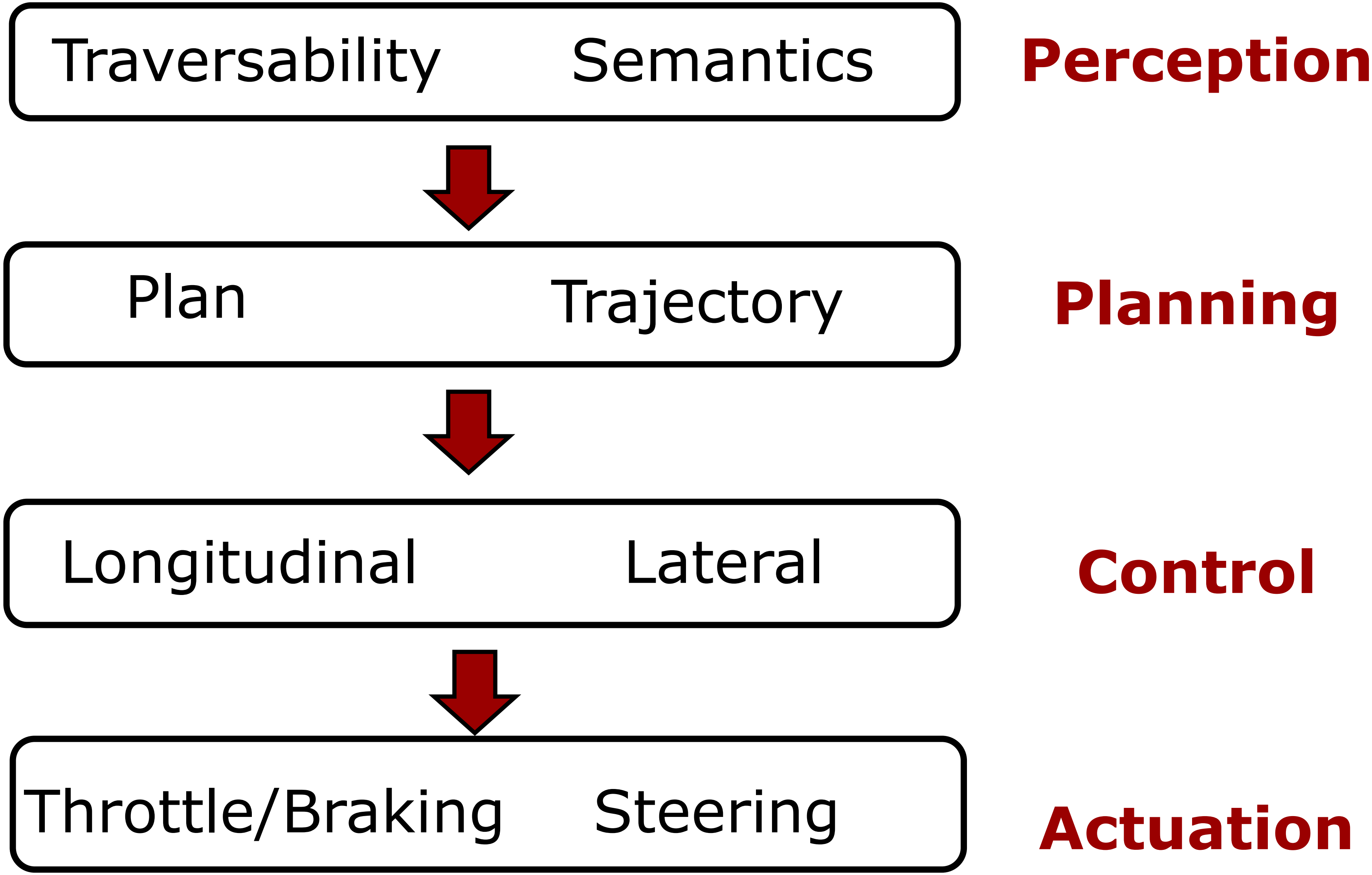
Control for Self-Driving Cars

Jens Behley, Nived Chebrolu

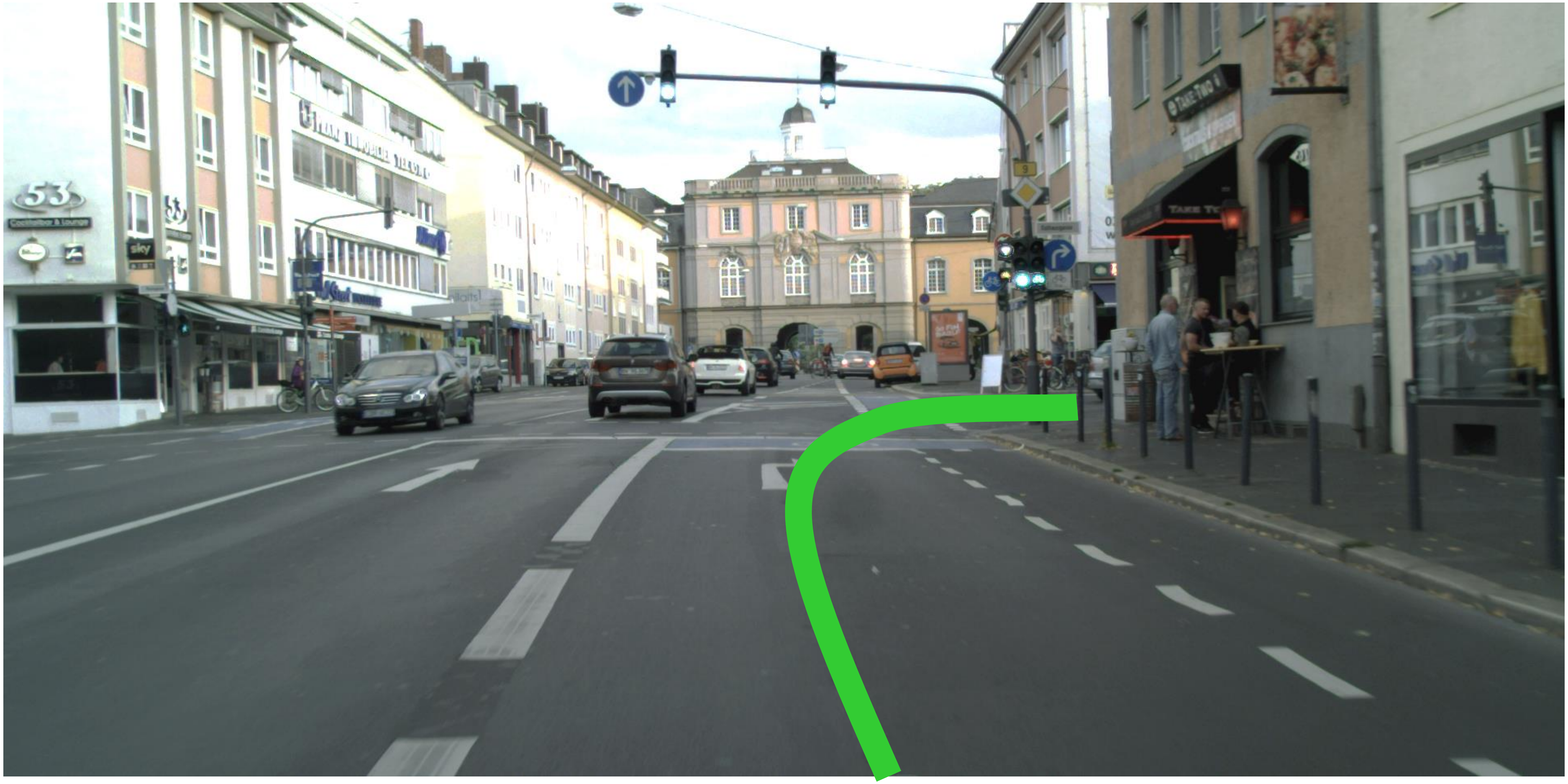
Part of the Course: Techniques for Self-Driving Cars by
C. Stachniss, J. Behley, N. Chebrolu, B. Mersch, I. Bogoslavskyi

Self-Driving Car Scenario

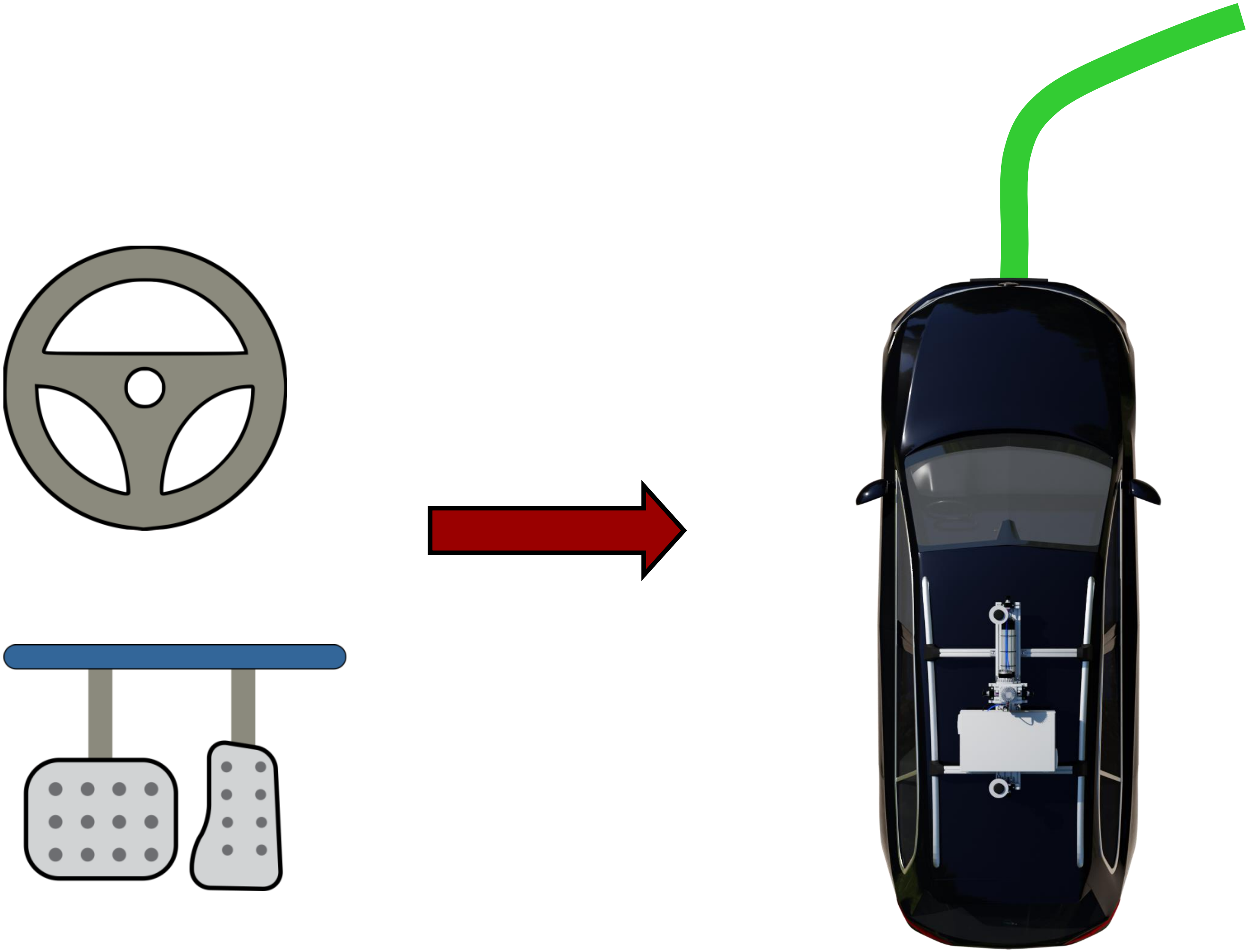
Control Strategy



How to follow a trajectory?



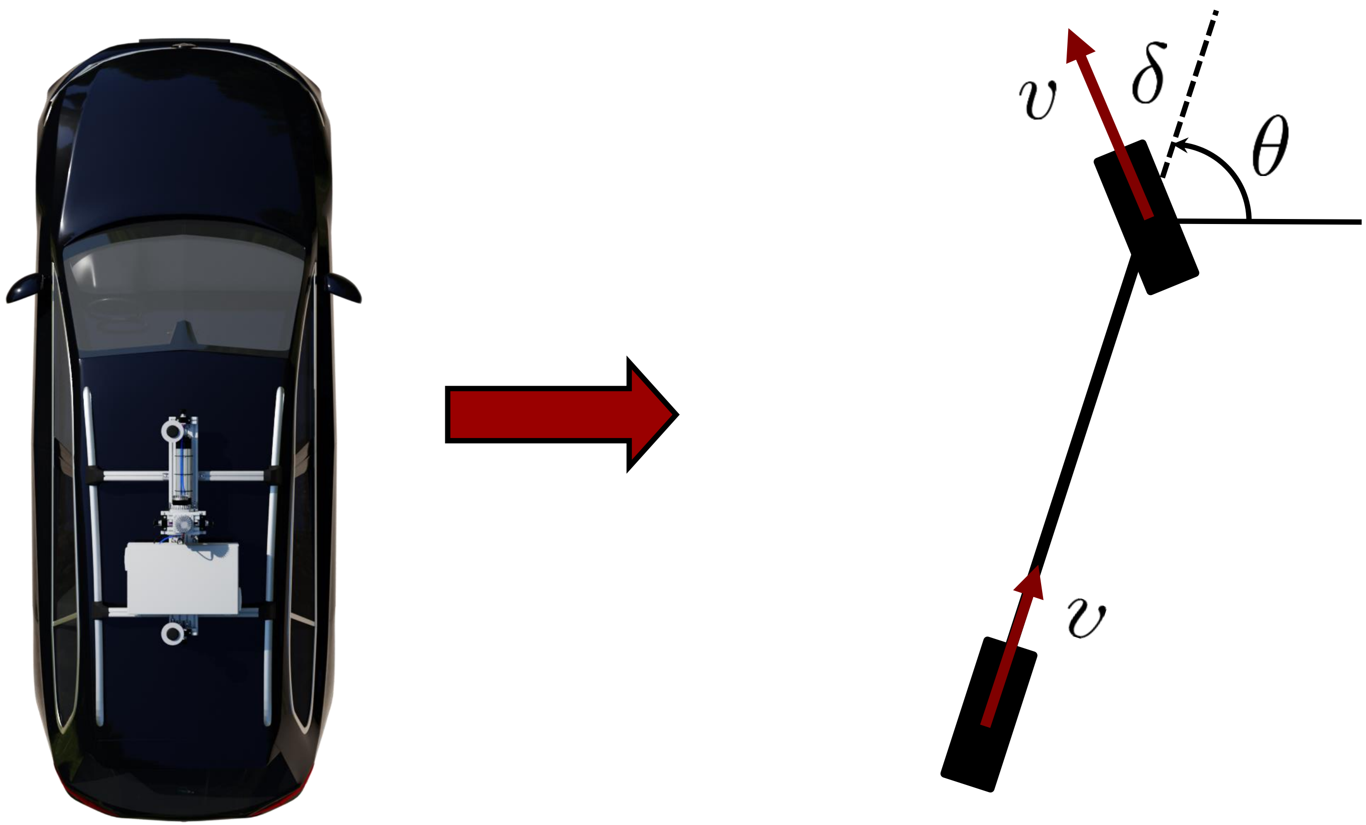
What controls are needed?



Understanding Motion

Kinematic Modelling

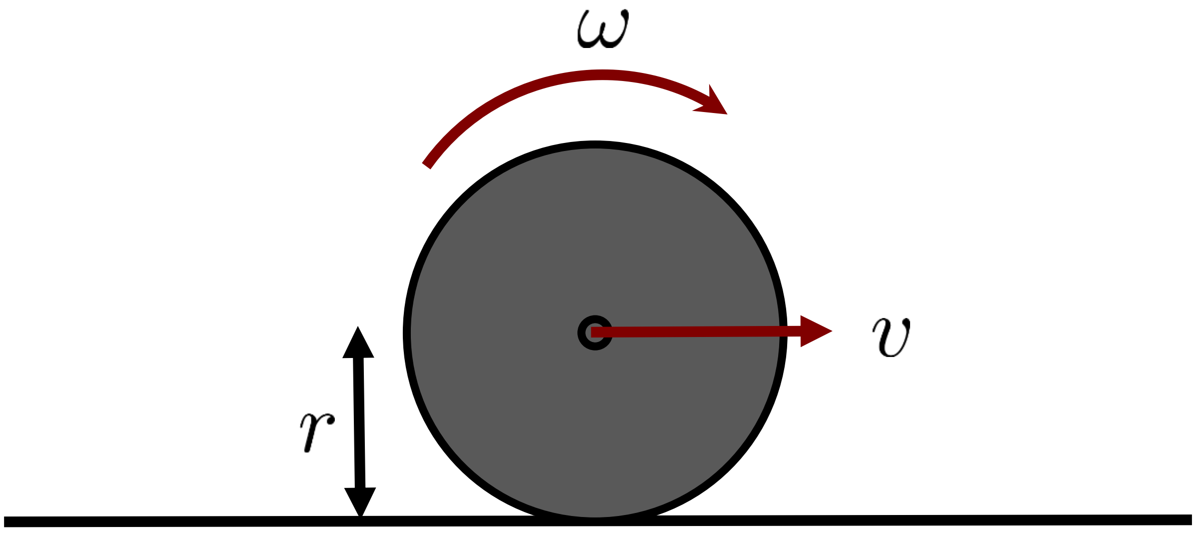
2D Bicycle Model



Rolling Condition for Wheels

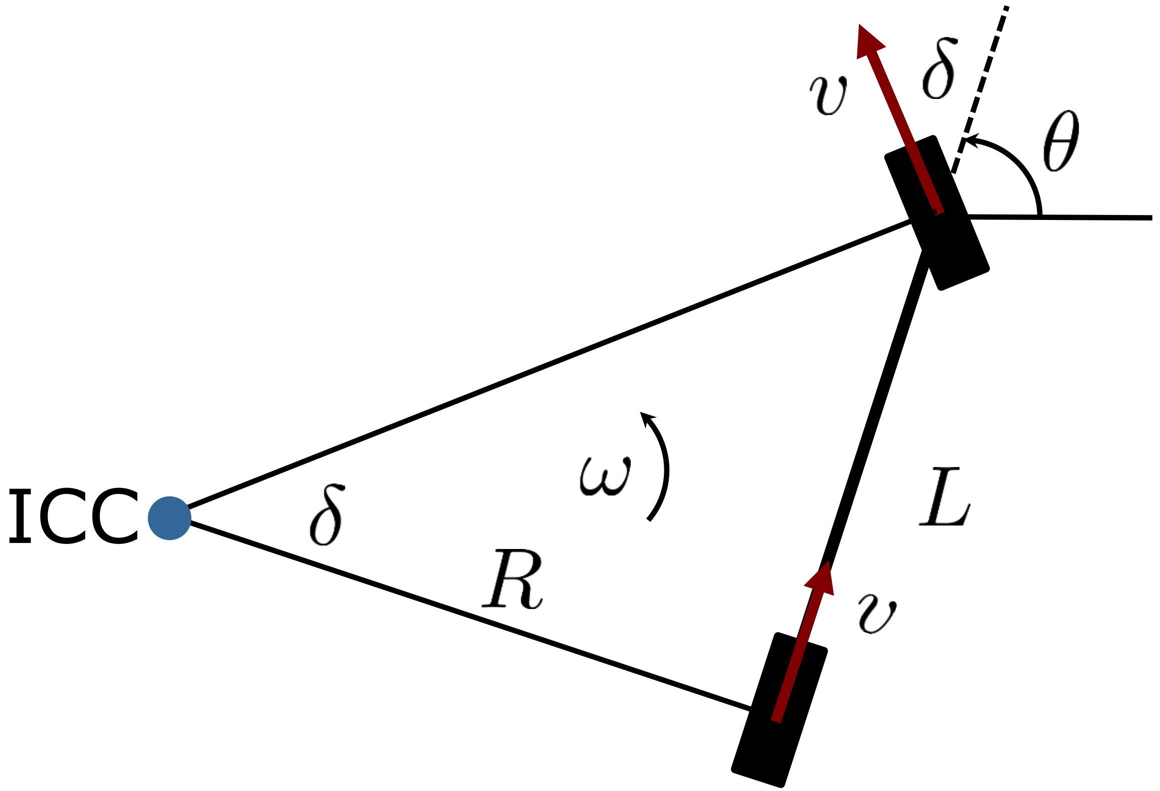
- Kinematic Constraint

$$v = r\omega$$



Instantaneous Center of Curvature

For rolling motion to occur, each wheel has to move along its y-axis



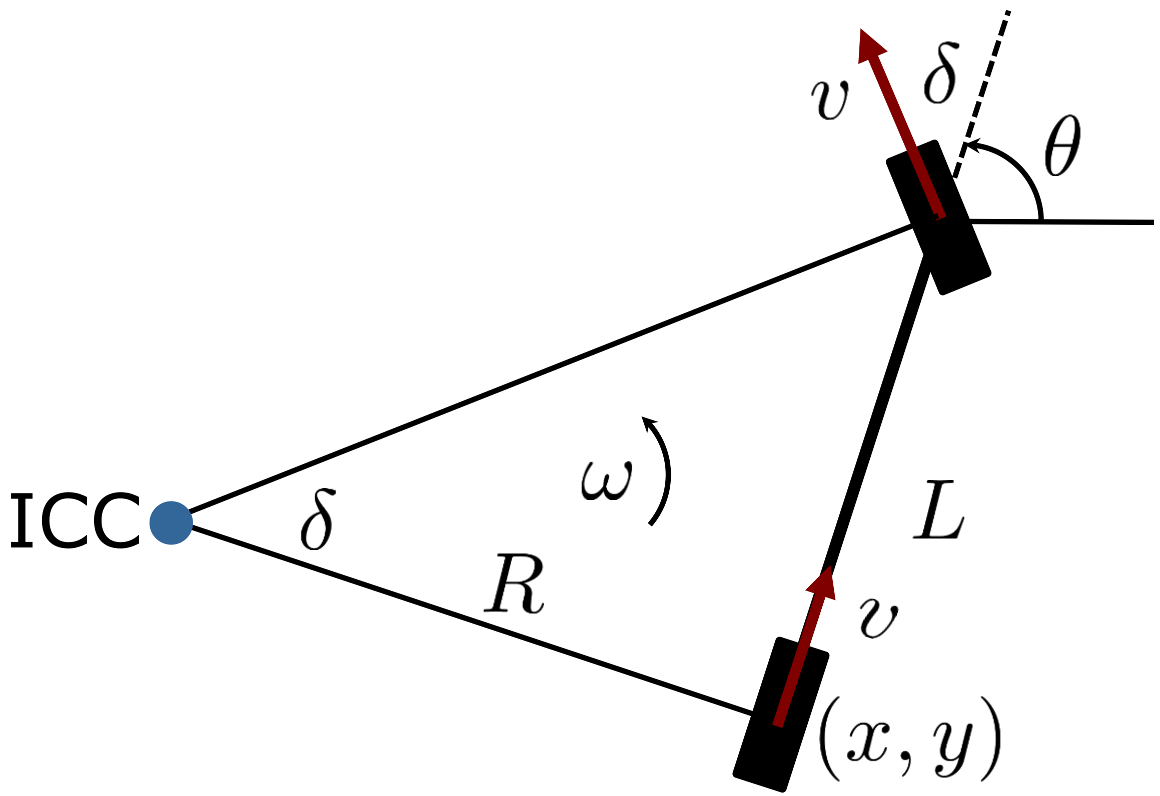
Bicycle Model Kinematics

- Desired point is center of rear axle

$$\dot{x} = v\cos(\theta)$$

$$\dot{y} = v\sin(\theta)$$

$$\dot{\theta} = \frac{v\tan(\delta)}{L}$$



Bicycle Model

State:

$$[x, y, \theta, \delta]^T$$

$$\delta = tan^{-1}(\frac{L}{R})$$

Control:

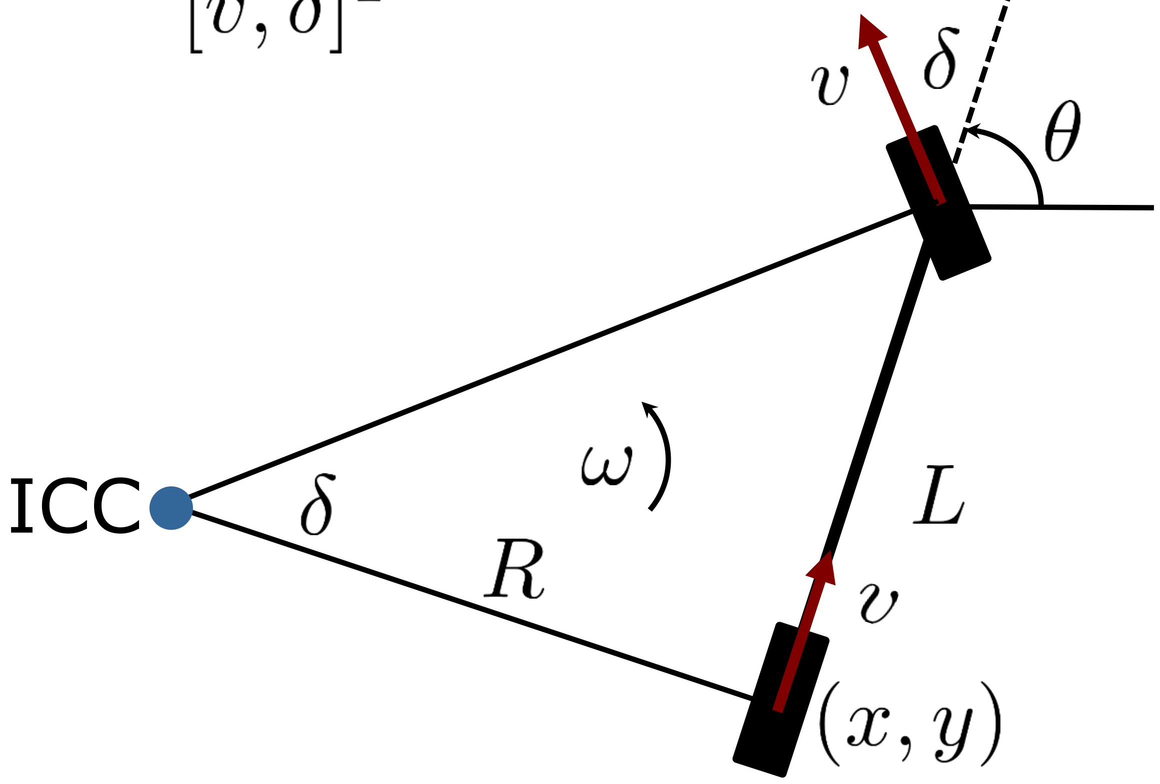
$$[v, \dot{\delta}]^T$$

Kinematics:

$$\dot{x} = v\, cos(\theta)$$

$$\dot{y} = v\, sin(\theta)$$

$$\dot{\theta} = \frac{v\, tan(\delta)}{L}$$

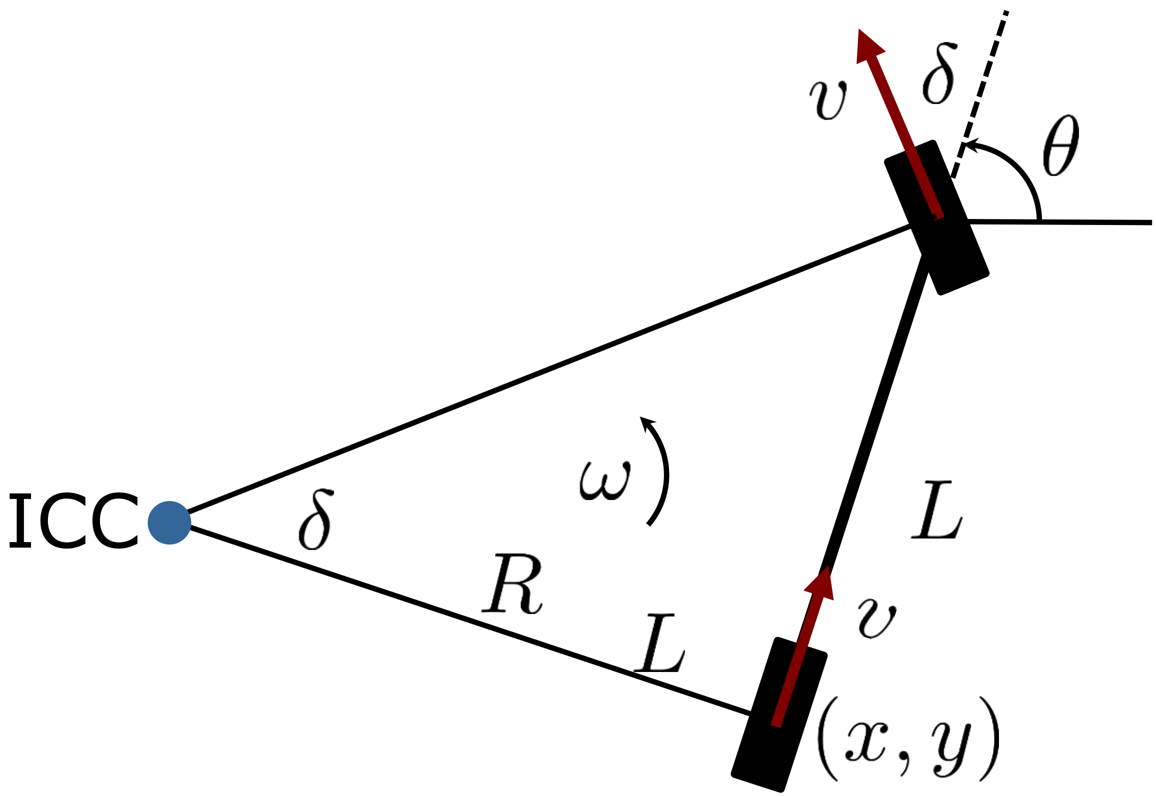


Bicycle Model

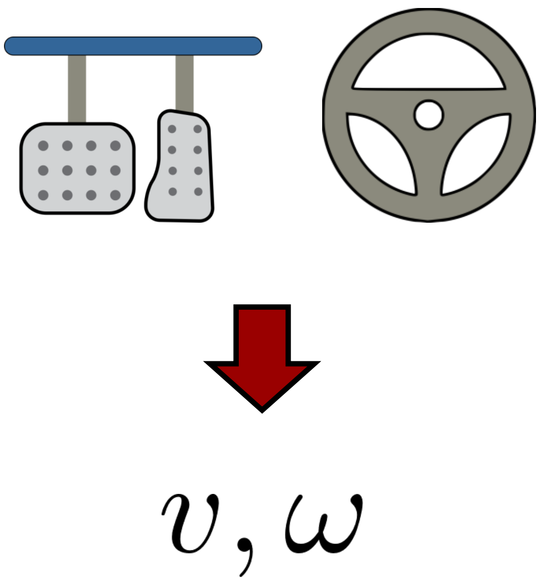
Constraints:

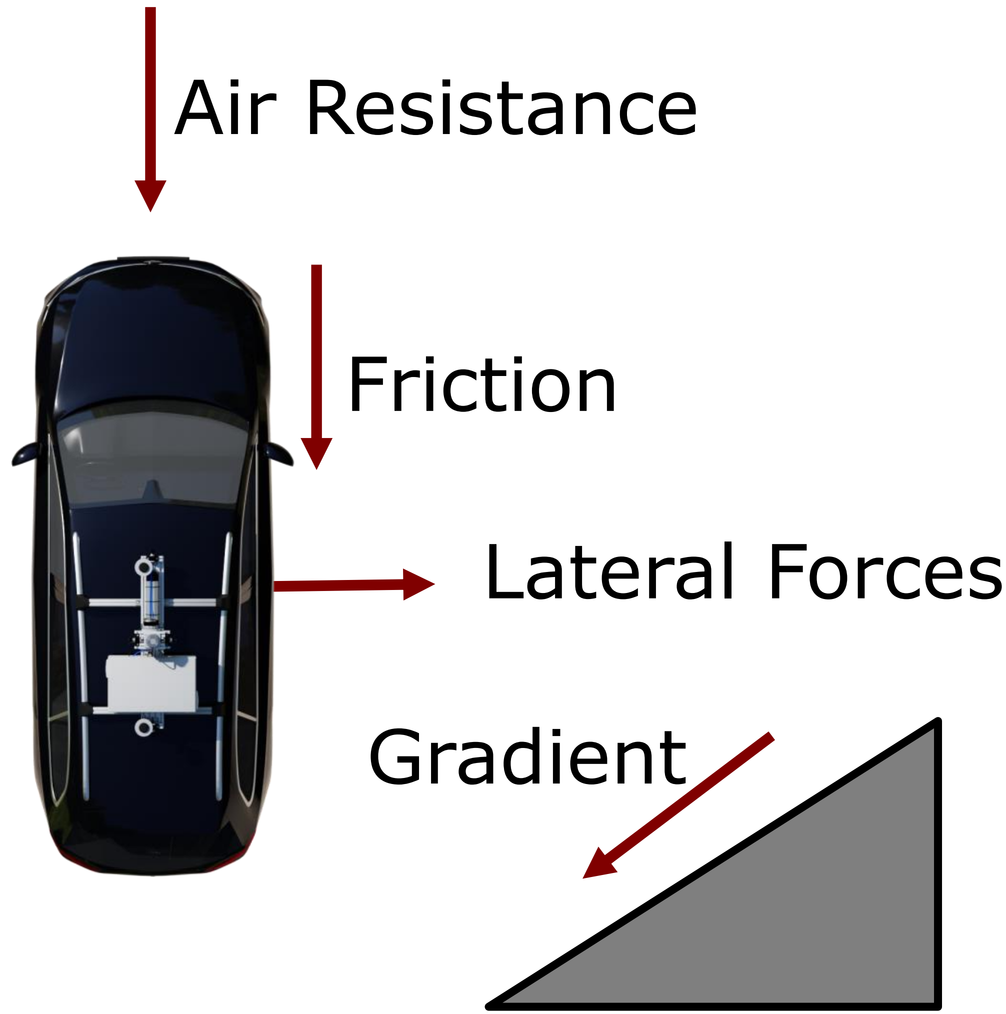
$$v < v_{\max}$$

$$\delta < |\delta_{\max}|$$



Kinematic Vs. Dynamic Modeling

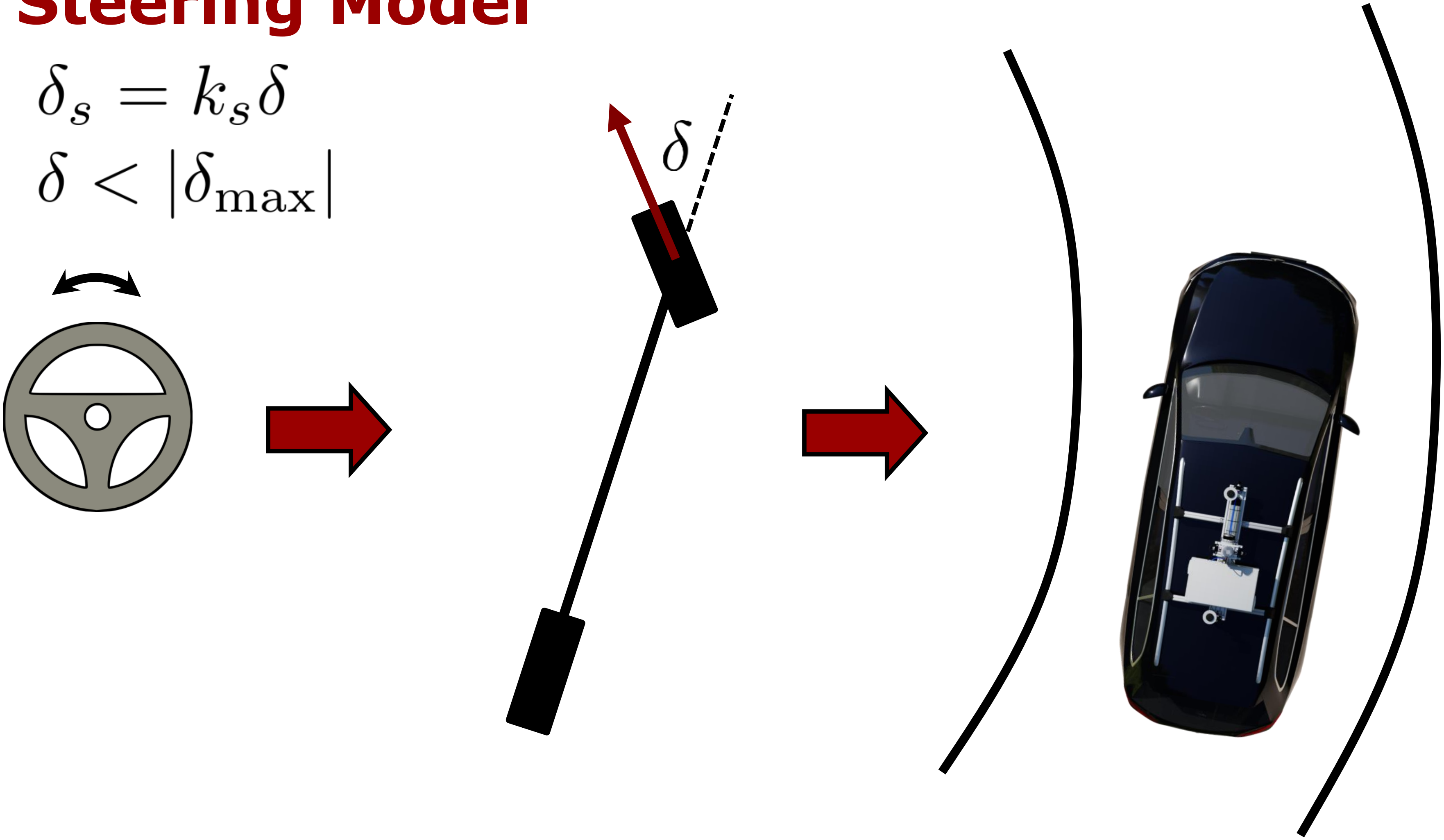




Vehicle Actuation

Steering Model

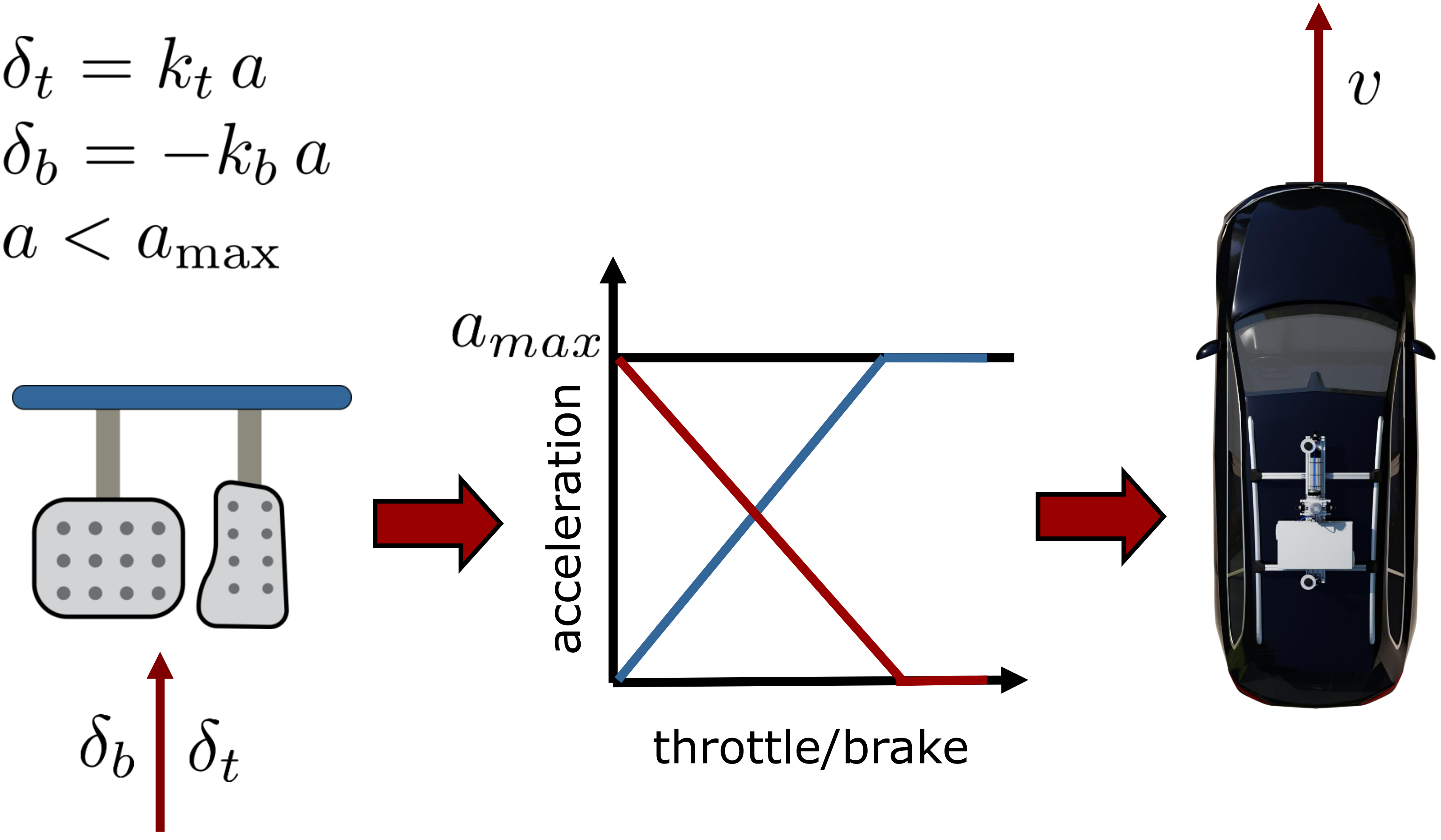
$$\delta_s = k_s \delta$$
$$\delta < |\delta_{\max}|$$



Vehicle Actuation

Throttle/Brake

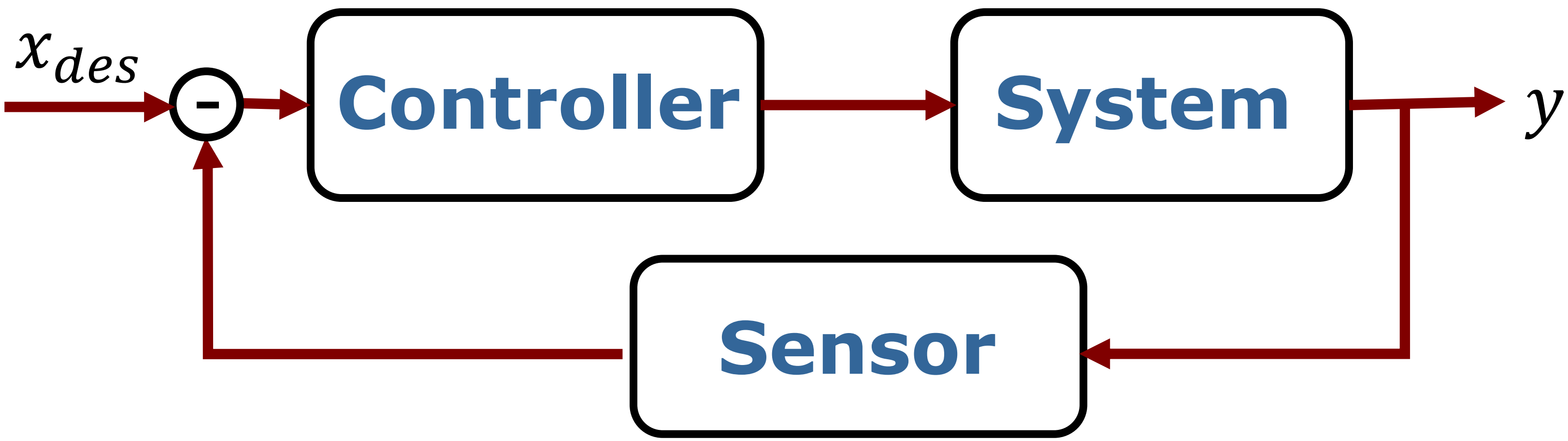
$$\delta_t = k_t \, a$$
$$\delta_b = -k_b \, a$$
$$a < a_{\max}$$



Feedback Control

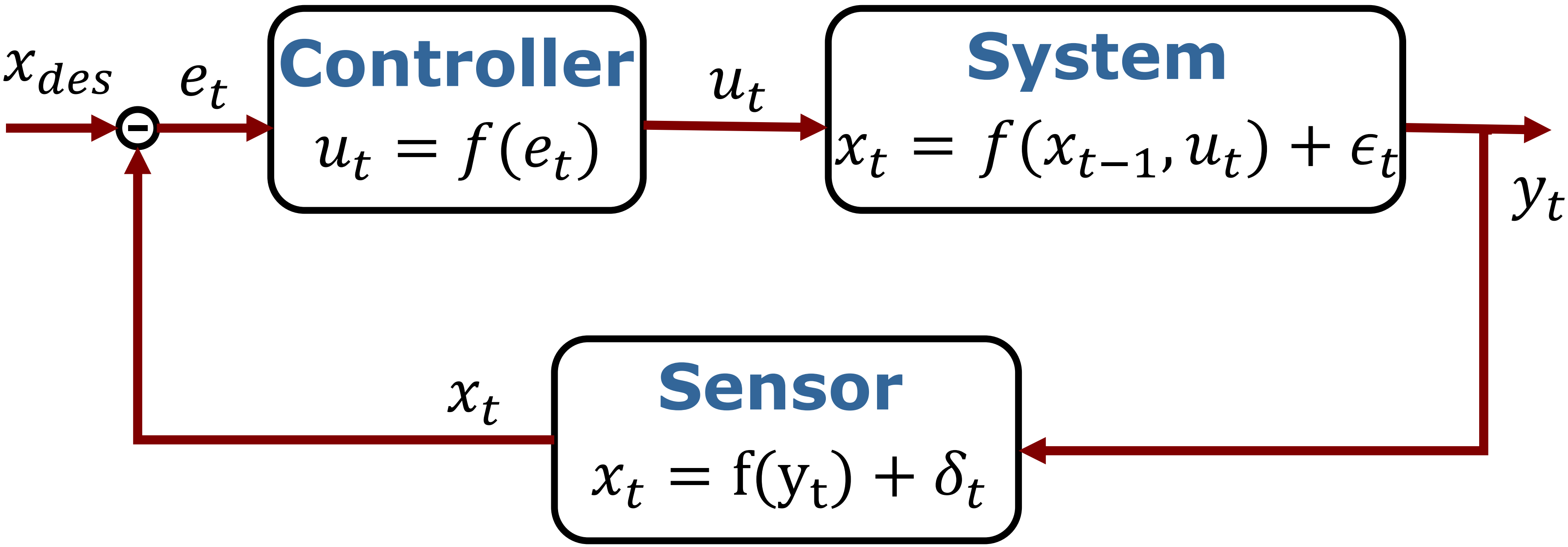
Open Loop vs. Feedback Control

(a) Open loop control

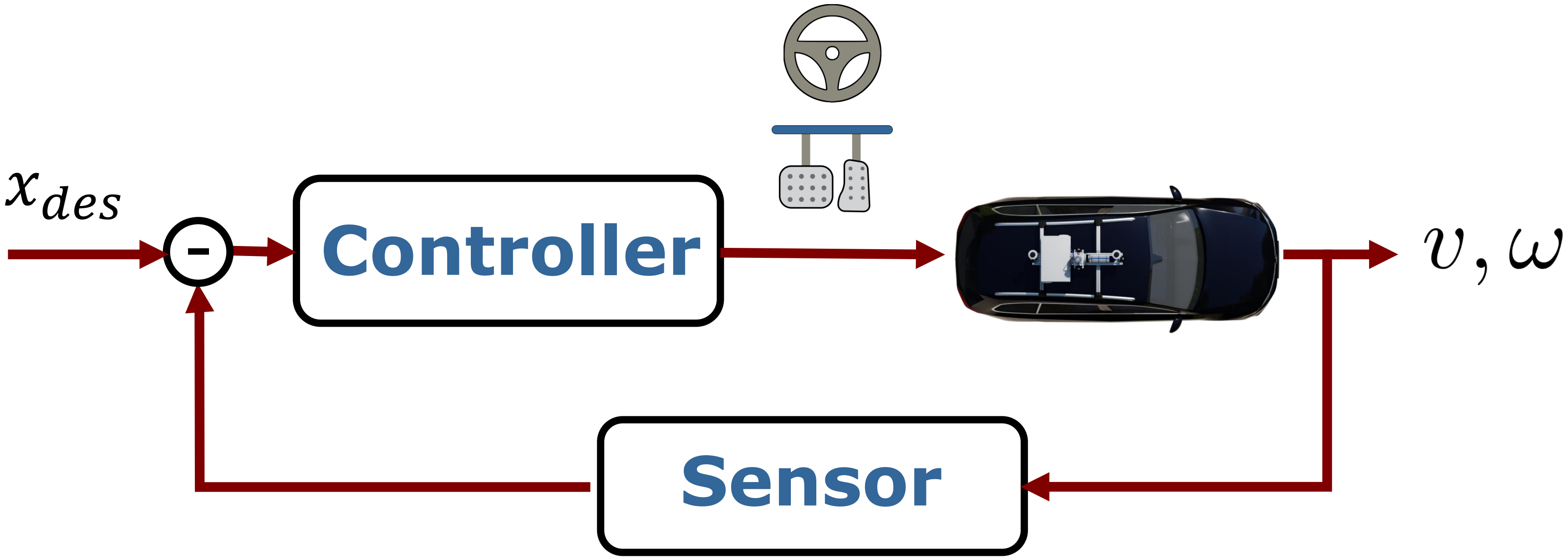


(b) Feedback control

Feedback Control



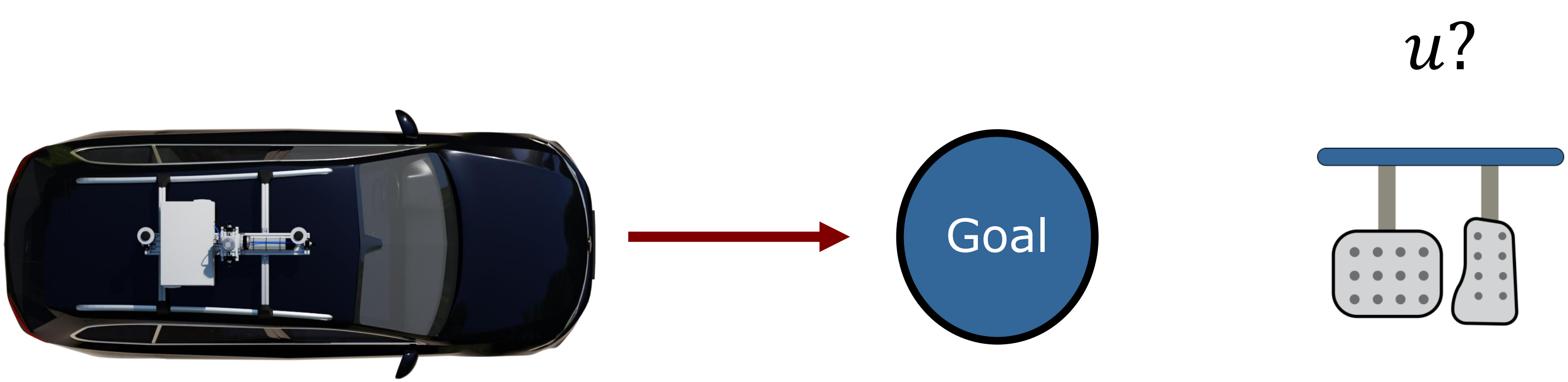
Feedback Control



PID Controller

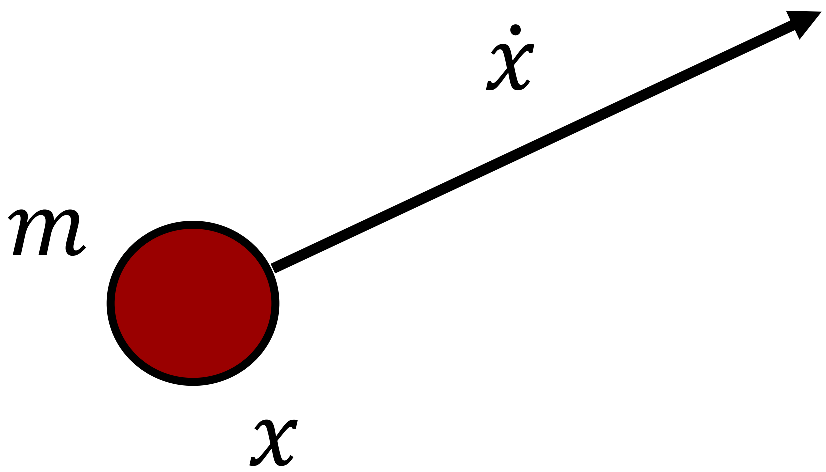
Position Control Task

- Move the robot to the desired goal location x_{des}
- How to generate the suitable control signal u ?
- Robot location estimated via sensor measurements z



Kinematics For A Point Mass

- Consider the robot as a point mass
- Moving freely in 1D space



Position Control Task

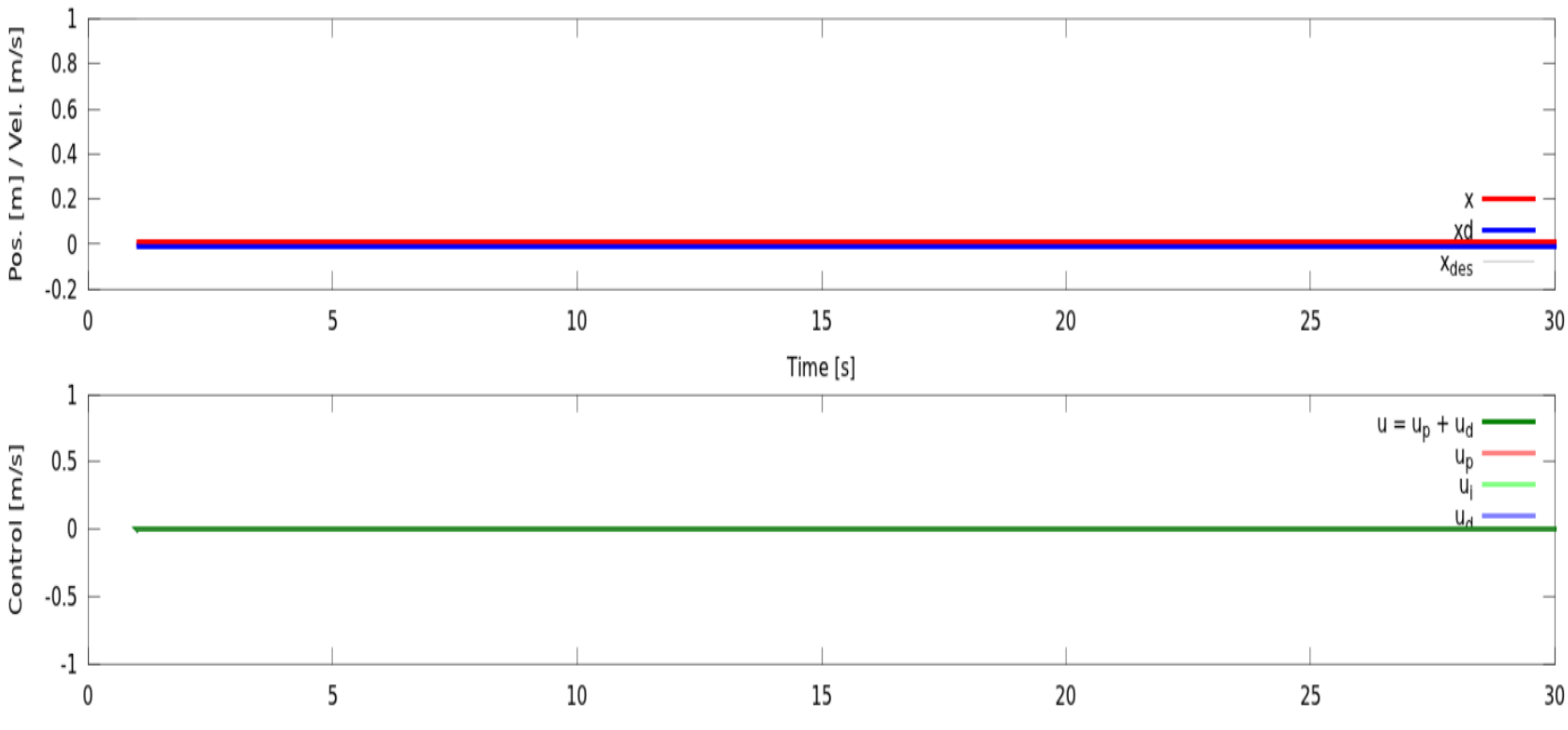
- Position control task is to reach the desired position $x_{des} = 1$ and stop there

- At each time instant, we apply a control u_t

- How to achieve this task using a PID controller?

Kinematics of a rigid body

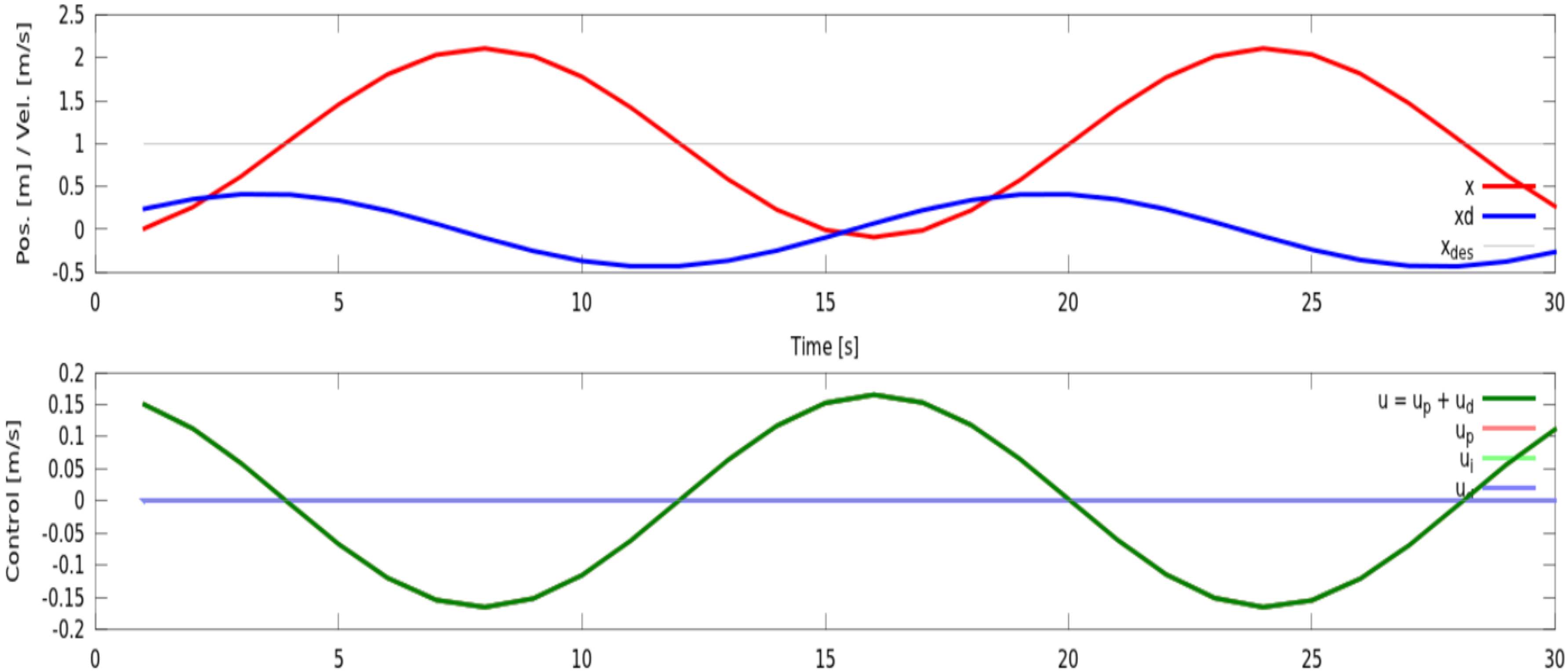
- System model : $x_t = x_{t-1} + \dot{x}\Delta t$
- Initial state: $x_0 = 0, \dot{x}_0 = 0$



P Control

- Proportional control law

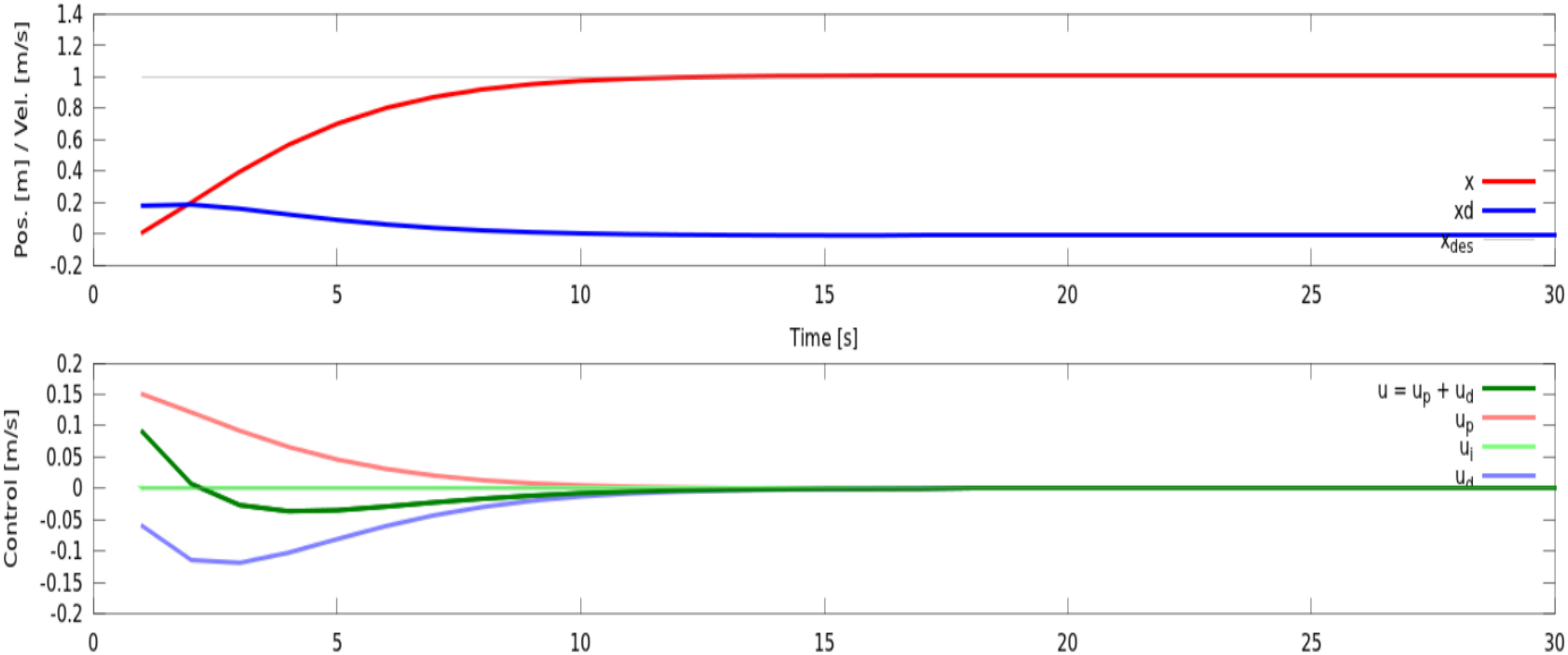
$$u_t = K_P(x_{des} - x_t)$$



PD Control

- Proportional-derivative control law

$$u_t = K_P(x_{des} - x_t) + K_D(\dot{x}_{des} - \dot{x}_t)$$

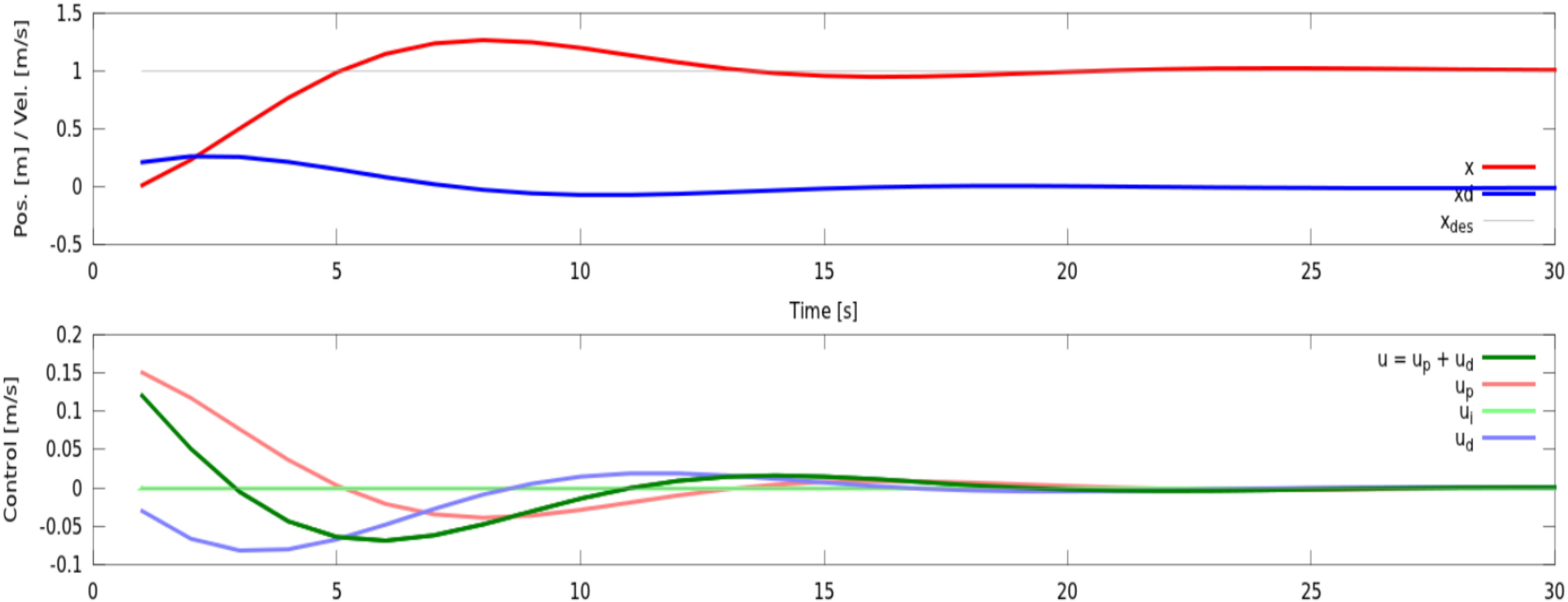


PD Control

- Proportional-derivative control law

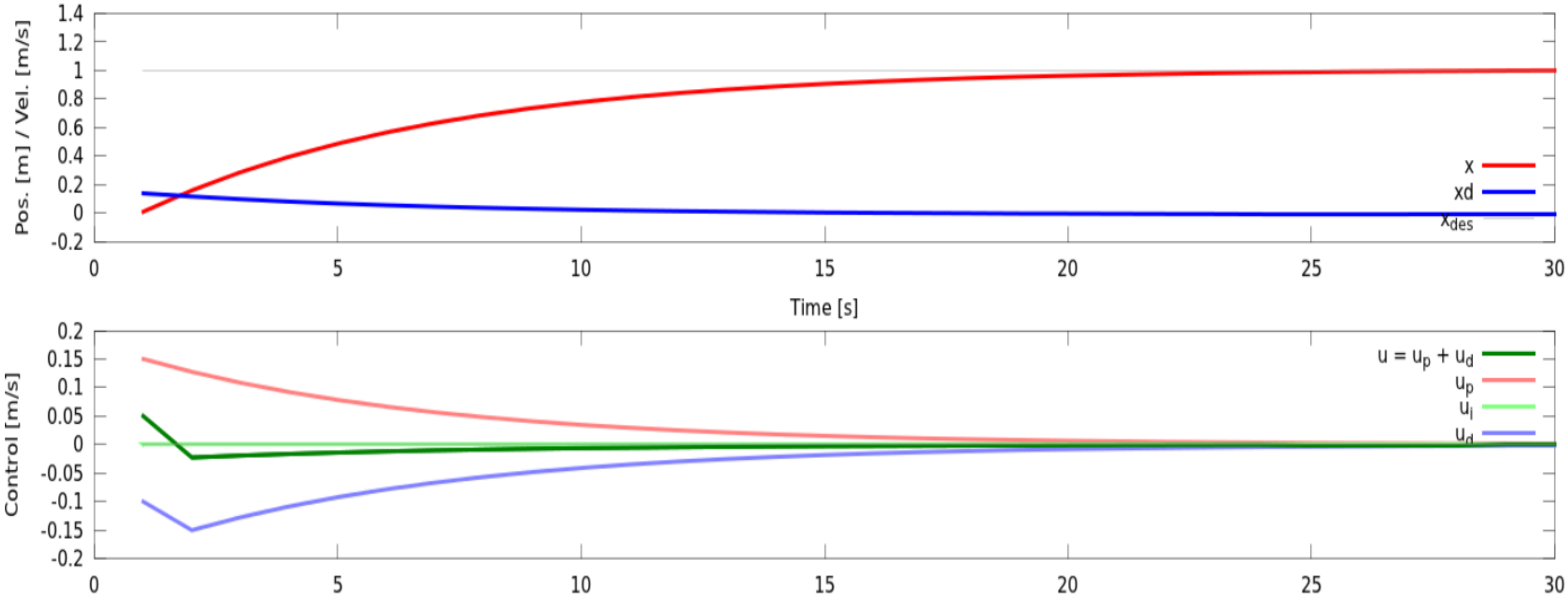
$$u_t = K_P(x_{des} - x_t) + K_D(\dot{x}_{des} - \dot{x}_t)$$

- What happens with high gains?



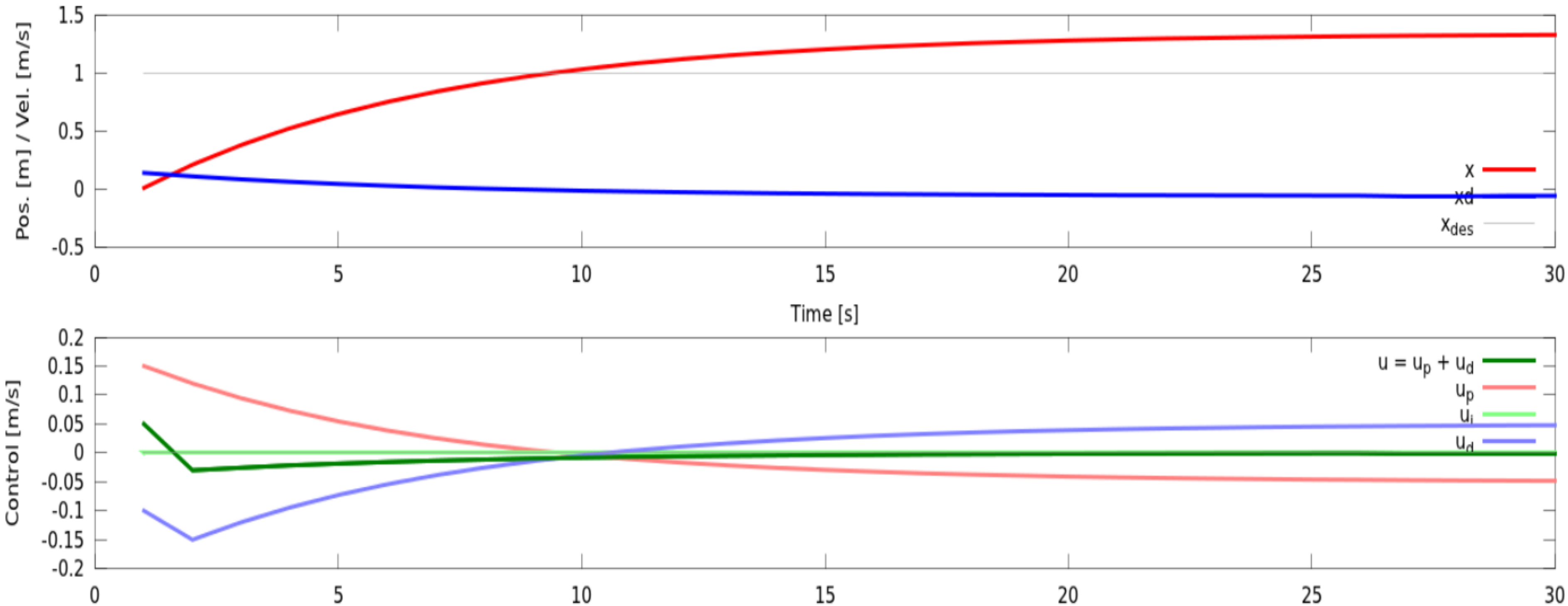
PD Control

- Proportional-derivative control law
  $$u_t = K_P(x_{des} - x_t) + K_D(\dot{x}_{des} - \dot{x}_t)$$
- What happens with low gains?



PD Control

- What happens when there is a systematic bias?
- Ex: robot wheels are not same size …



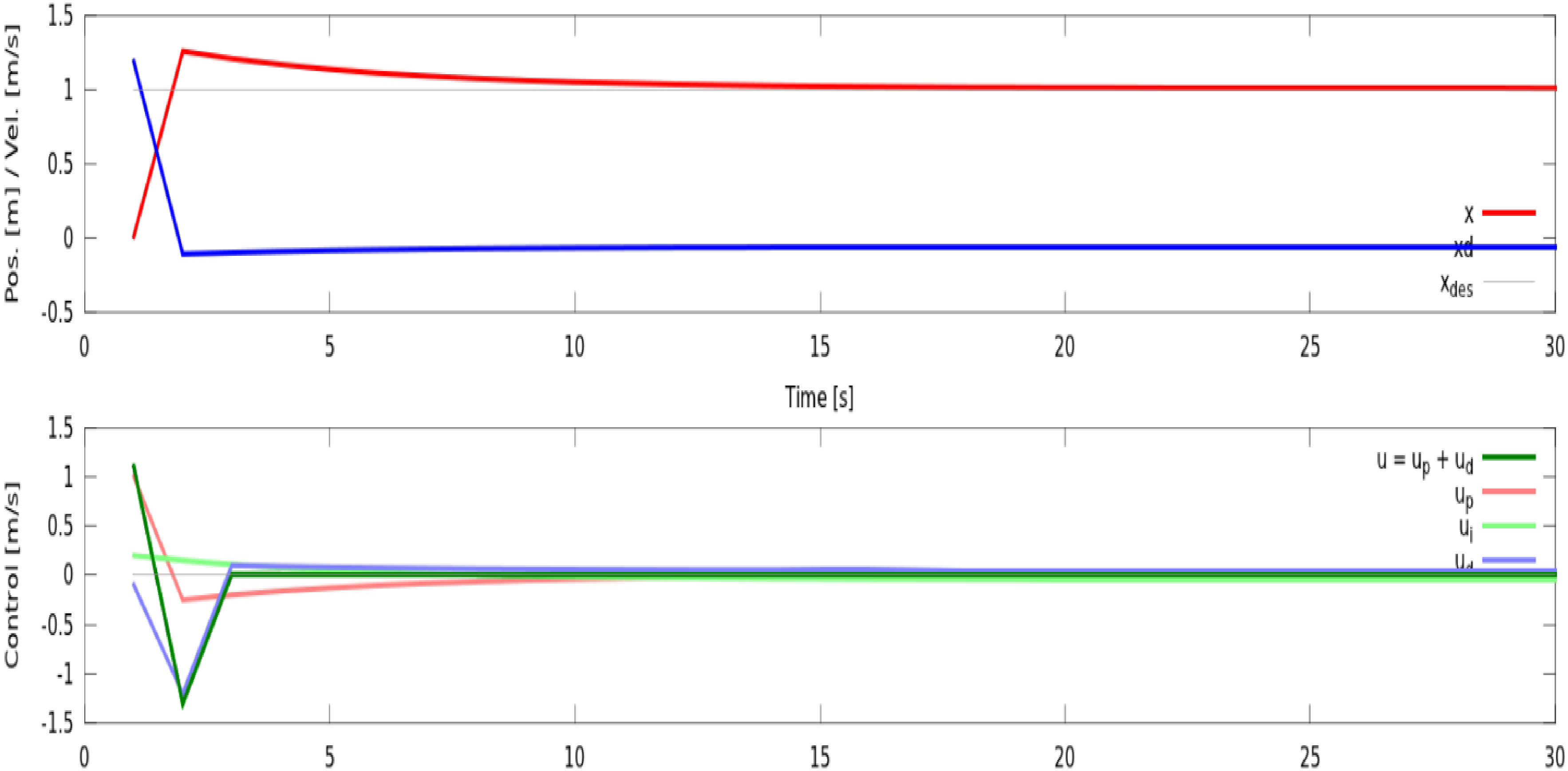
PID Control

- **Idea:** Estimate the systematic error …

$$u_t = K_P(x_{des} - x_t) + K_D(\dot{x}_{des} - \dot{x}_t) + K_I \int_0^t (x_{des} - x_t)dt$$

PID Controller

- **Idea:** Estimate the systematic error …



PID Controller

- **Idea:** Estimate the systematic error …

$$u_t = K_P(x_{des} - x_t) + K_D(\dot{x}_{des} - \dot{x}_t) + K_I \int_0^t (x_{des} - x_t)dt$$

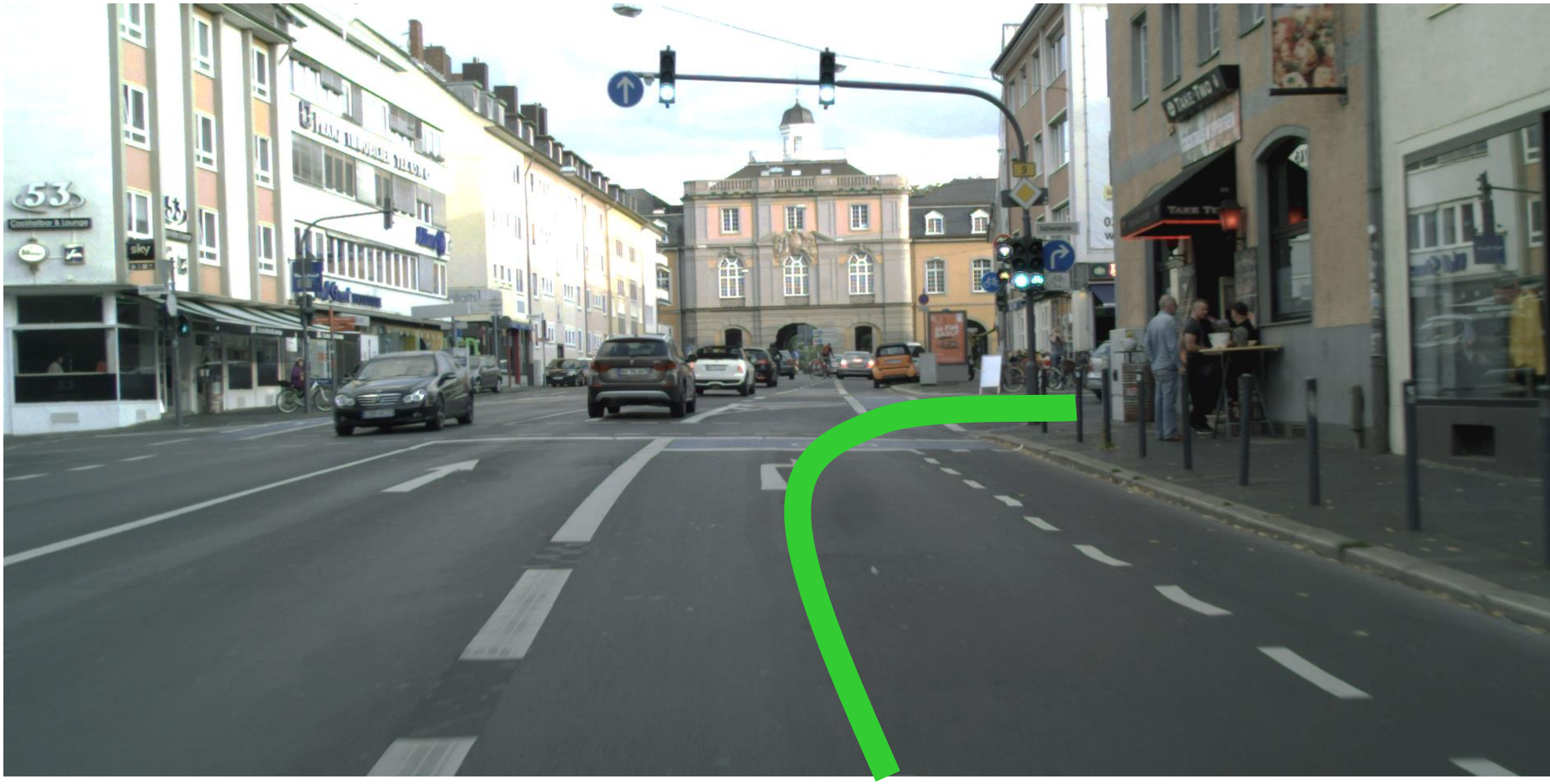
- Reasonable for steady state system
- May be dangerous to error build up (wind-up effect)

PID Control - Summary

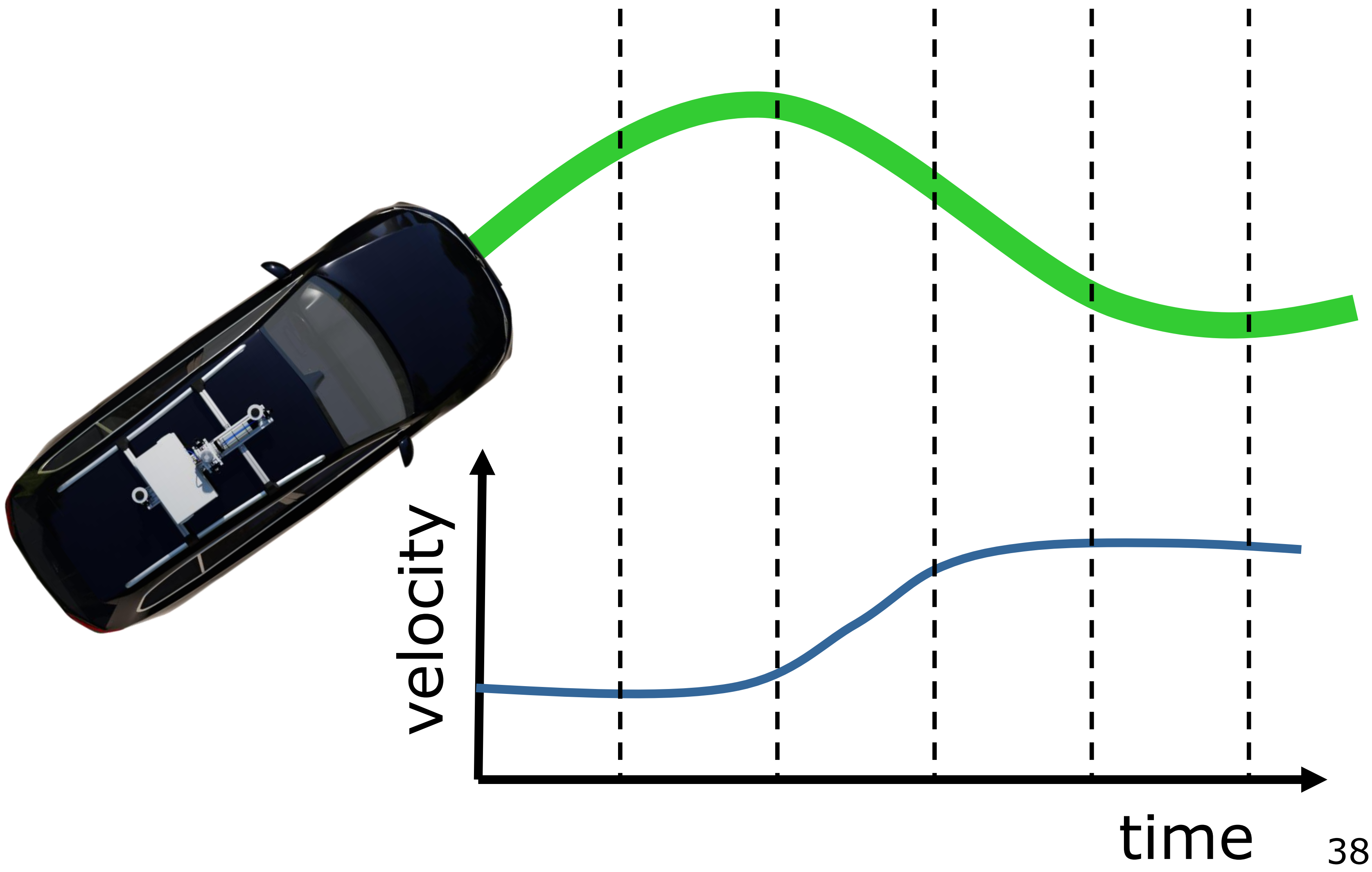
- P = simple proportional control, sufficient in most cases.
- PD = reduce overshoot (e.g. when acceleration can be controlled)
- PI = compensate for systematic error/bias
- PID = combination of the above properties.

Following A Trajectory

How to follow a trajectory?



Longitudinal Control



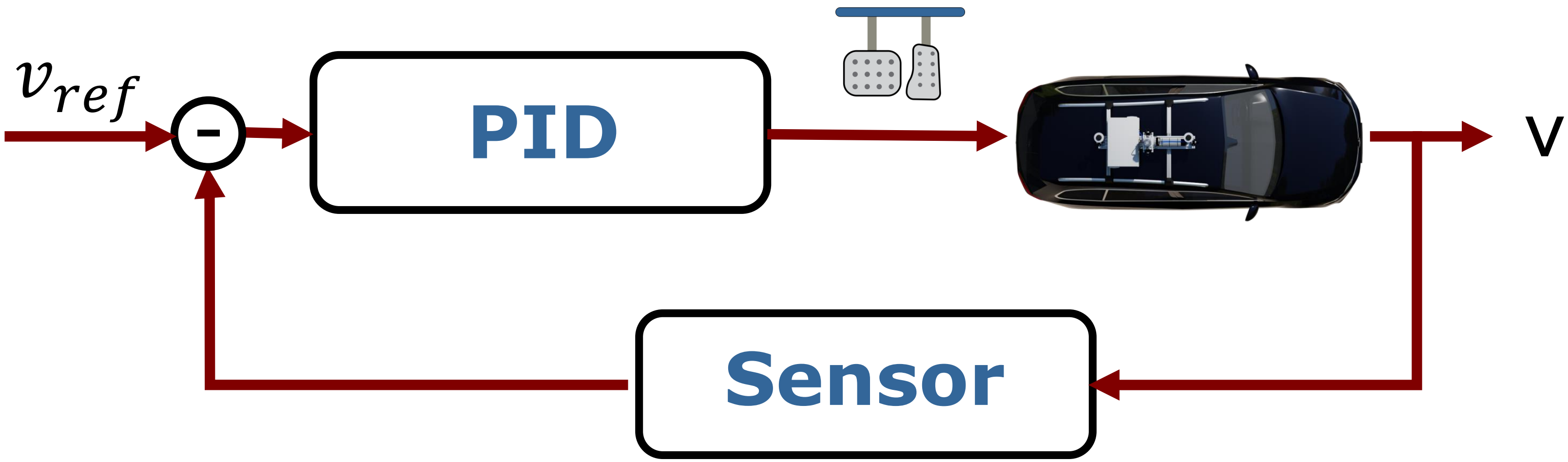
Longitudinal PID Controller

Reference velocity

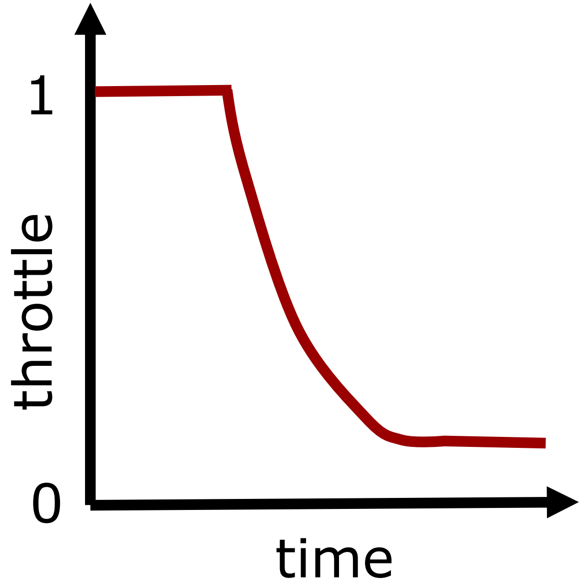
Desired acceleration

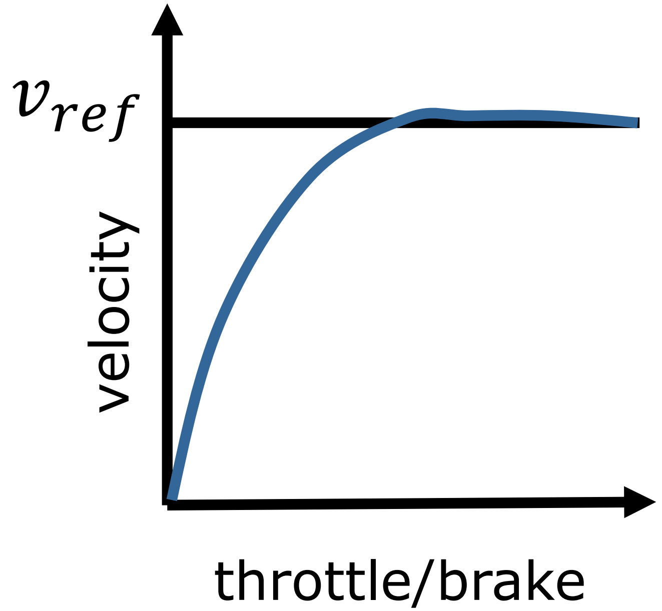
car velocity

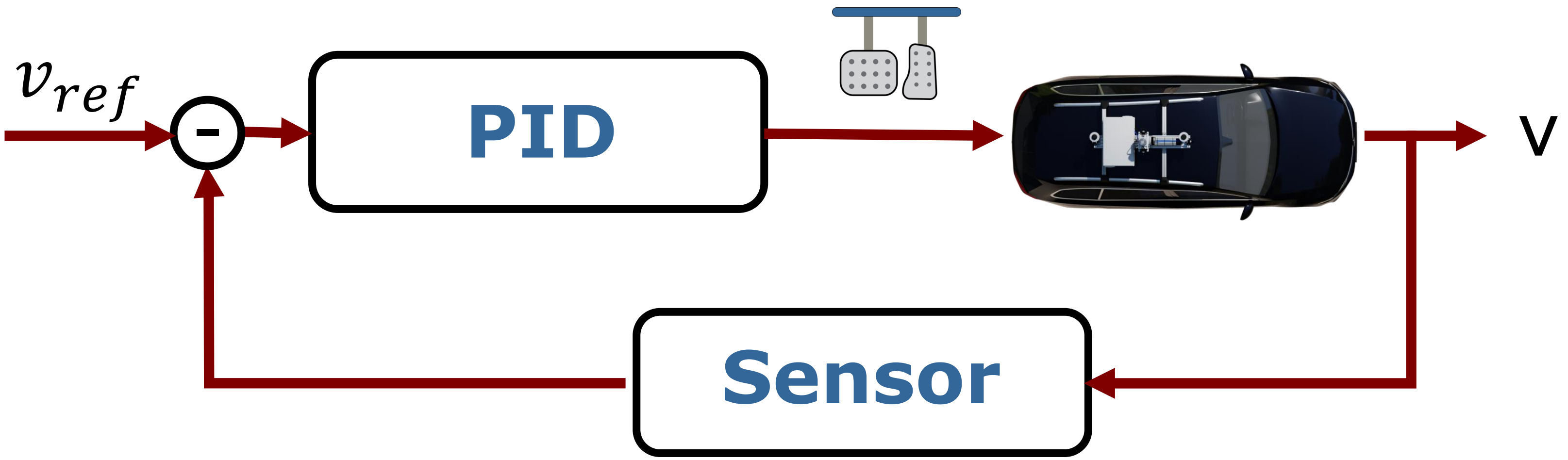
$$\ddot{x}_{des} = K_P(\dot{x}_{ref} - \dot{x}) + K_D \frac{d(\dot{x}_{ref} - \dot{x})}{dt} + K_I \int_0^t (\dot{x}_{ref} - \dot{x}) dt$$



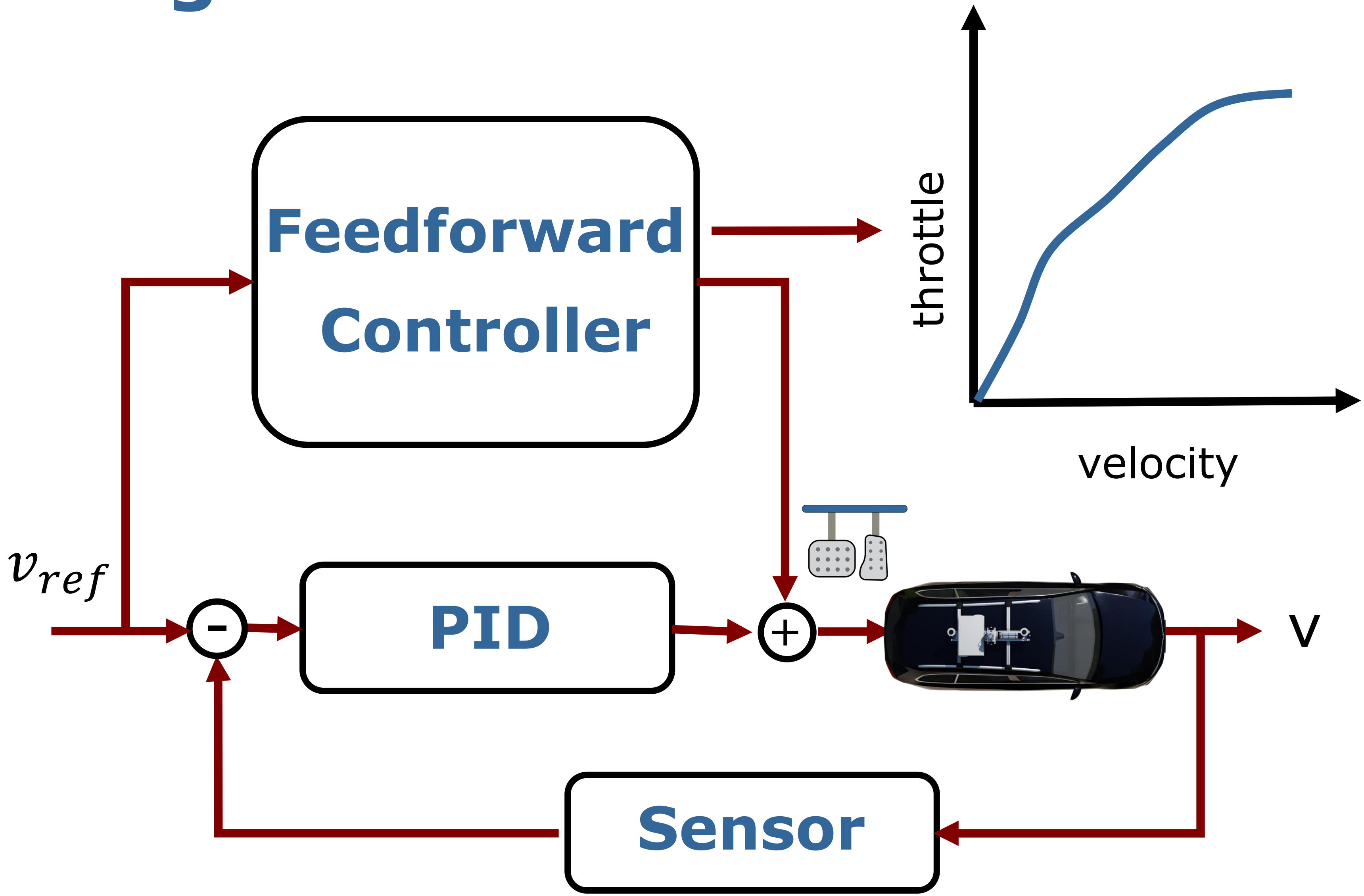
Longitudinal PID Controller





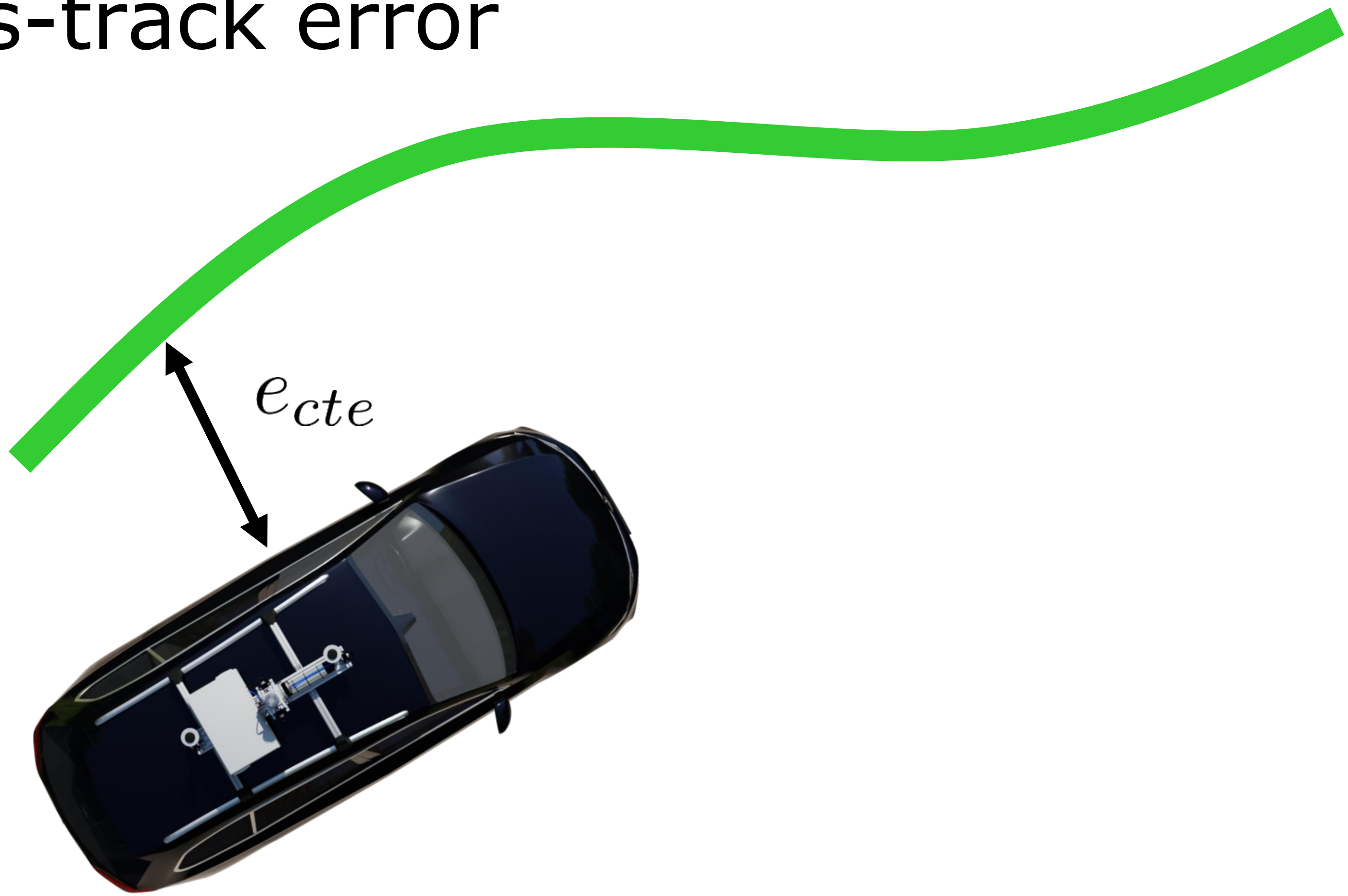


Longitudinal PID Controller



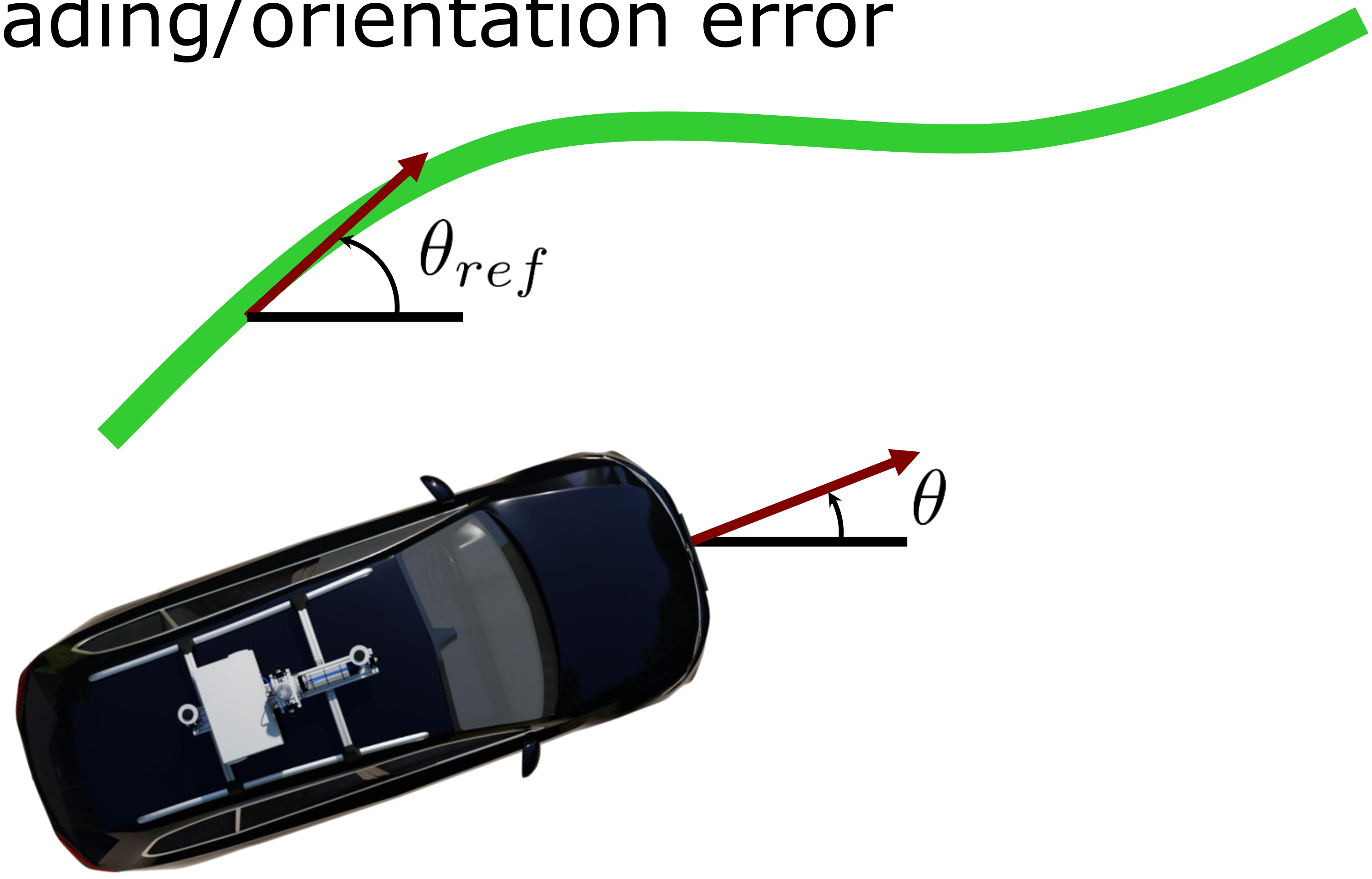
Lateral Control

- Cross-track error



Lateral Control

- Heading/orientation error

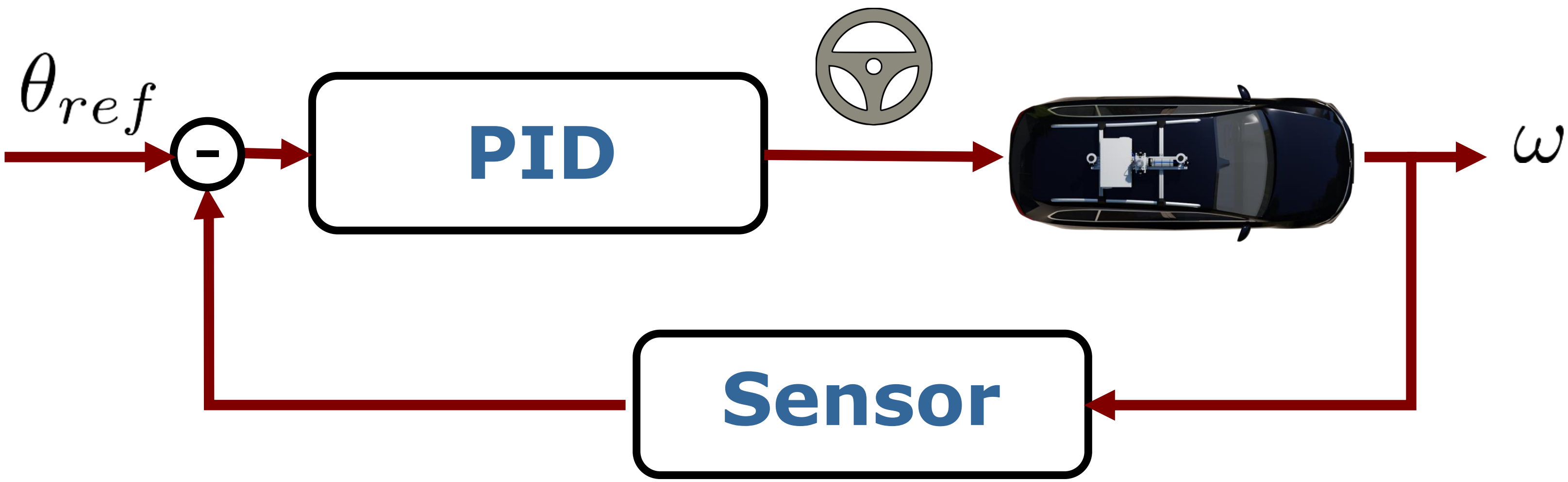


Lateral PID Controller

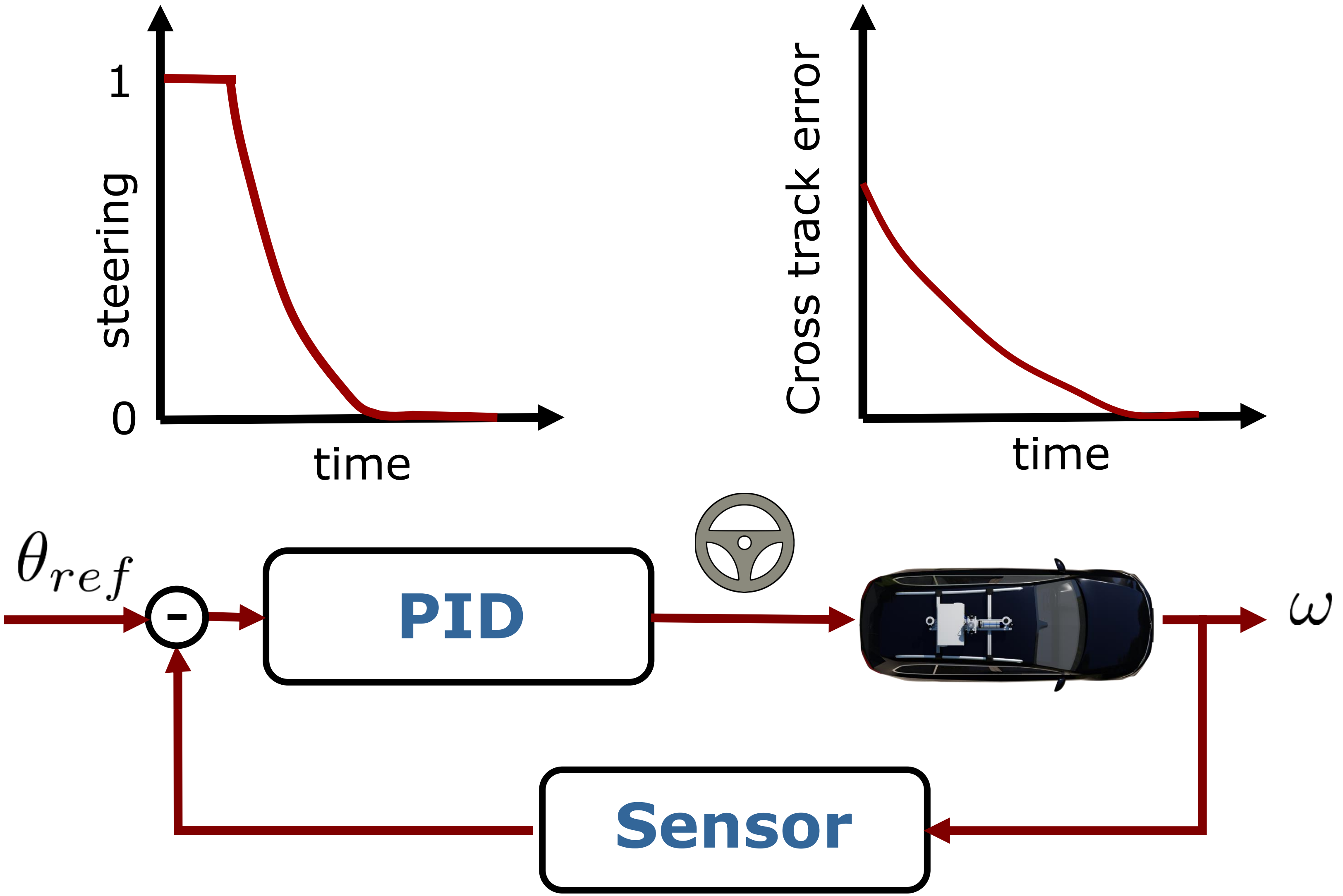
Desired steering rate

Cross-track error

$$\dot{\delta}_{des} = -K_P\, e_{cte} - K_D \frac{de_{cte}}{dt} - K_I \int_0^t e_{cte} dt$$

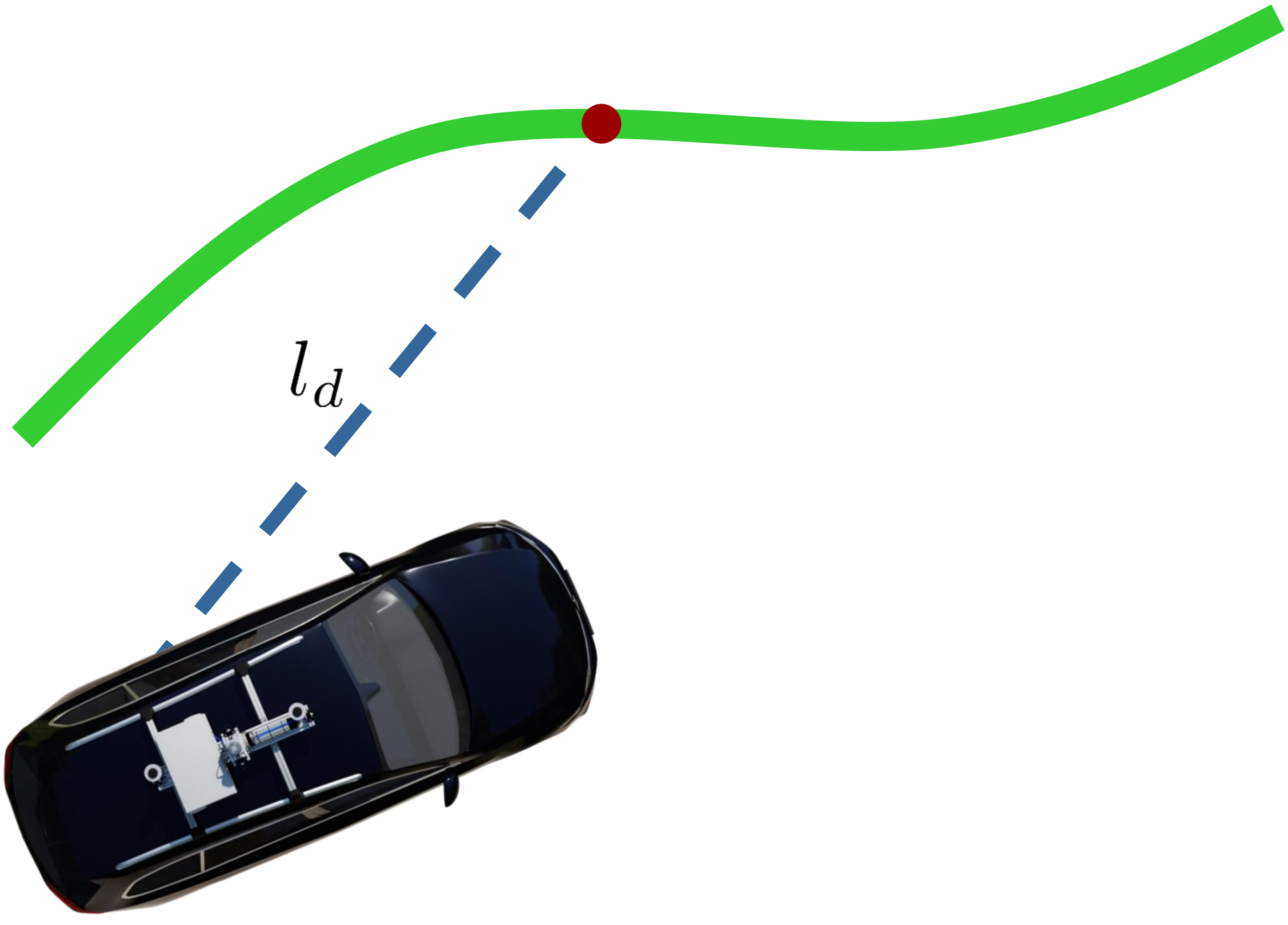


Lateral PID Controller

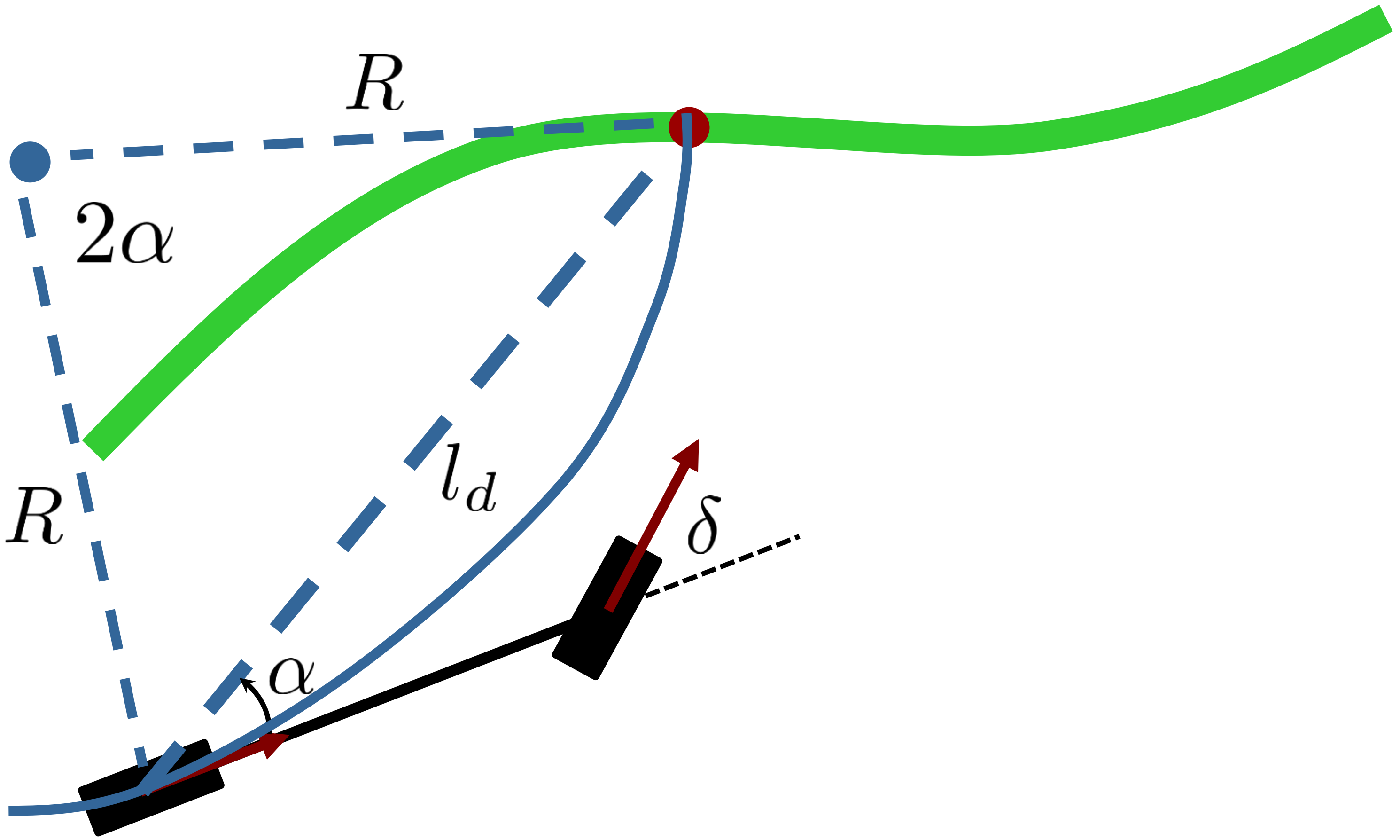


Geometric Steering Control

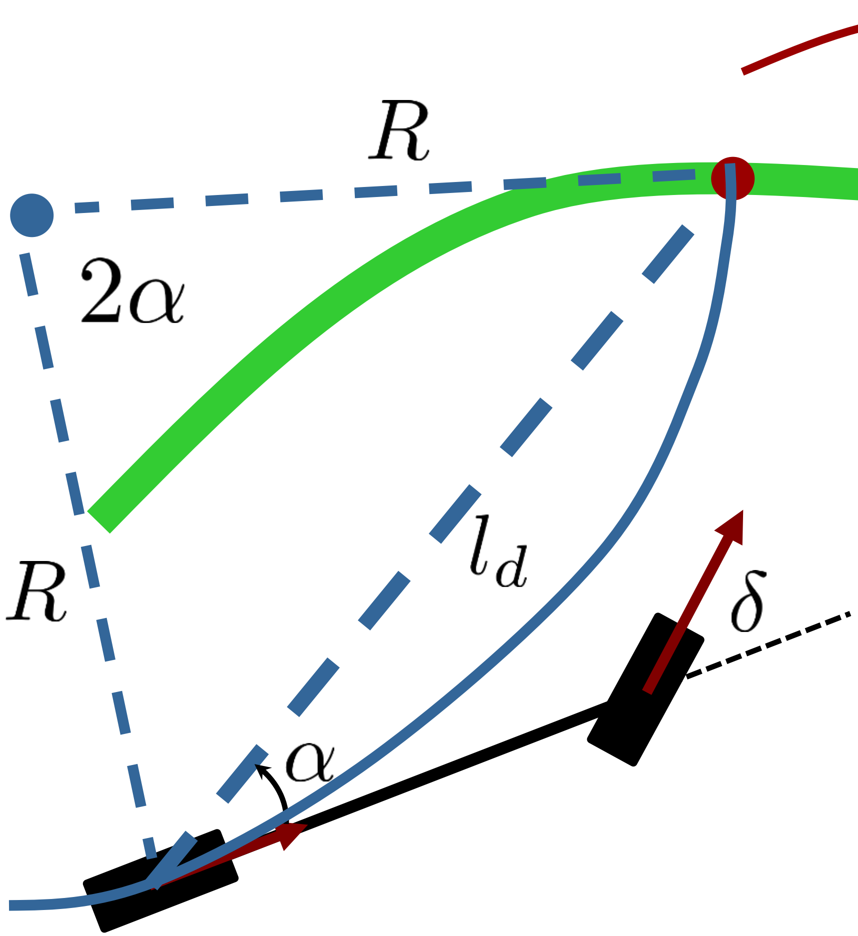
Pure Pursuit Controller



Pure Pursuit Controller



Pure Pursuit Controller



Law of sines

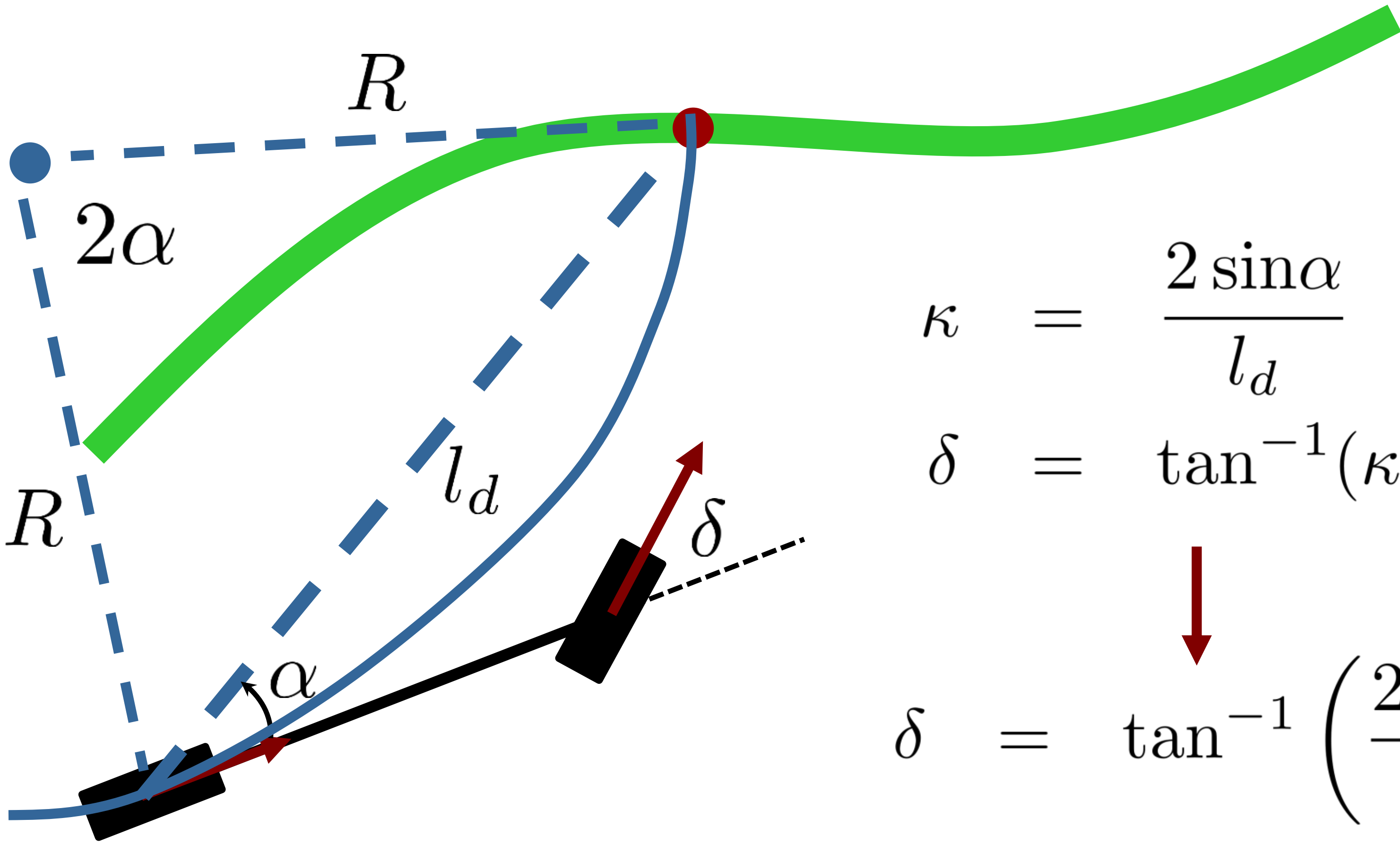
$$\frac{l_d}{\sin 2\alpha} = \frac{R}{\sin\frac{\pi}{2} - \alpha}$$

$$\frac{l_d}{2\sin\alpha\cos\alpha} = \frac{R}{\cos\alpha}$$

$$\frac{l_d}{\sin\alpha} = 2R$$

$$\kappa = \frac{1}{R} = \frac{2\sin\alpha}{l_d}$$

Pure Pursuit Controller



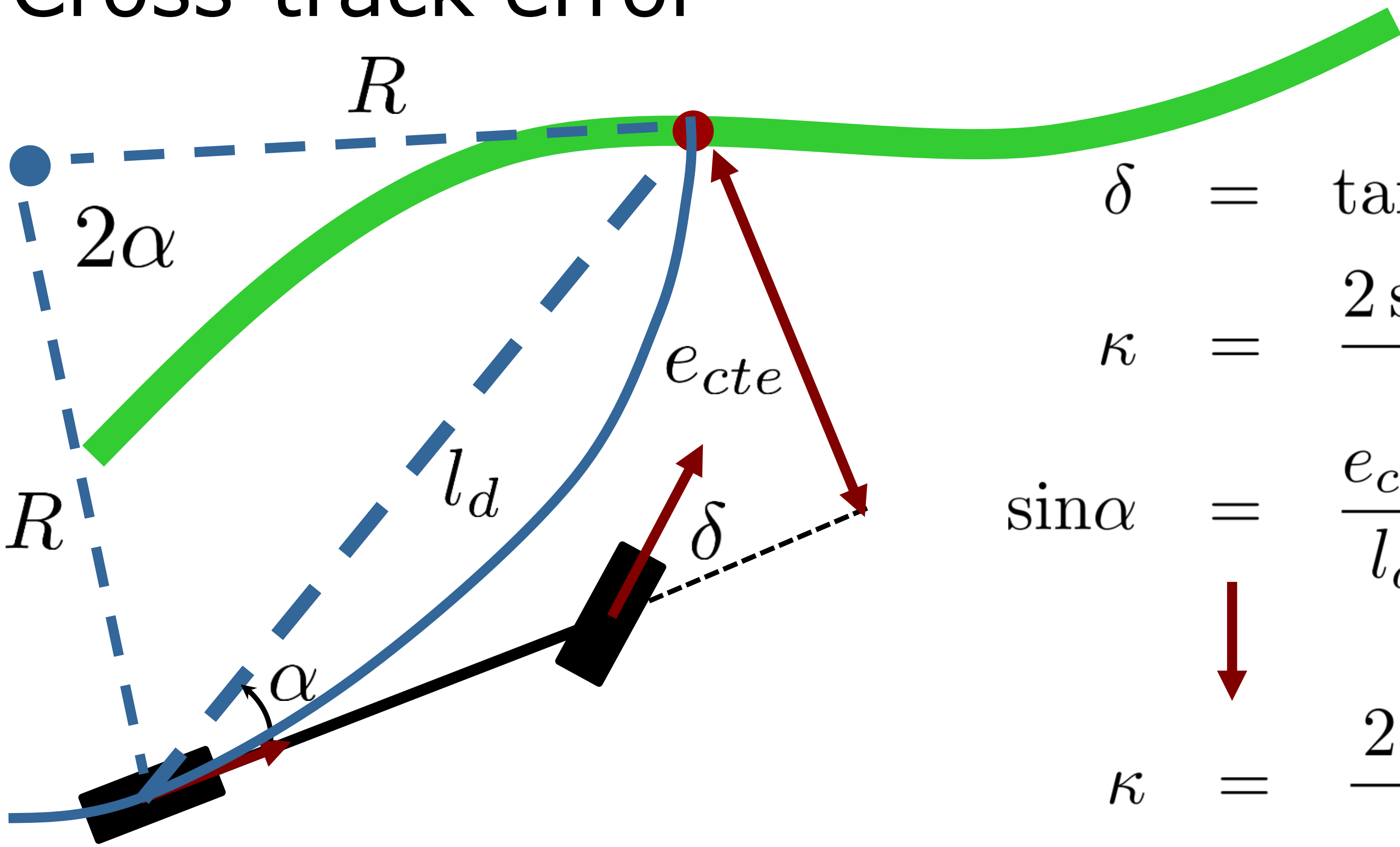
$$\kappa = \frac{2\sin\alpha}{l_d}$$

$$\delta = \tan^{-1}(\kappa L)$$

$$\delta = \tan^{-1}\left(\frac{2L\sin\alpha}{l_d}\right)$$

Pure Pursuit Controller

- Cross-track error



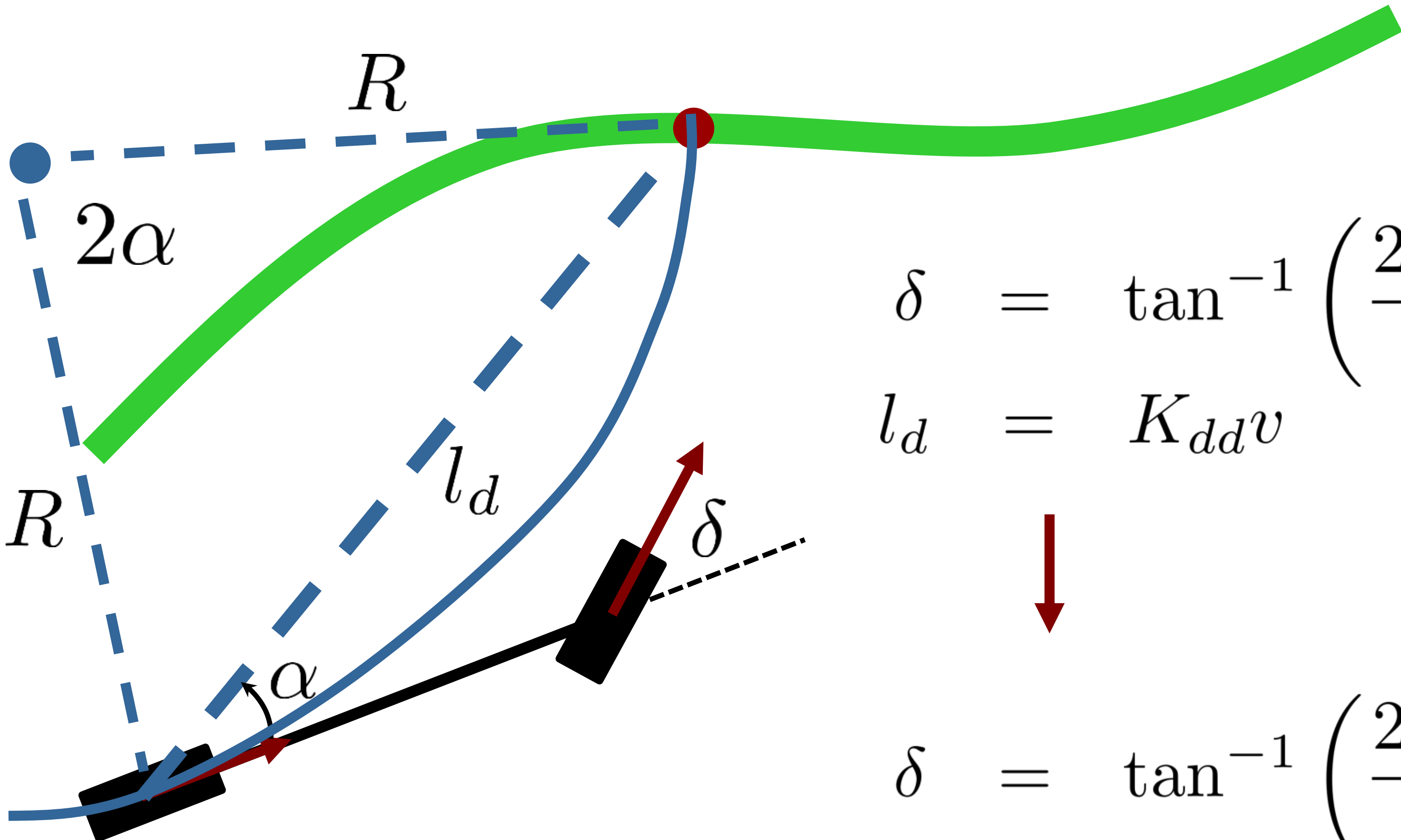
$$\delta = \tan^{-1}(\kappa L)$$

$$\kappa = \frac{2\sin\alpha}{l_d}$$

$$\sin\alpha = \frac{e_{cte}}{l_d}$$

$$\kappa = \frac{2e_{cte}}{l_d^2}$$

Pure Pursuit Controller



$$\delta = \tan^{-1}\left(\frac{2L\sin\alpha}{l_d}\right)$$

$$l_d = K_{dd}v$$

$$\downarrow$$

$$\delta = \tan^{-1}\left(\frac{2L\sin\alpha}{K_{dd}\,v}\right)$$

Stanley Controller

- Used successfully in the DARPA Grand Challenge

[Courtesy of S. Thrun]

Stanley Controller

- Reduce both the error in heading and the nearest point on the reference trajectory
- Align Heading:

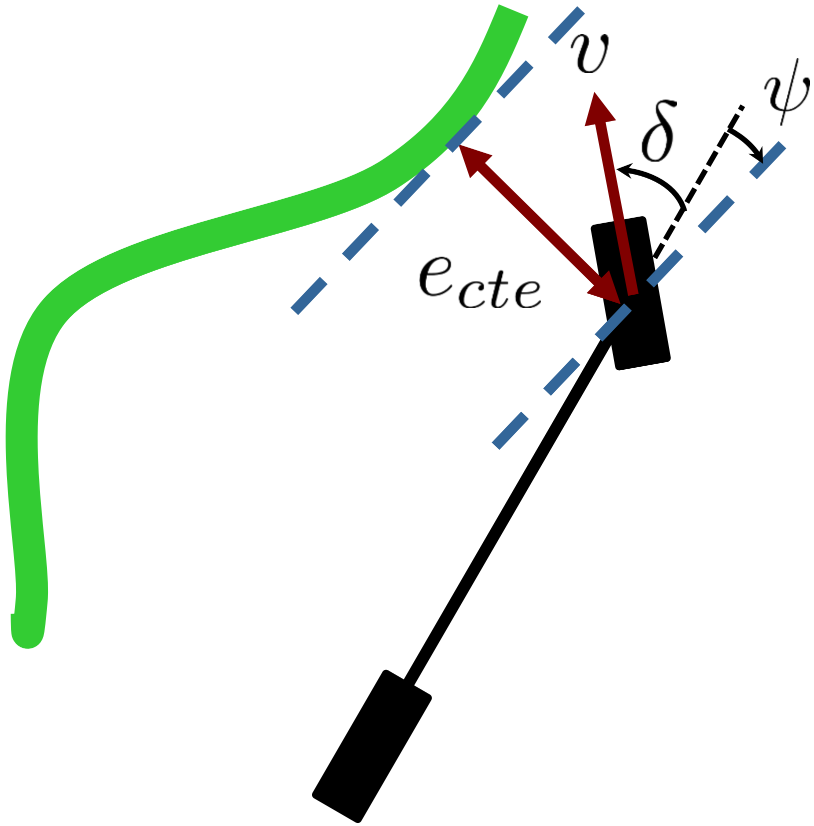
$$\delta = \psi$$

- Cross-track error:

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

- Steering limit:

$$\delta \in [\delta_{min}, \delta_{max}]$$

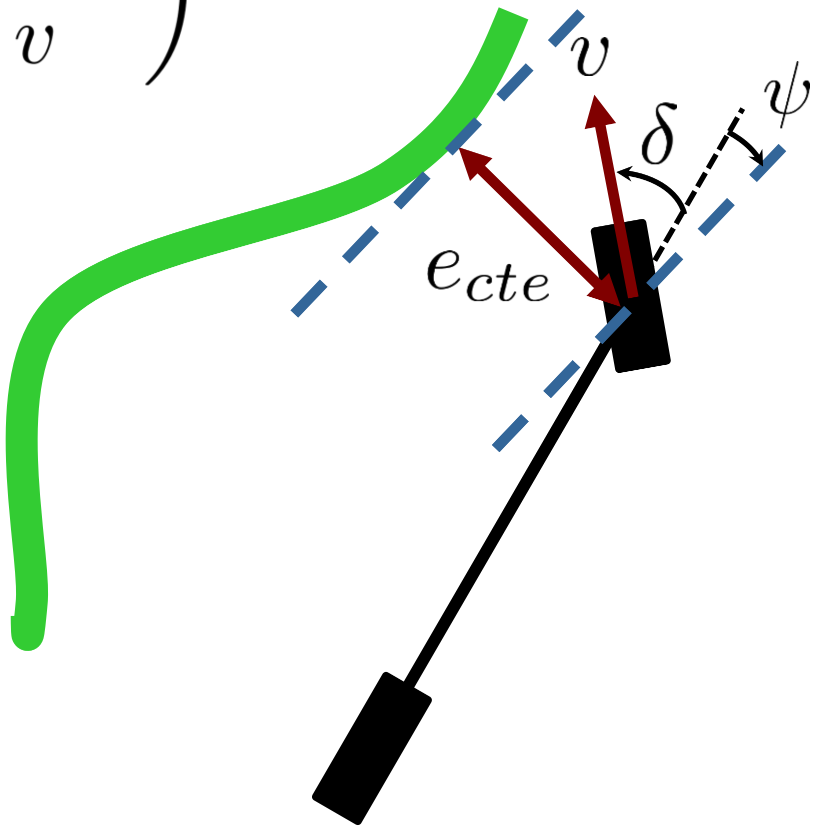


Stanley Controller

- Combined control law:

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

$$\delta \in [\delta_{min}, \delta_{max}]$$



Pros and Cons of Reactive Control

- **Pros**
  - Simple control rules
  - Highly efficient to compute
- **Cons**
  - Cannot account for external constraints
  - Gains must be hand-tuned
  - Separation into longitudinal and lateral controllers ignores coupling
  - Ignores future decisions

Advanced Control Paradigms

- Model Predictive Control (MPC)
- Learning-based Approaches
  - Reinforcement Learning
  - Imitation Learning

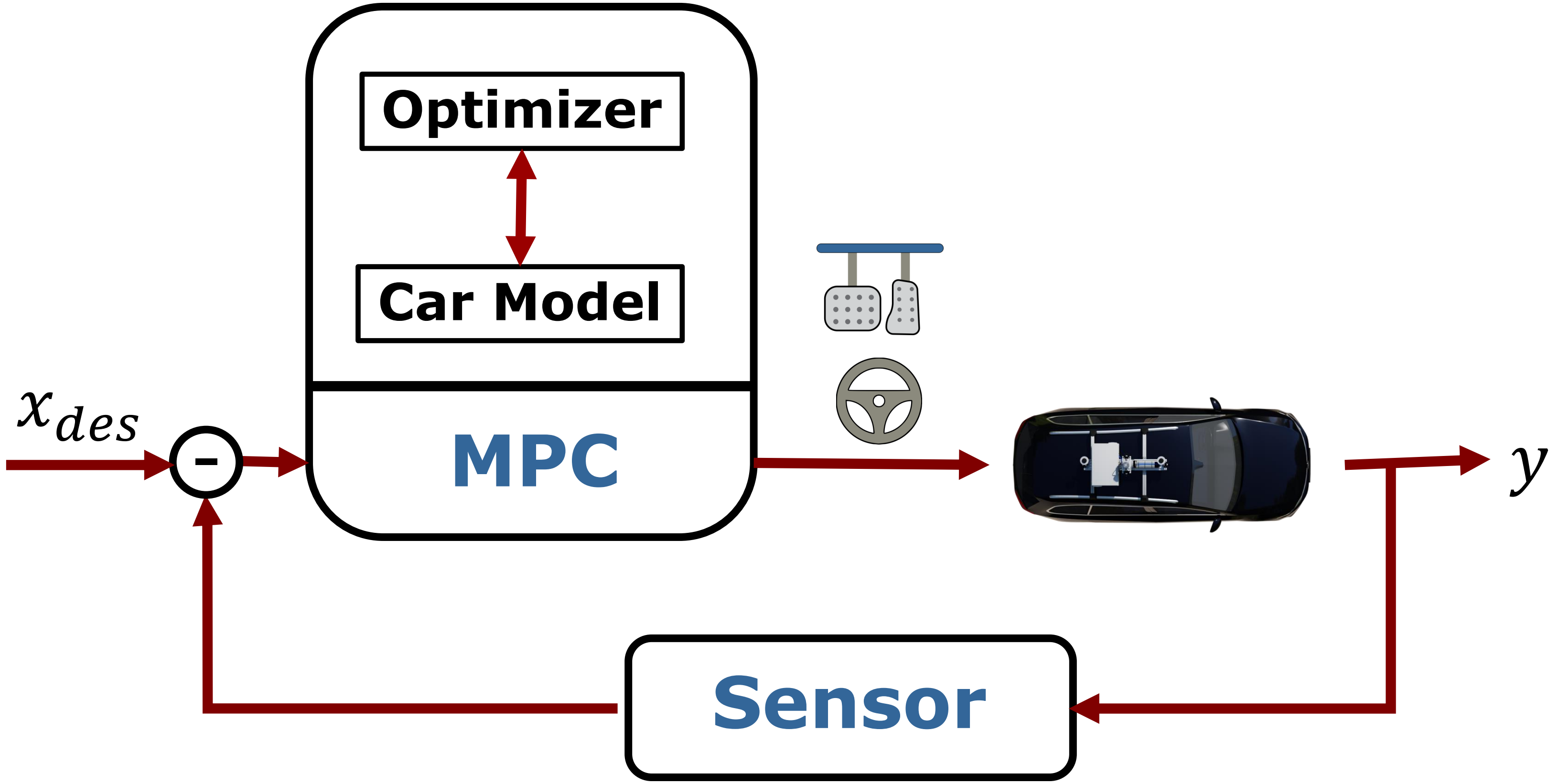
Model Predictive Control (MPC)

Control as Optimization

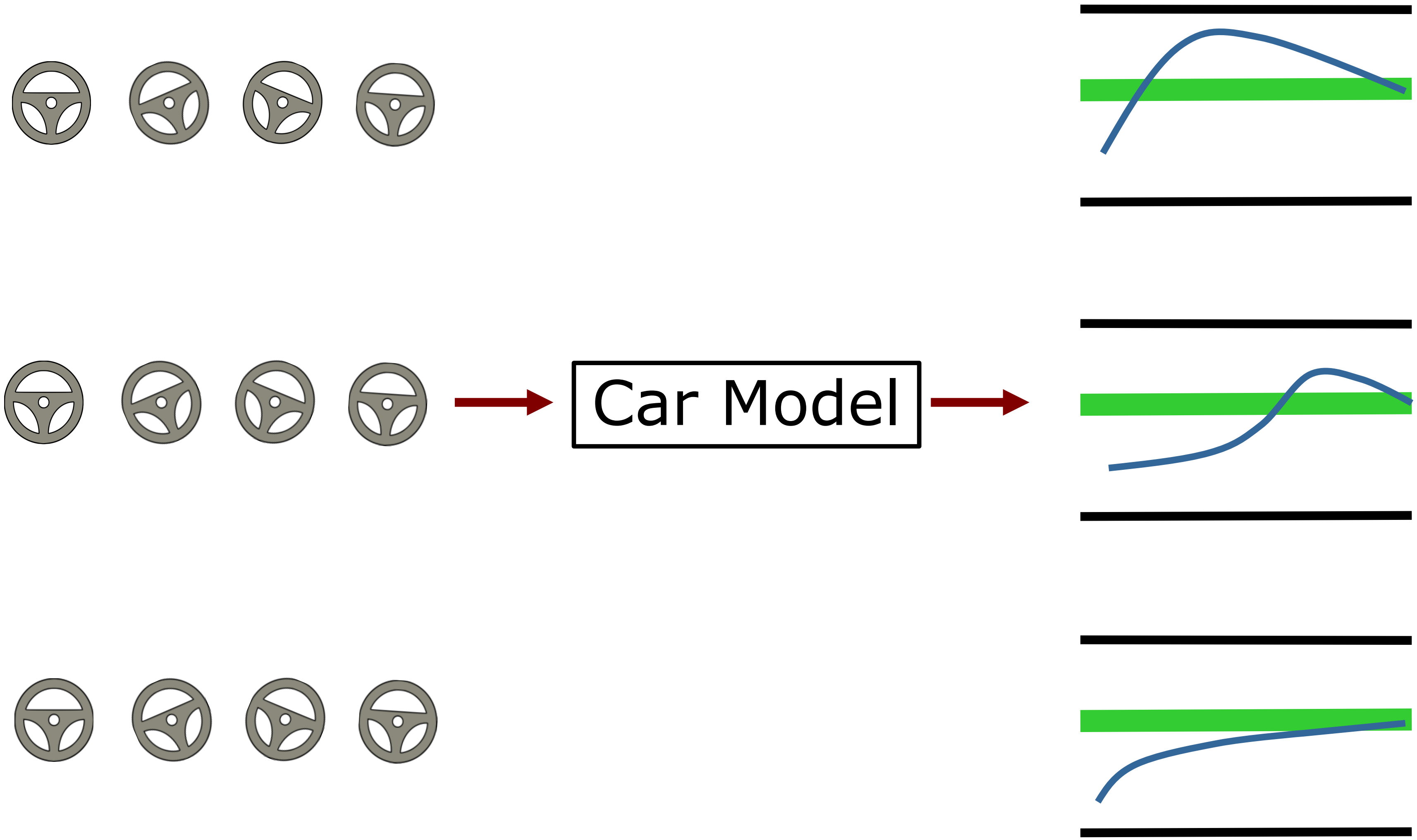
- As before:
  - Plan trajectory → Follow trajectory
- Use **optimization** to find control commands using a simulation
- MPC uses predicted vehicle states to find optimal controls

- More technical details in next lectures

MPC for Self-Driving Cars

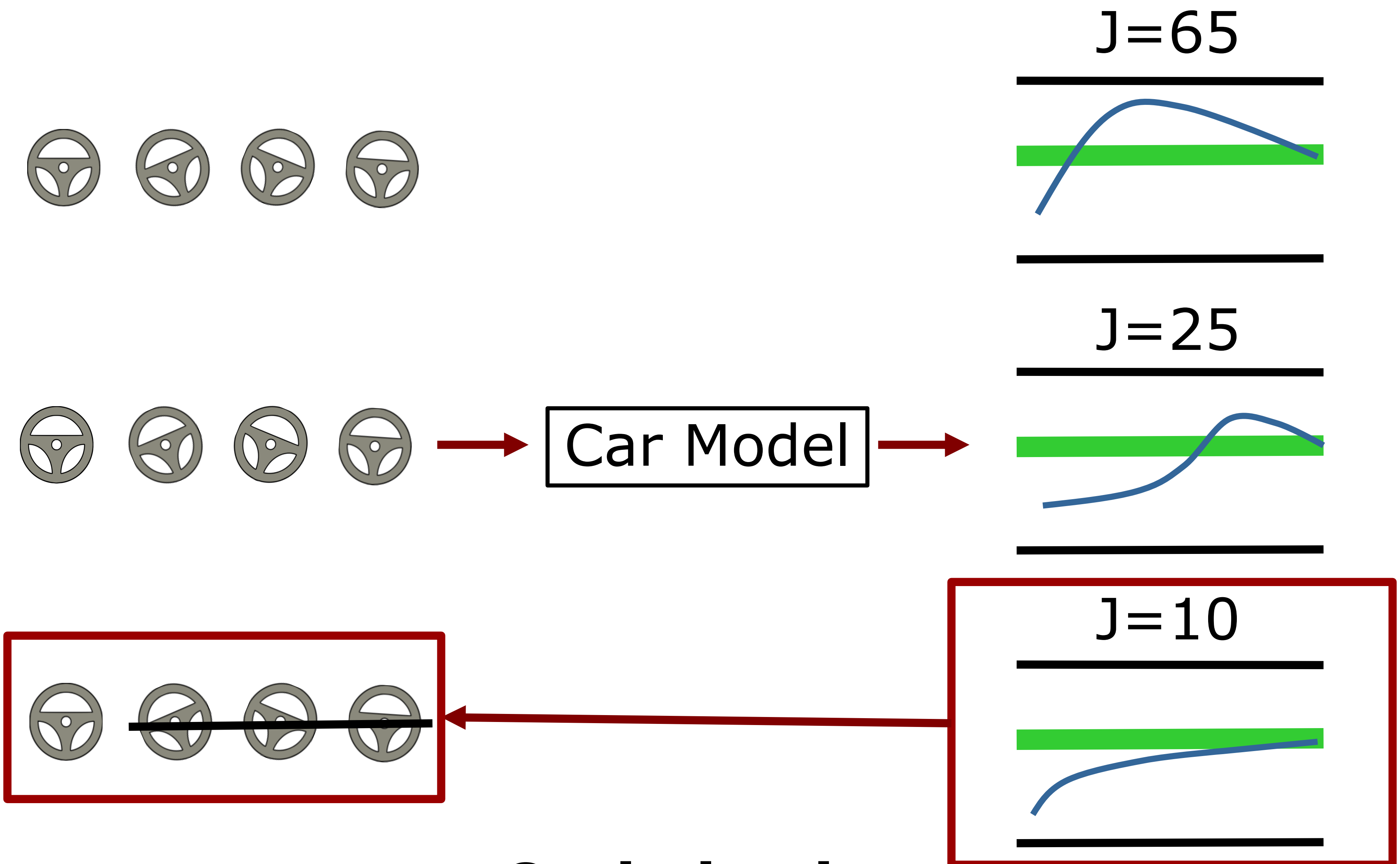


MPC for Self-Driving Cars



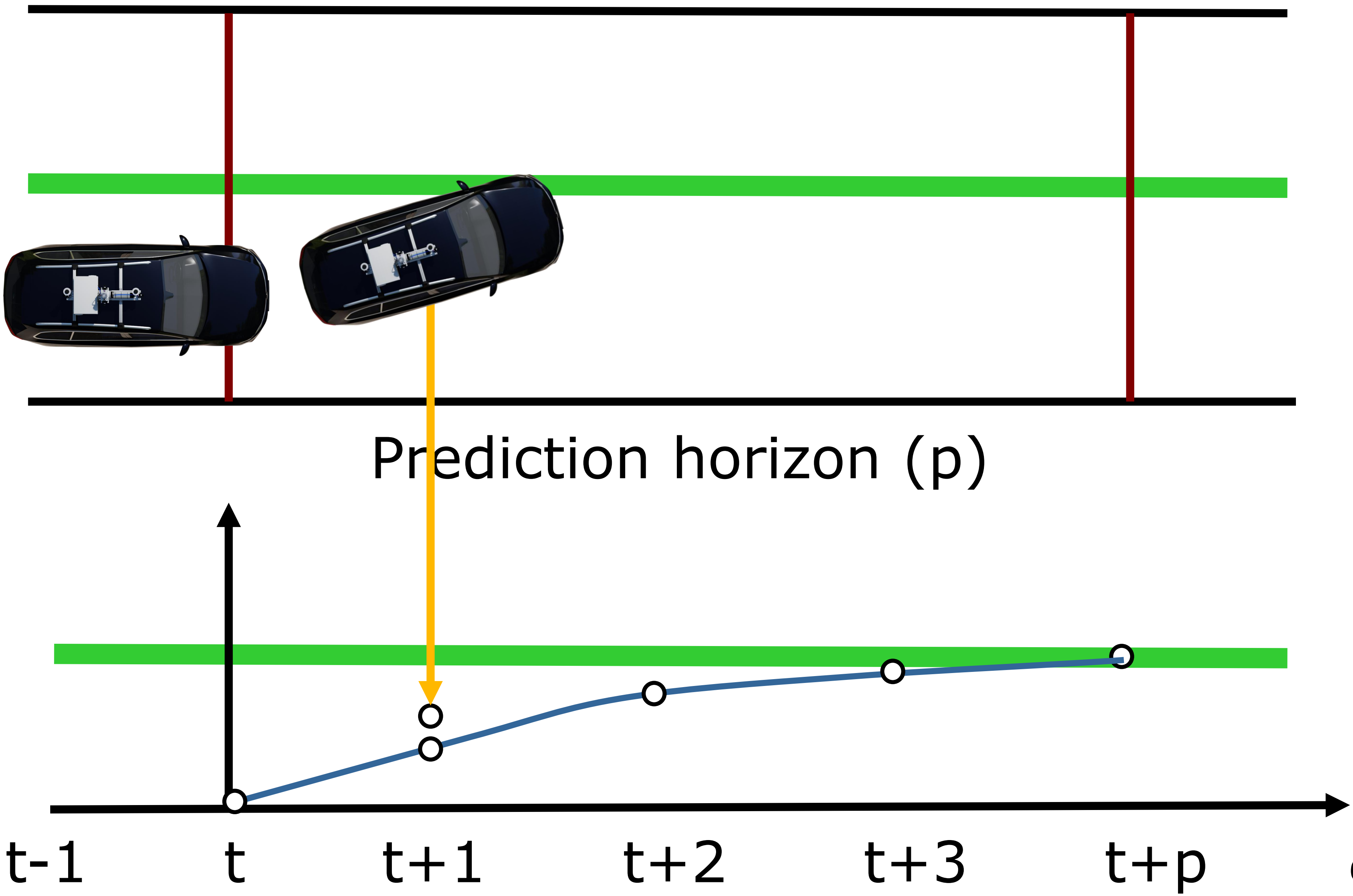
Simulation/Prediction

MPC for Self-Driving Cars

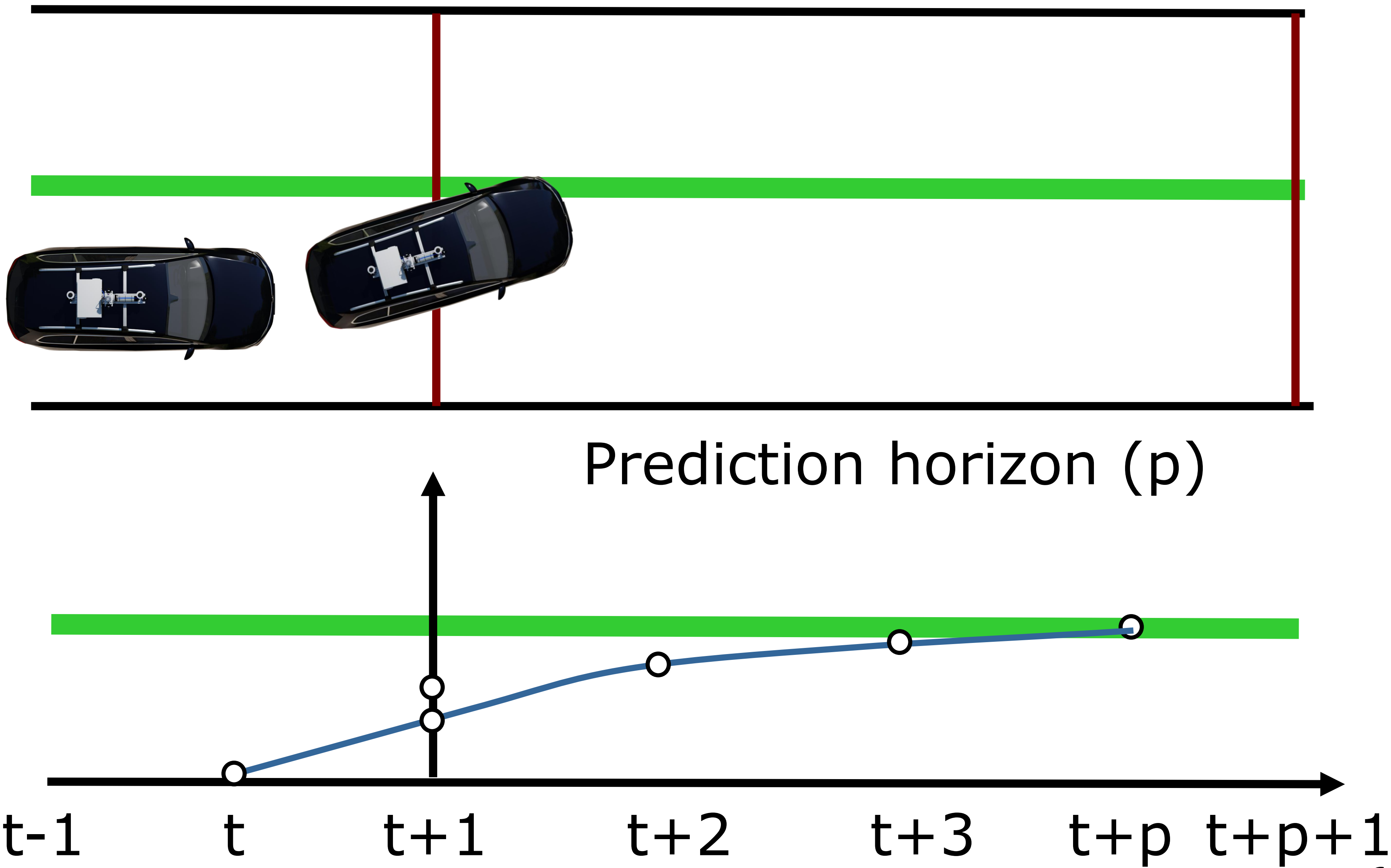


Optimization

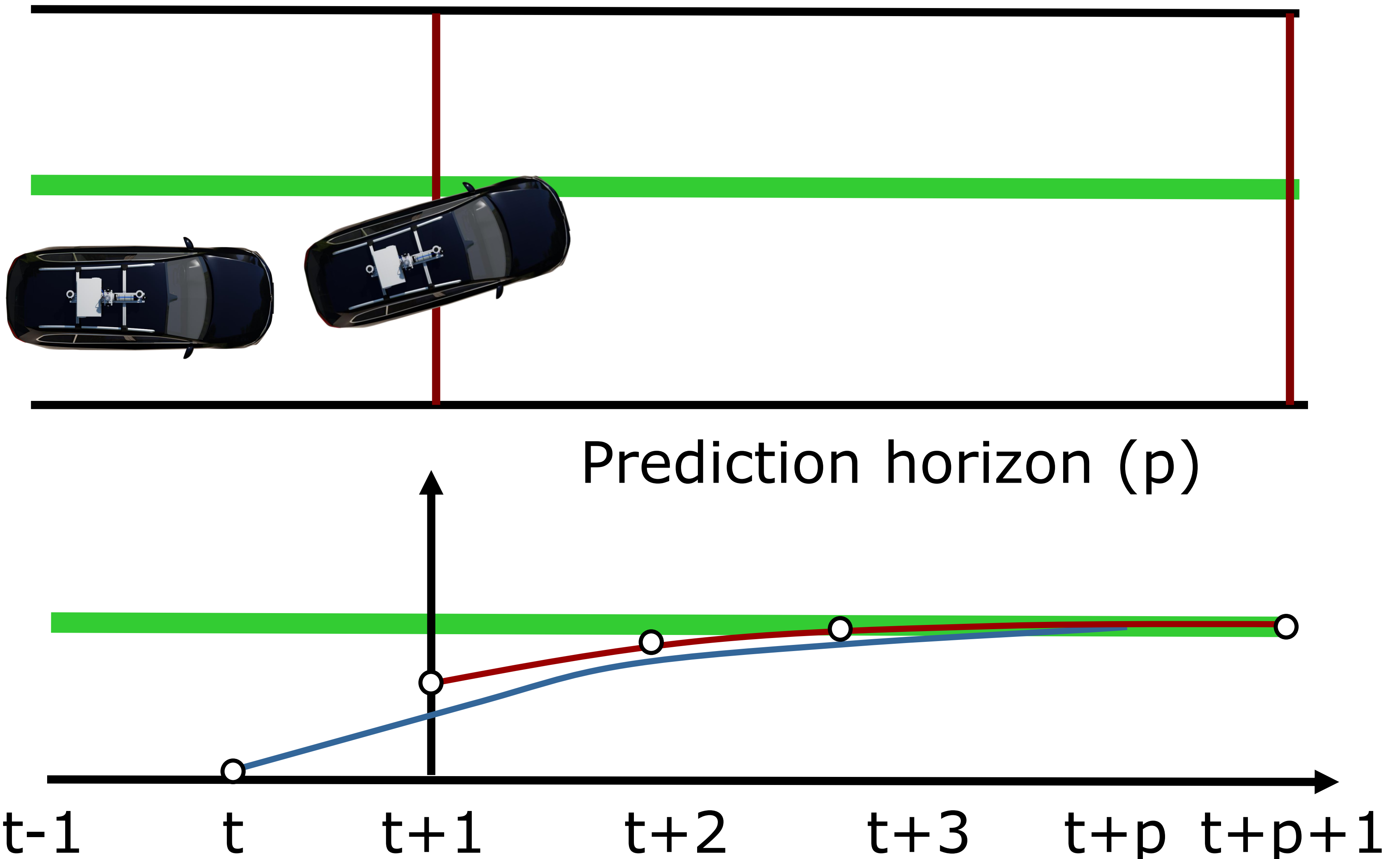
MPC for Self-Driving Cars



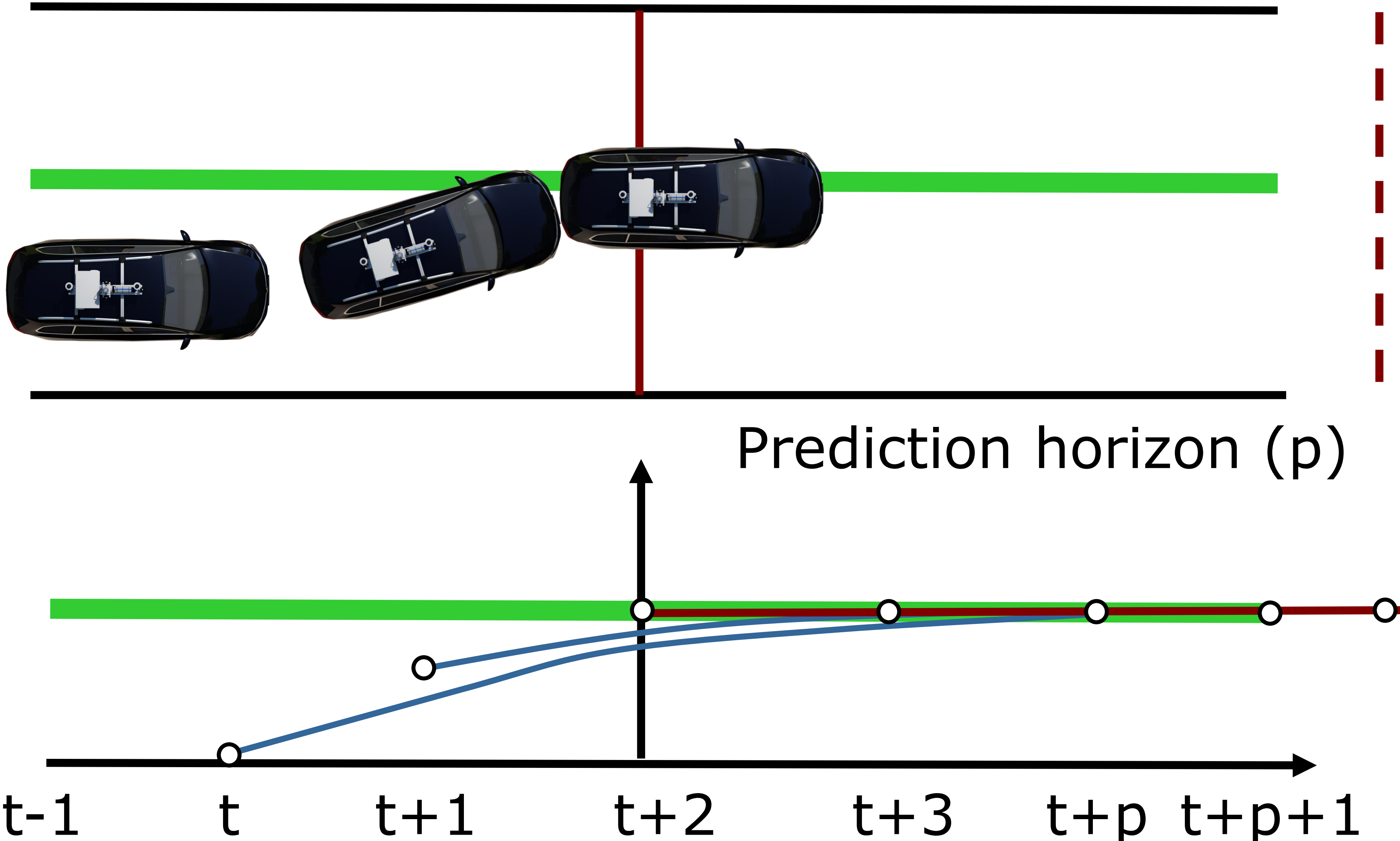
MPC for Self-Driving Cars



MPC for Self-Driving Cars



MPC for Self-Driving Cars



Pros and Cons of MPC

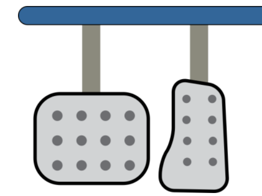
- **Pros**
  - Preview accounts for future decisions
  - Systematic procedure to derive controllers even for complex systems

- **Cons**
  - Higher computational and memory requirements than reactive controllers
  - Still dependent on planned trajectory

Reinforcement Learning

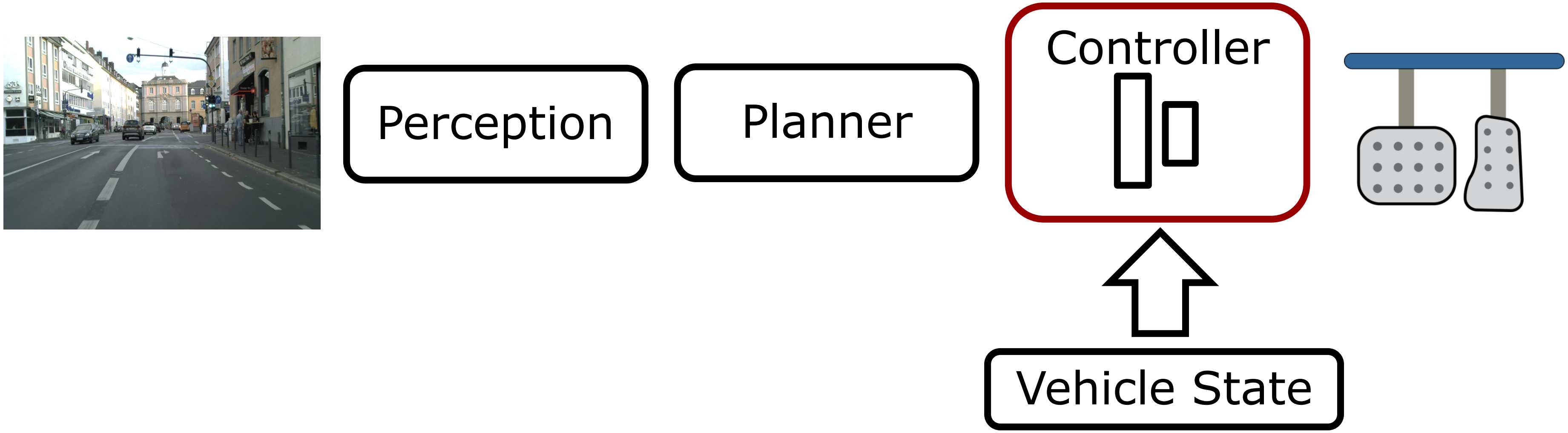
Modular Approach

Perception → Planner → Controller



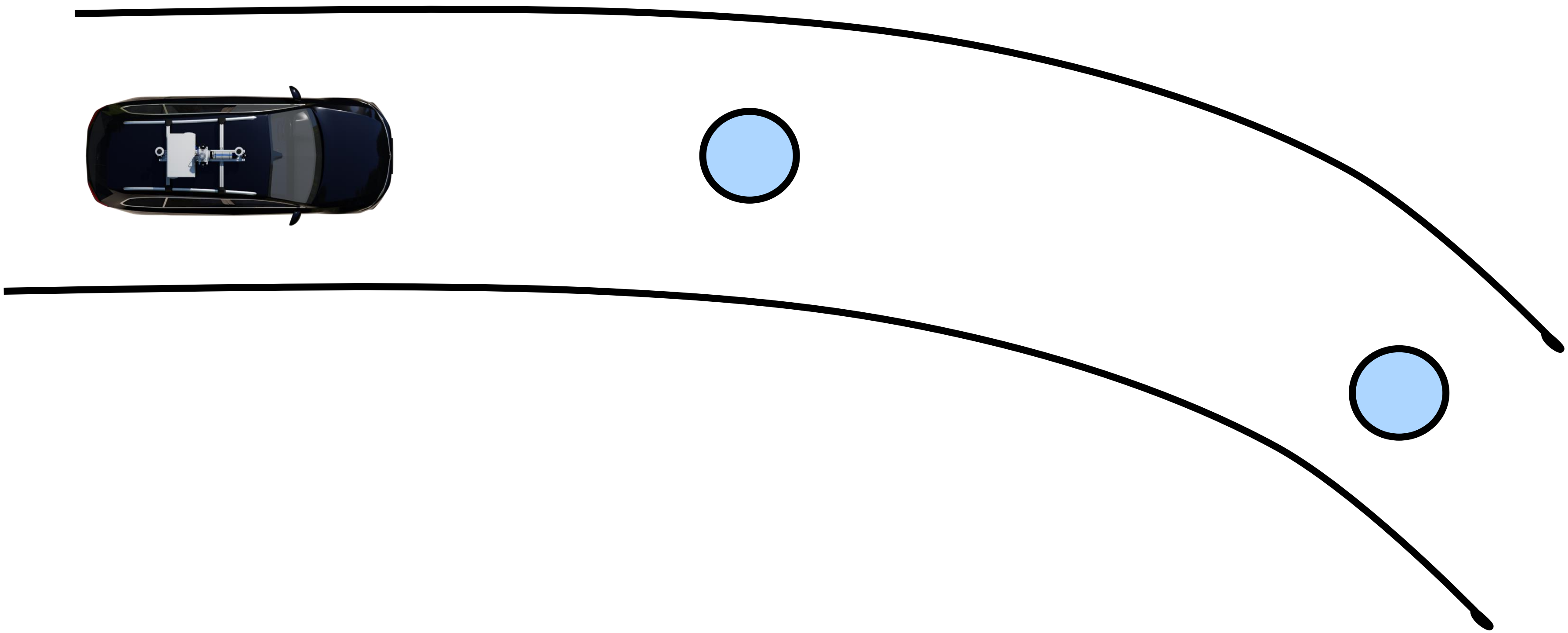
- Modular approach decomposes driving into components
- Decomposition allows separate development

Learning a Control Policy



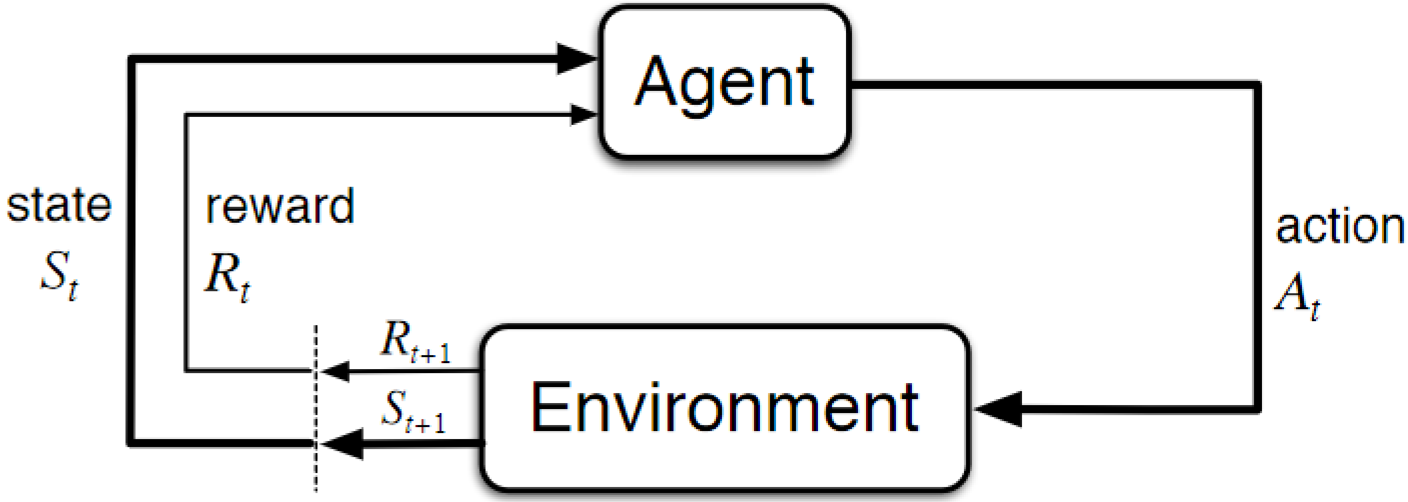
- Learning a controller has potential to have improved controls
- Less assumptions & dependence on trajectories

Example: High-level planning → Vehicle Controls



- Provide only waypoints by planner
- Learn **policy** to reach waypoints

RL in a Nutshell



- Agent interacts with environment via action A_t based on state S_t
- Agent receives reward R_{t+1} and observes next state S_{t+1}

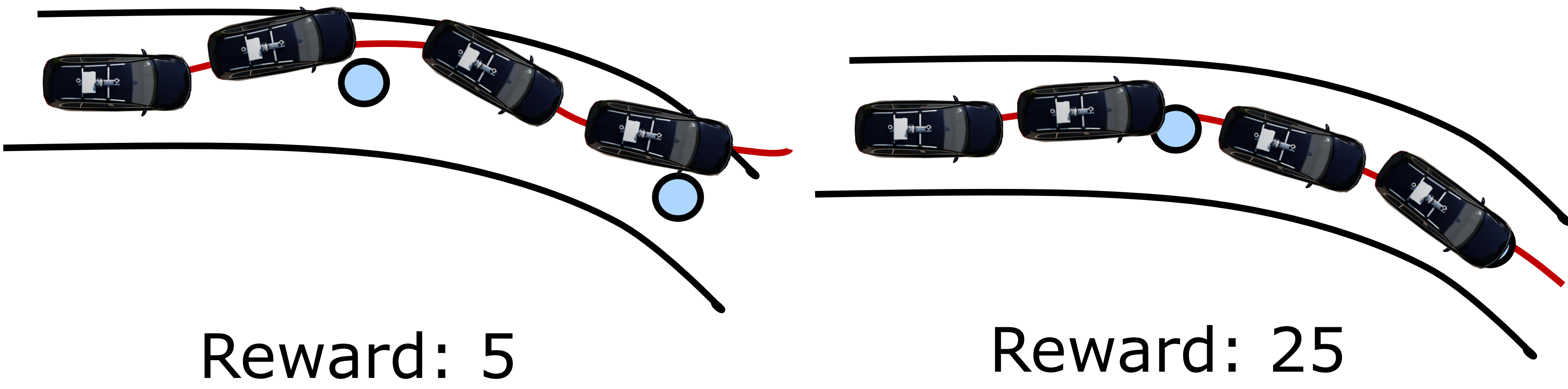
[Image from Sutton & Barto]

Task: Learn Policy

- **Goal:** Learn **policy** $\pi(S_t) = A_t$ that maximizes reward

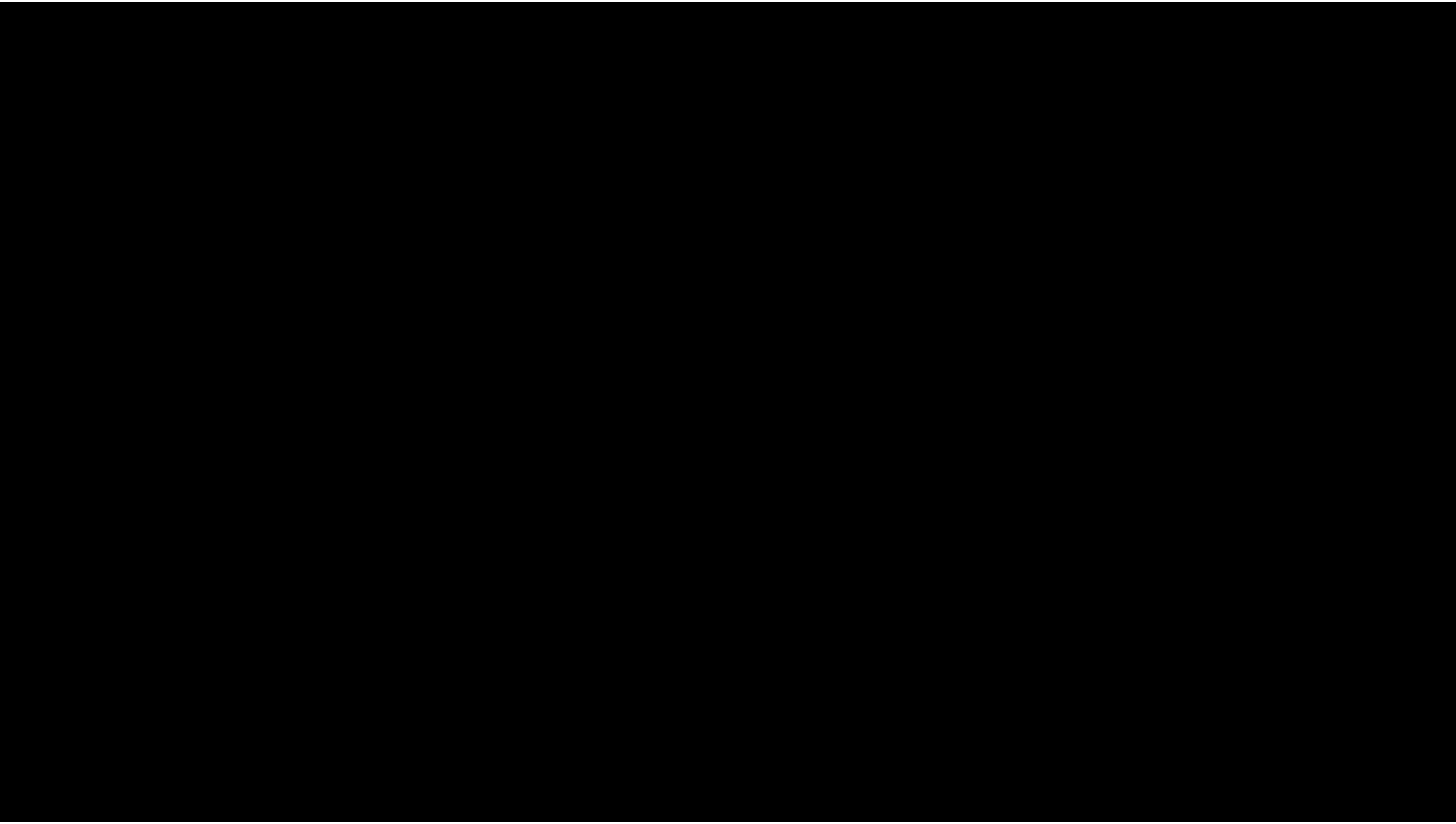
- Examples
  - Positive reward (+10) for reaching goal location, negative reward (-1) for every action
  - Positive reward (+1) for staying in lane, negative reward (-10) for leaving lane

Simulation for Policy Learning



- Learning policy by **trial-and-error**
- Agent improves policy over time and discovers “good” actions

Example: Control of a Drone



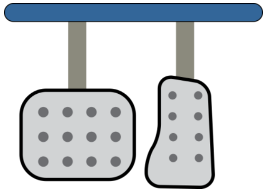
Pros and Cons of Reinforcement Learning

- **Pros**
  - Learned controller can lead to superior performance
  - Non-linear/non-continuous reward functions
- **Cons**
  - Interpretability & Explainability
  - Training needs simulations
  - Generalization to unseen conditions

Imitation Learning

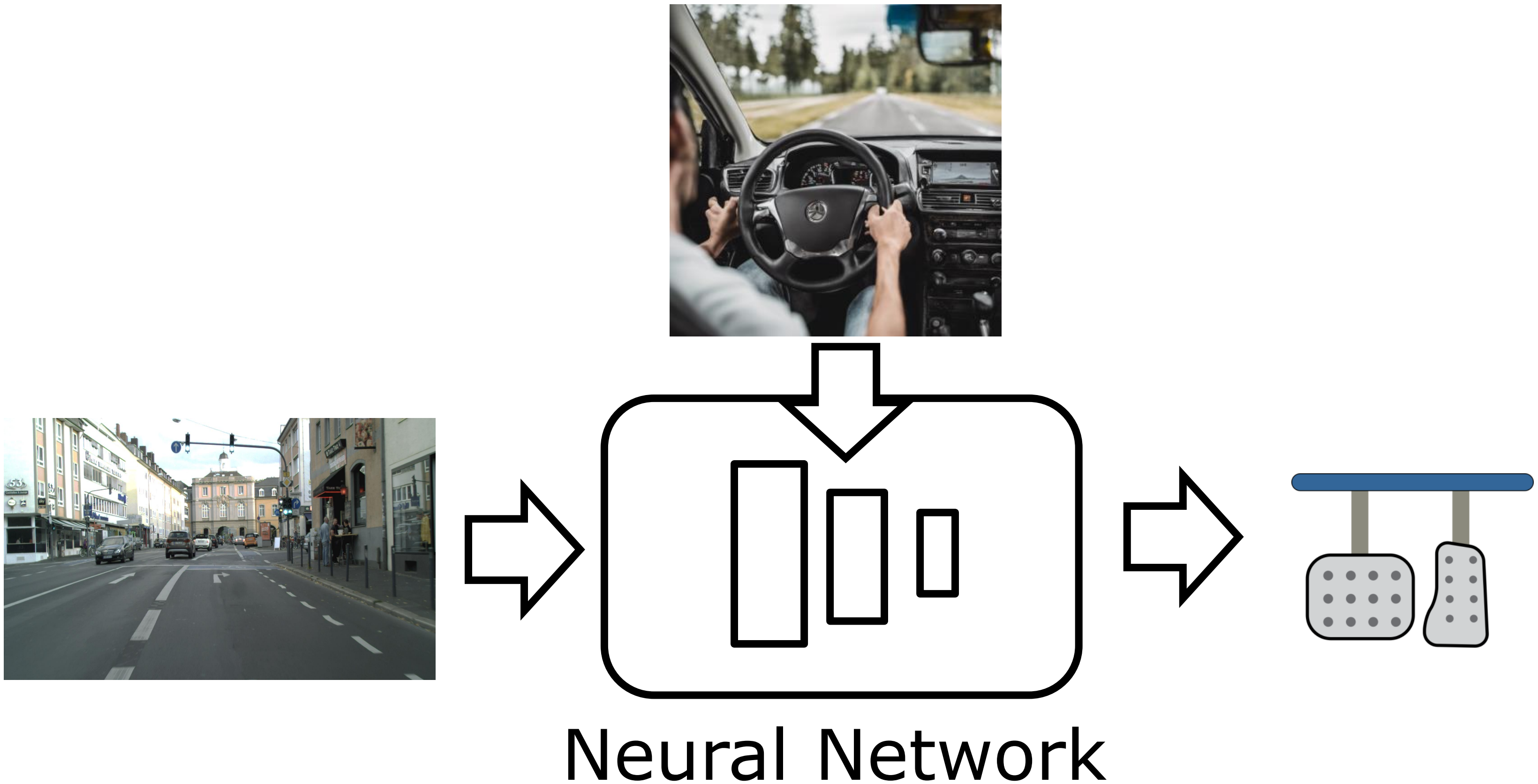
End-to-end Driving

Neural Network



- Replace modules with single neural network that maps directly to control

Learn from Demonstrations



- **Input:** demonstrations from an expert
- **Goal:** train policy to mimic decisions

Pros and Cons of End-to-End Autonomous Driving

- **Pros**
  - Direct usage of relevant sensor information
  - Demonstrations simple to generate, no annotations
- **Cons**
  - Interpretability & Validation harder
  - Generalization to unseen conditions or extreme situations

Summary

- Kinematic modeling for a car
- Idea of feedback control
- Trajectory control using **PID control**
- Lateral control strategy based on **geometry**
- Dynamic control strategy using **Model Predictive Control (MPC)**
- Learning-based **approaches relax certain** assumptions

Resources

- “Robotics, Control and Vision” by Dr. Peter Corke
- “Introduction to Self-driving Cars” by Steven Waslander
- “Visual navigation for flying robots” by Dr. Jürgen Strum

Link: https://www.edx.org/course/autonomous-navigation-flying-robots-tumx-autonavx-0

- “Control for Mobile Robots” by Dr. Magnus Egerstedt

Link: https://www.coursera.org/learn/mobile-robot

Resources (cont.)

- "Reinforcement Learning: An Introduction" (2^{nd} Edition) by R. Sutton & A. Barto, 2020.
- "Deep Reinforcement Learning for Autonomous Driving: A Survey" by Kiran et al., Trans. on Intell. Transp. Sys., 2022.
- "Reaching the limit in autonomous racing: Optimal Control versus Reinforcement Learning" by Song et al., Science Robotics, 2023.

Thank you for your attention