SELF-DRIVING CARS CONTROL

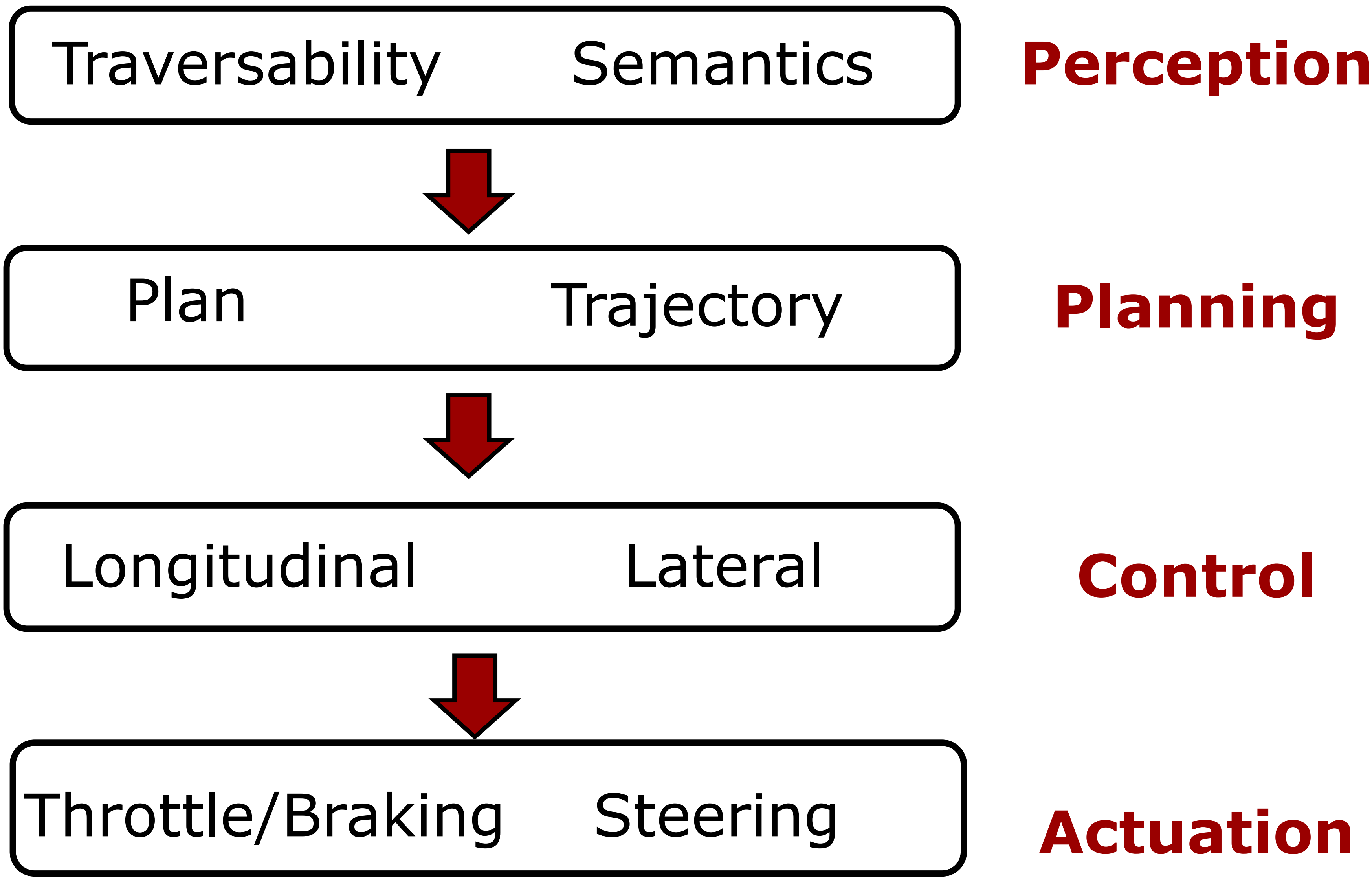
Control for Self-Driving Cars

Nived Chebrolu

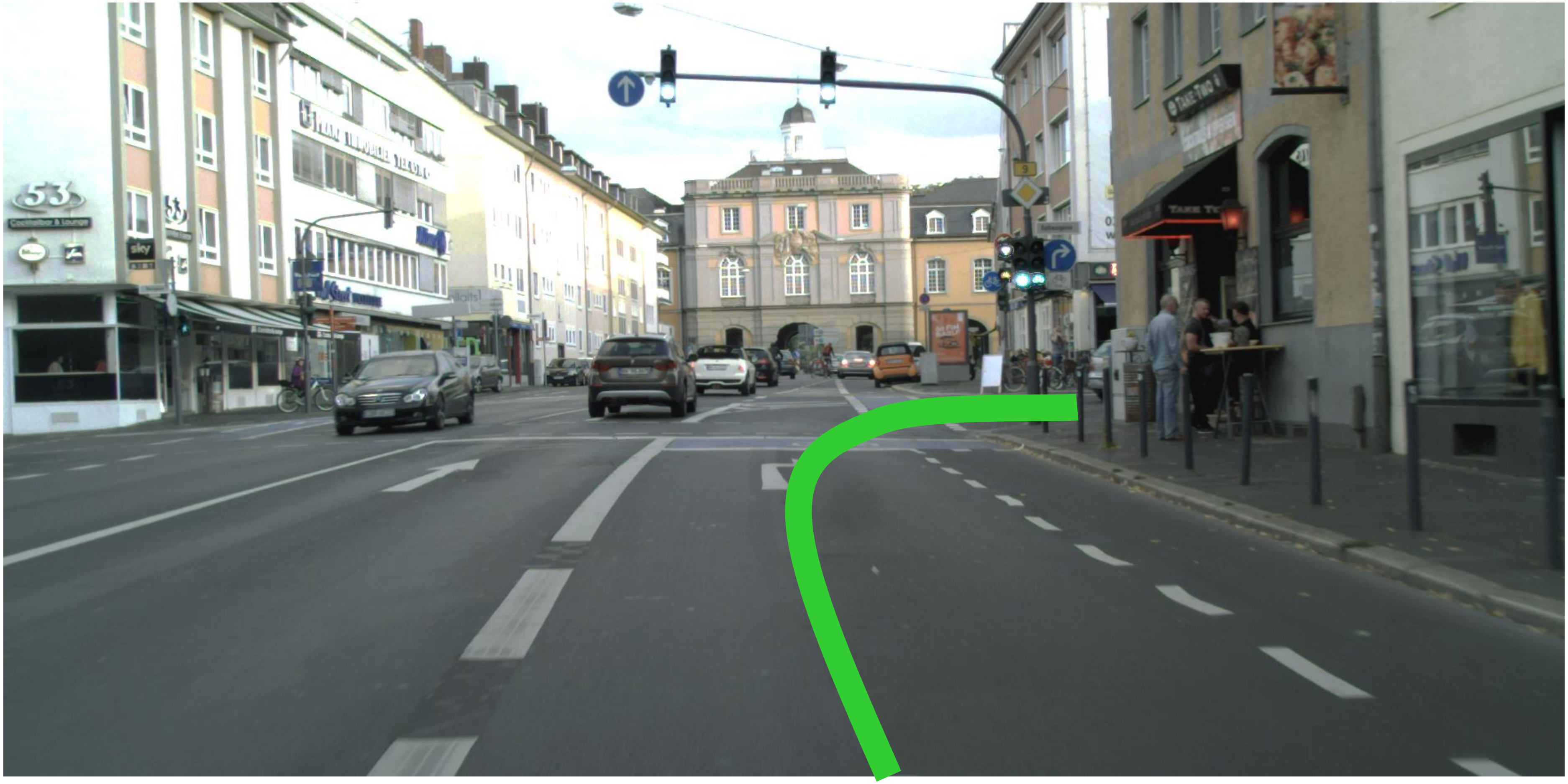
Part of the Course: Techniques for Self-Driving Cars by
C. Stachniss, J. Behley, N. Chebrolu, B. Mersch, I. Bogoslavskyi

Self-Driving Car Scenario

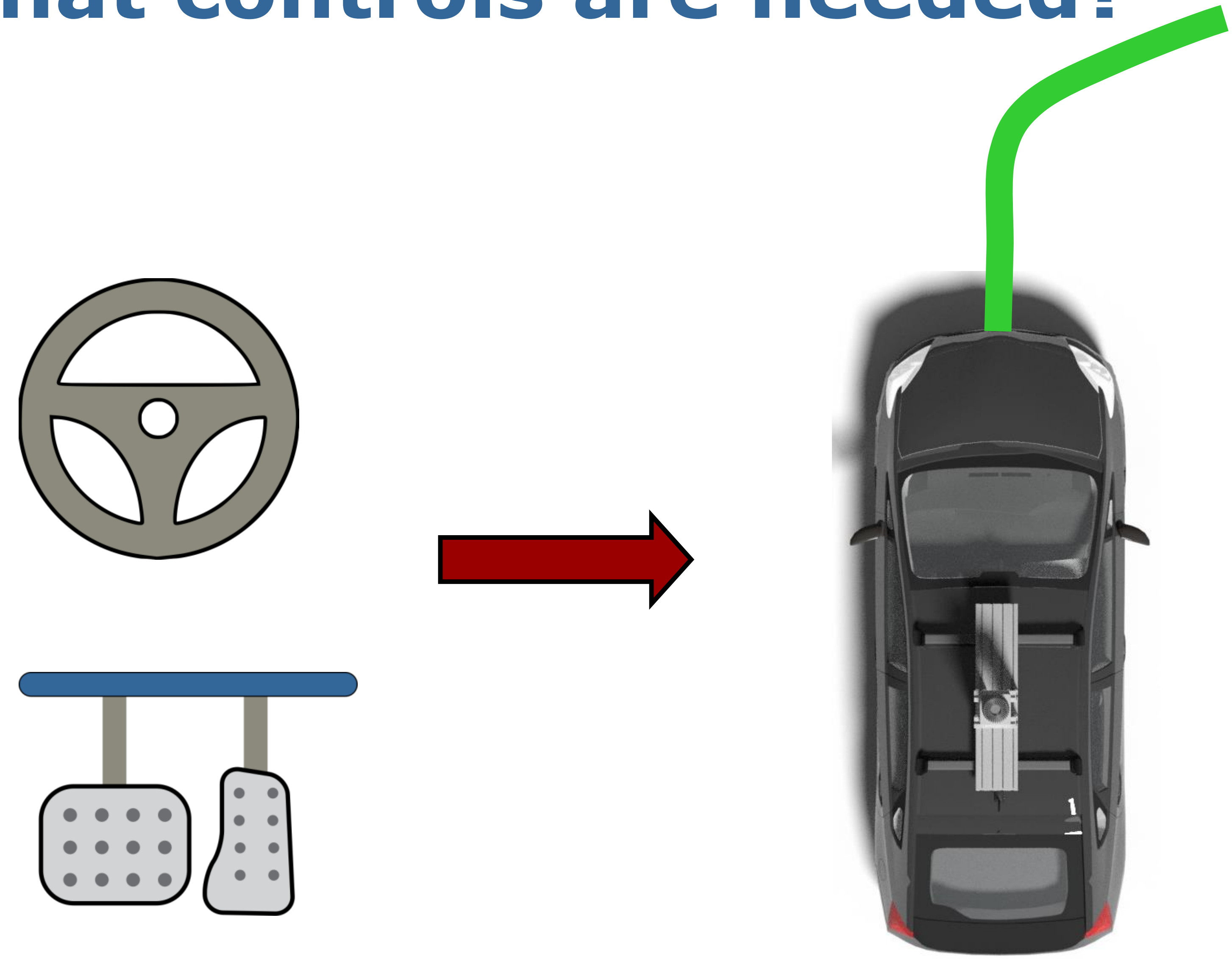
Control Strategy



How to follow a trajectory?



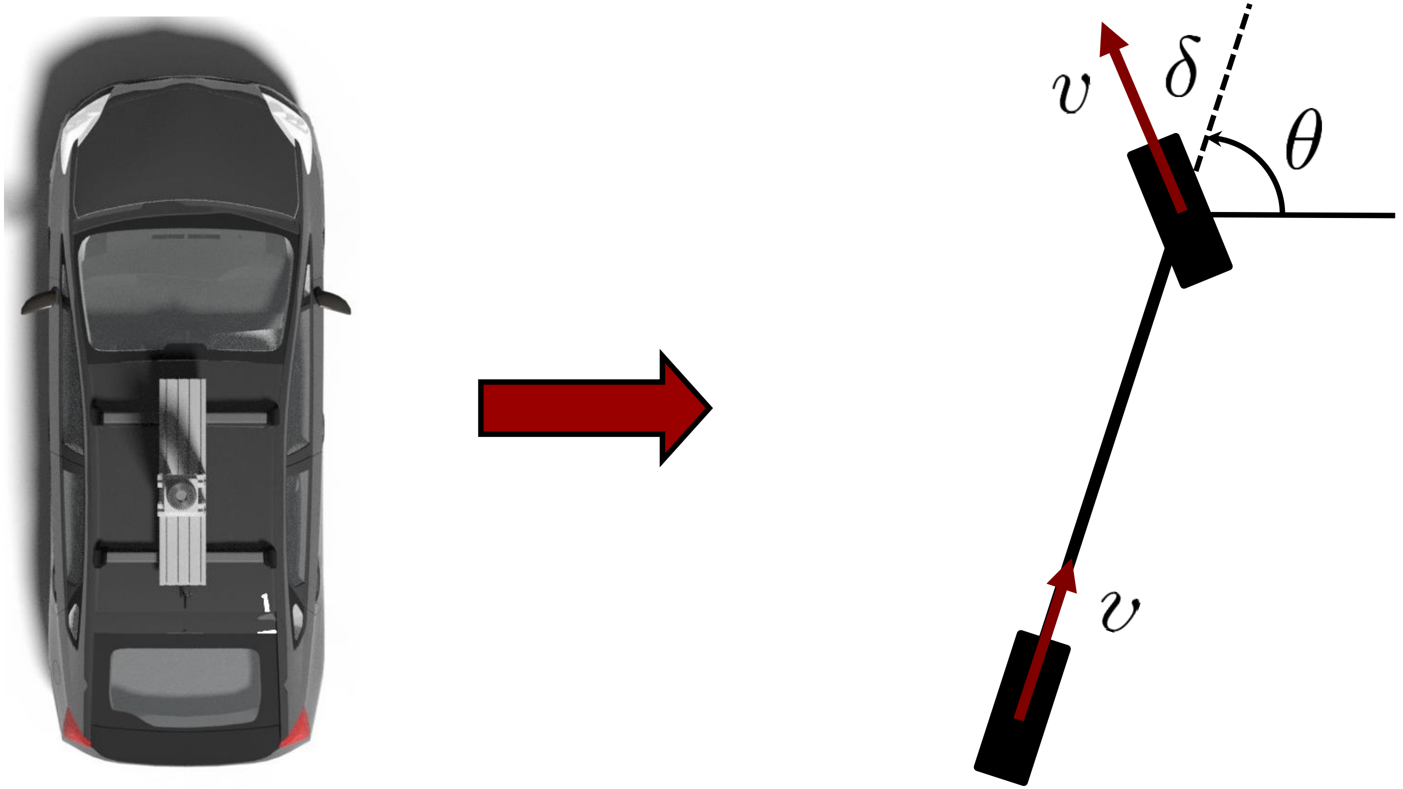
What controls are needed?



Understanding Motion

Kinematic Modelling

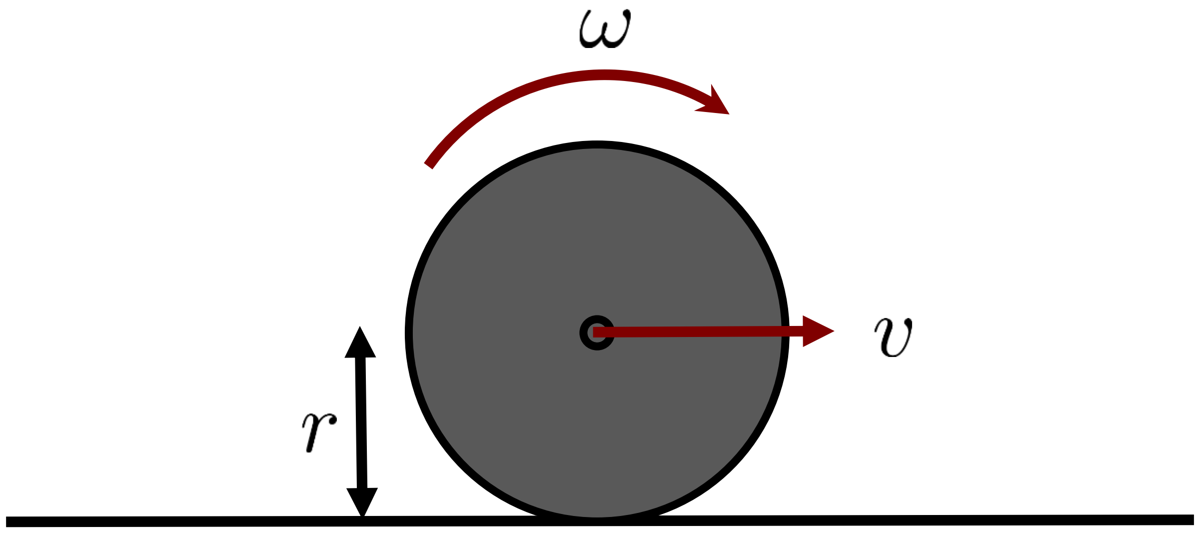
2D Bicycle Model



Rolling Condition for Wheels

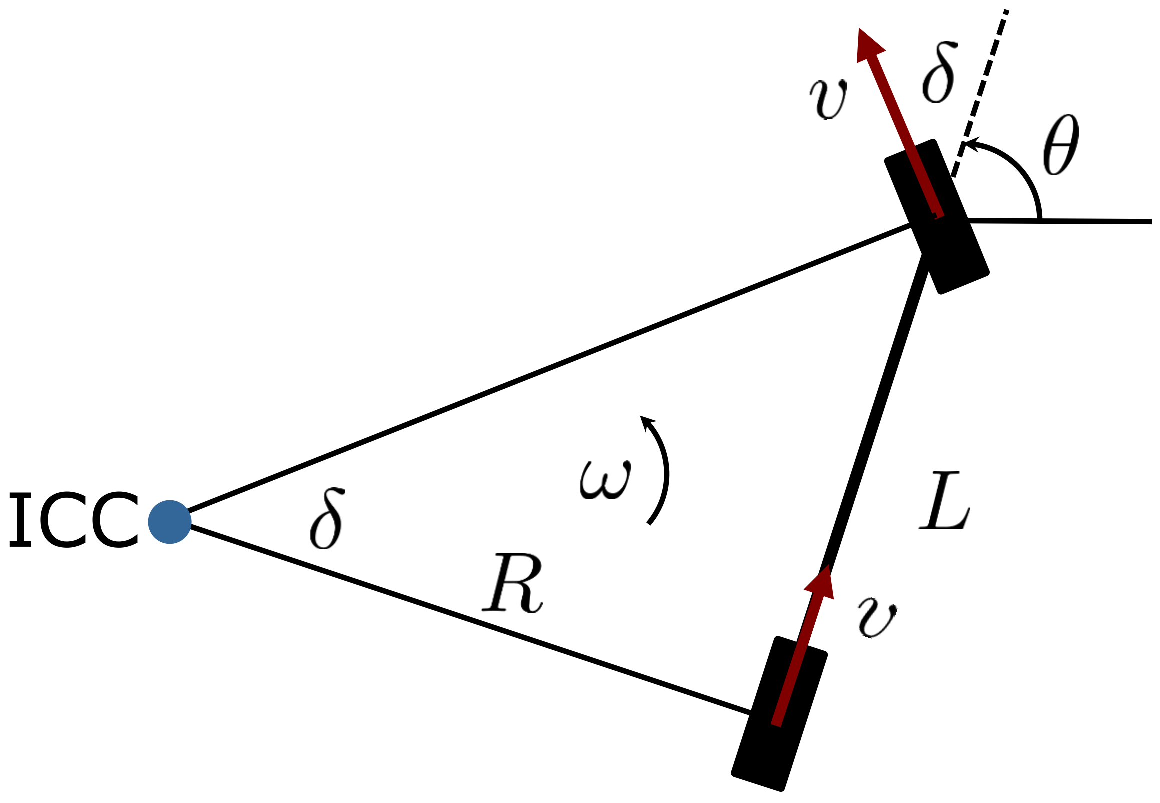
- Kinematic Constraint

$$v = r\omega$$



Instantaneous Center of Curvature

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



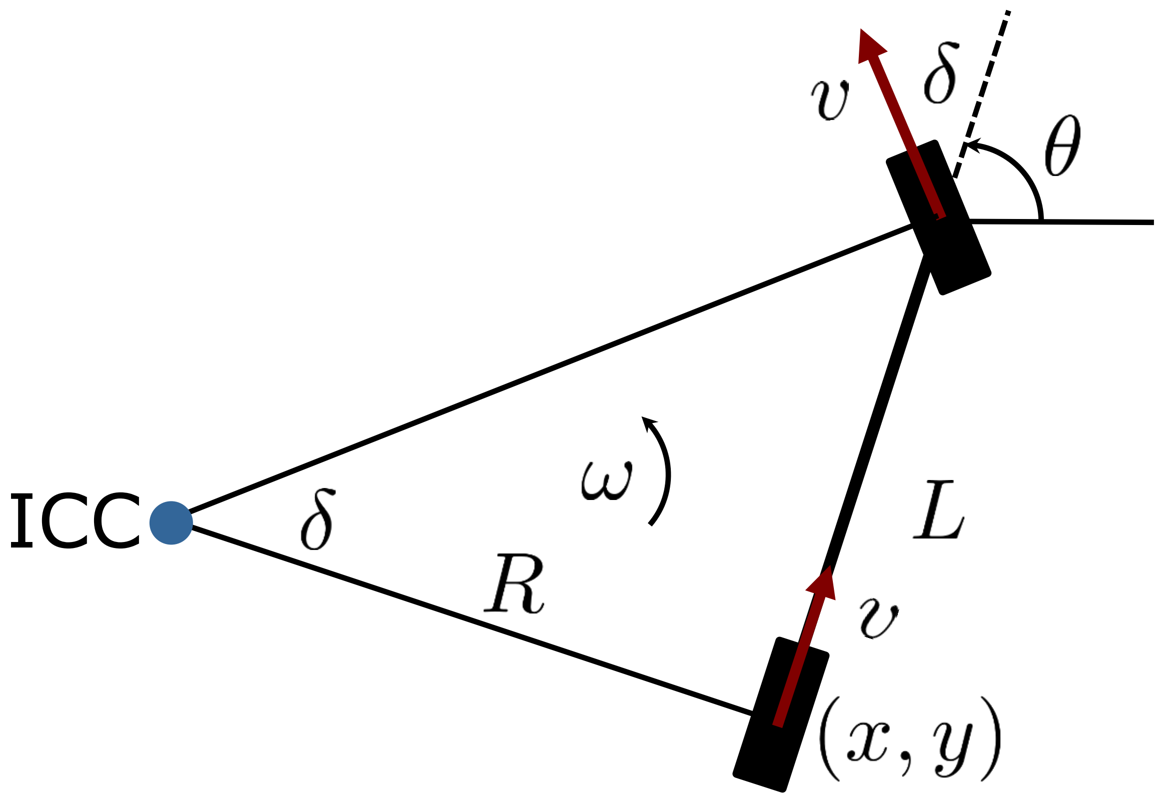
Bicycle Model Kinematics

- 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:

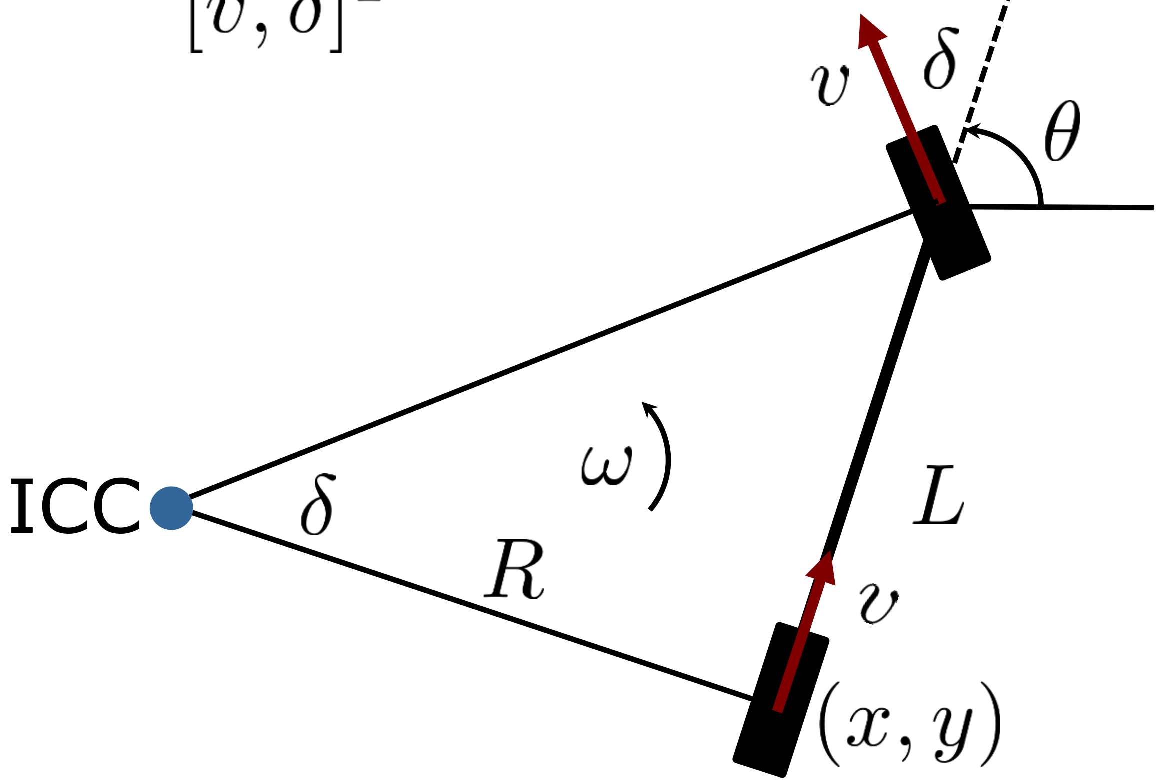
$$[v, \dot{\delta}]^T$$

Kinematics:

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

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

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

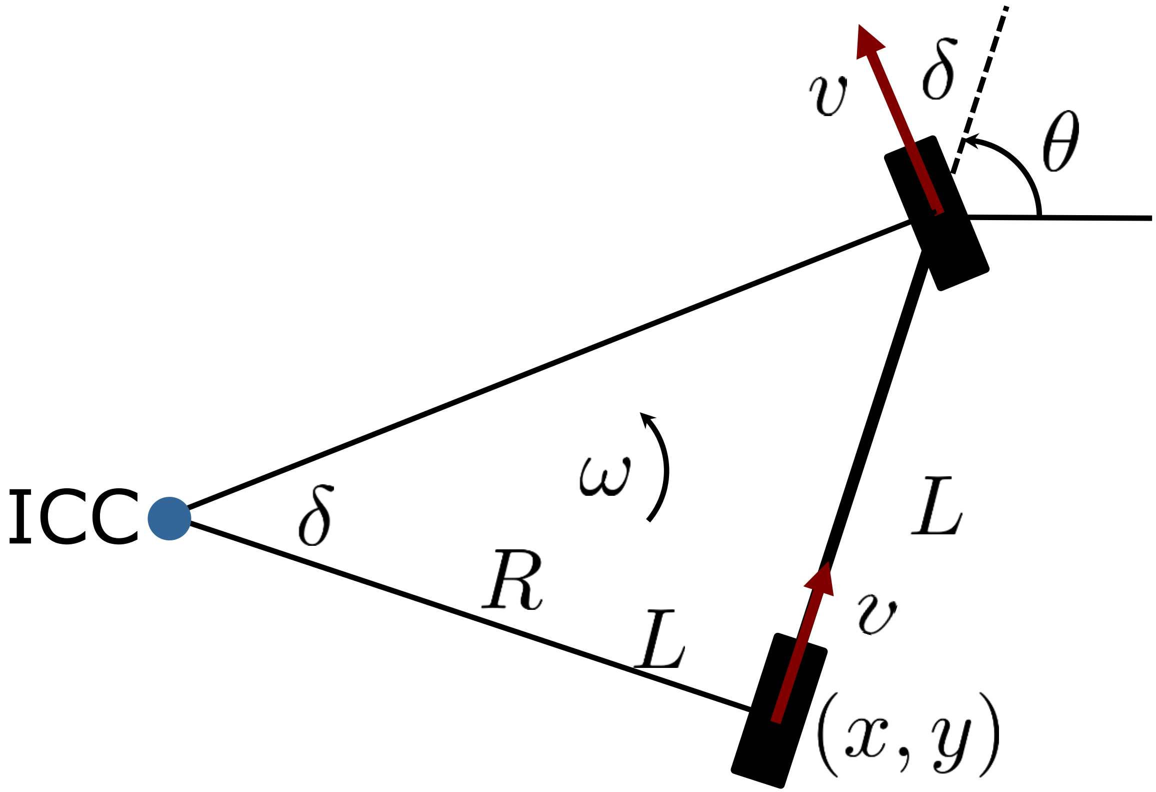


Bicycle Model

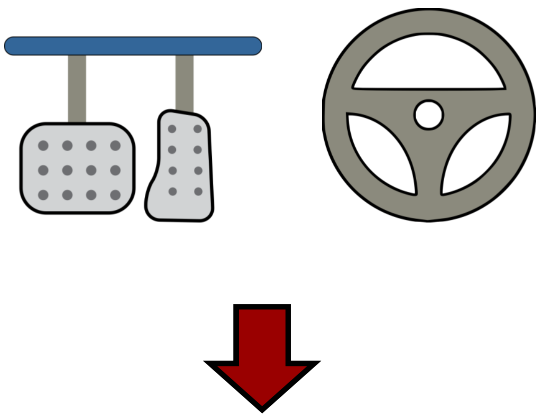
Constraints:

$$v < v_{\max}$$

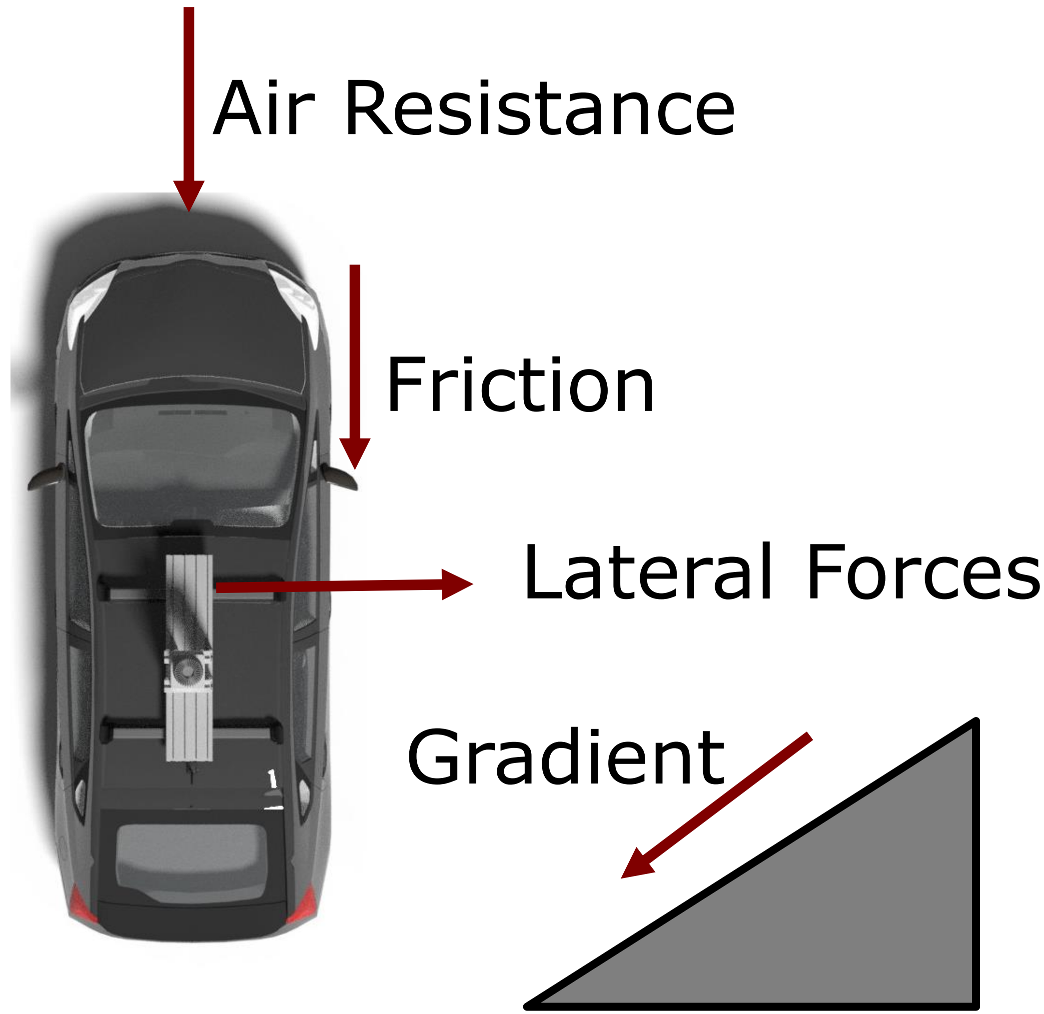
$$\delta < |\delta_{\max}|$$



Kinematic Vs. Dynamic Modeling



$$v, \omega$$

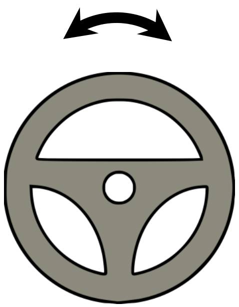


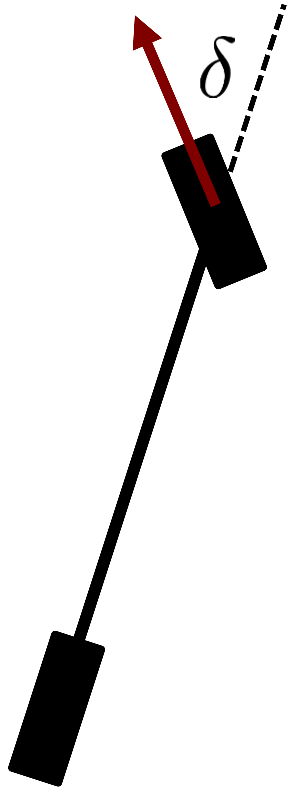
Vehicle Actuation

Steering Model

$$\delta_s = k_s \delta$$

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

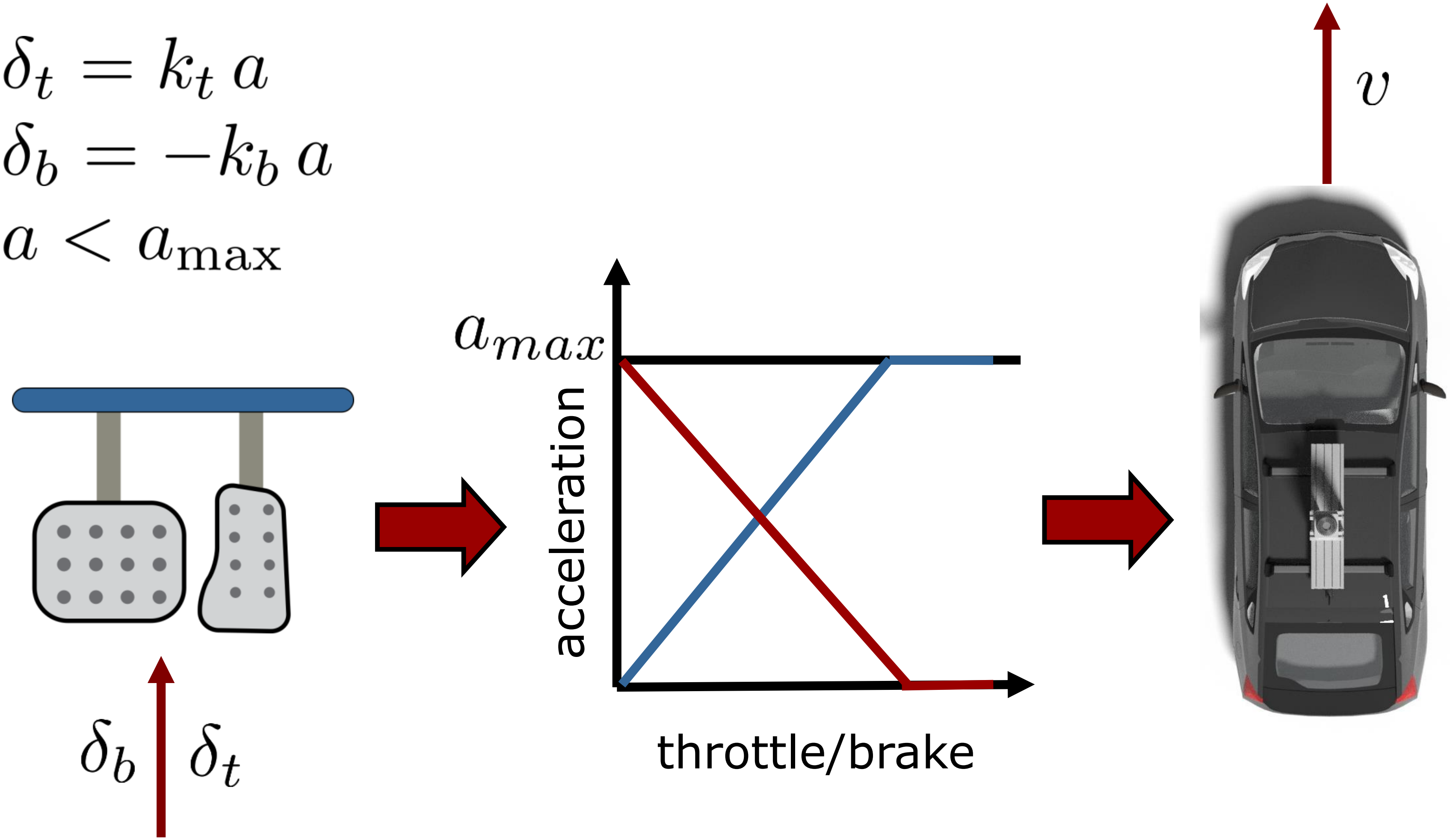




Vehicle Actuation

Throttle/Brake

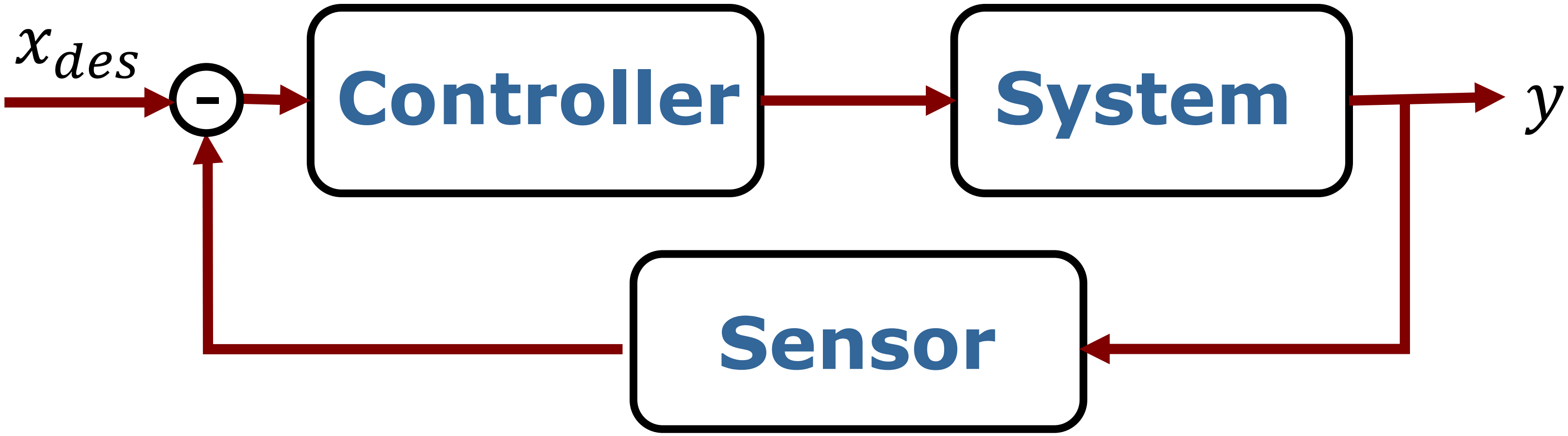
$$\delta_t = k_t\, a$$
$$\delta_b = -k_b\, a$$
$$a < a_{\max}$$



Feedback Control

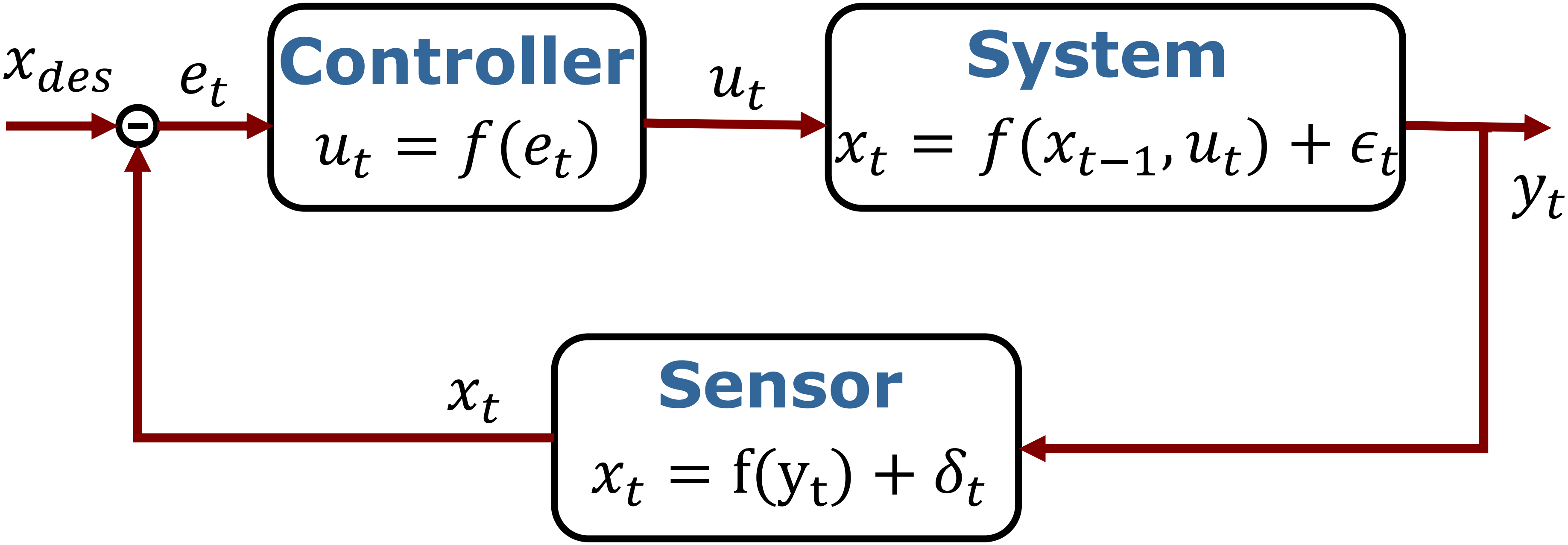
Open Loop vs. Feedback Control

(a) Open loop control

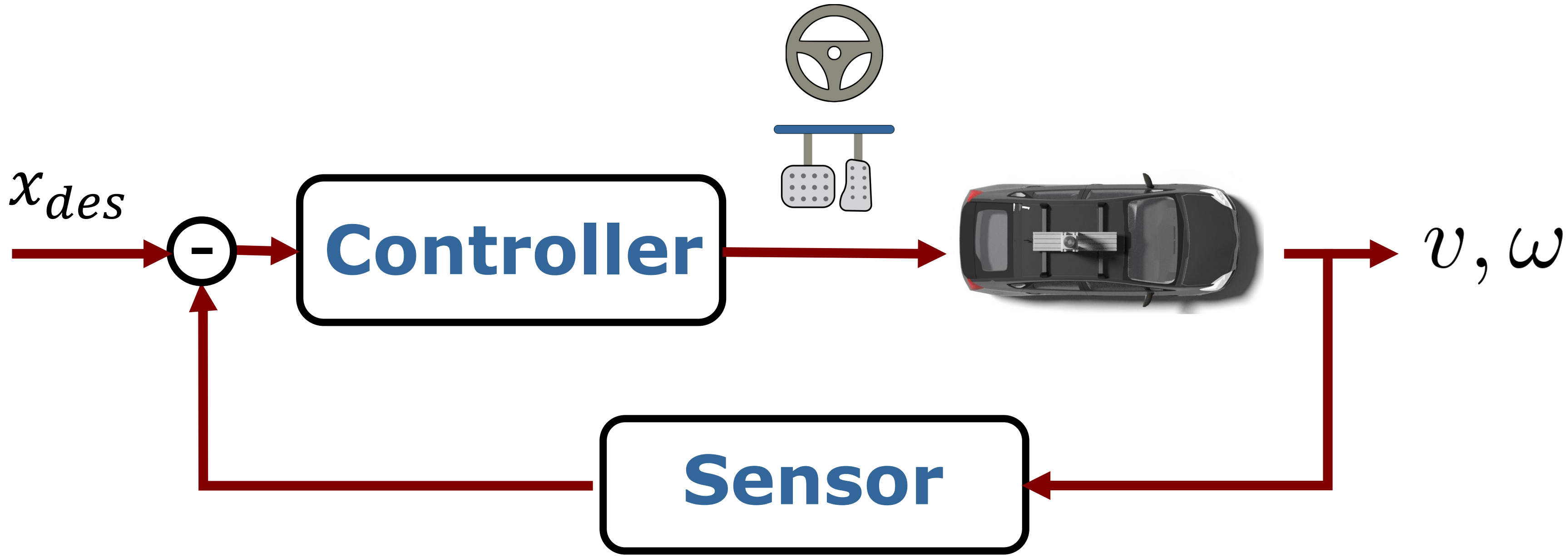


(b) Feedback control

Feedback Control



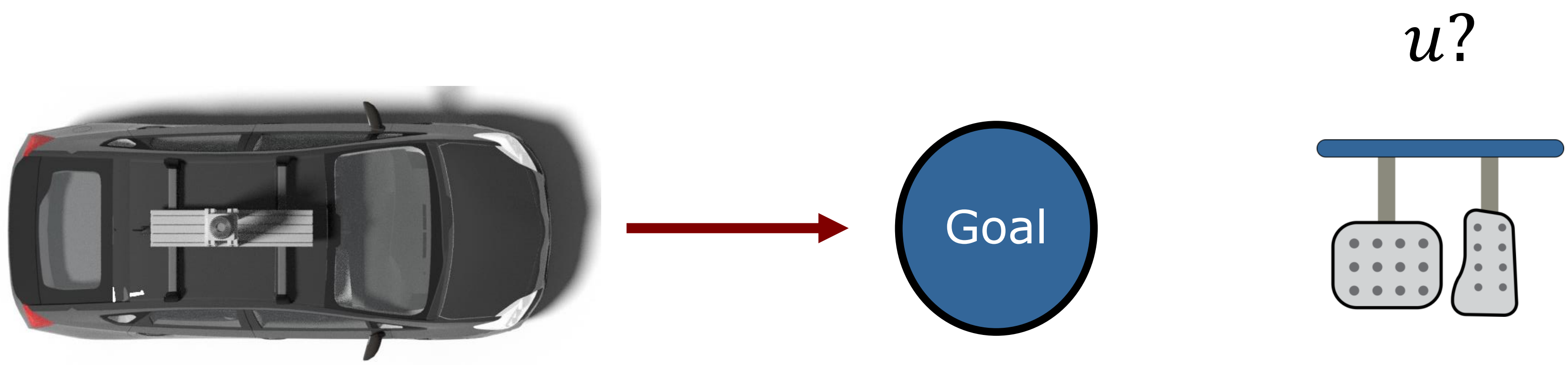
Feedback Control



PID Controller

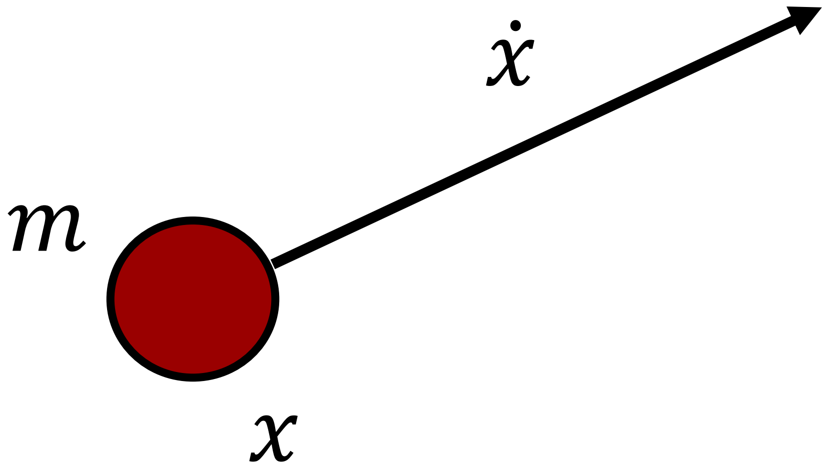
Position Control Task

- 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

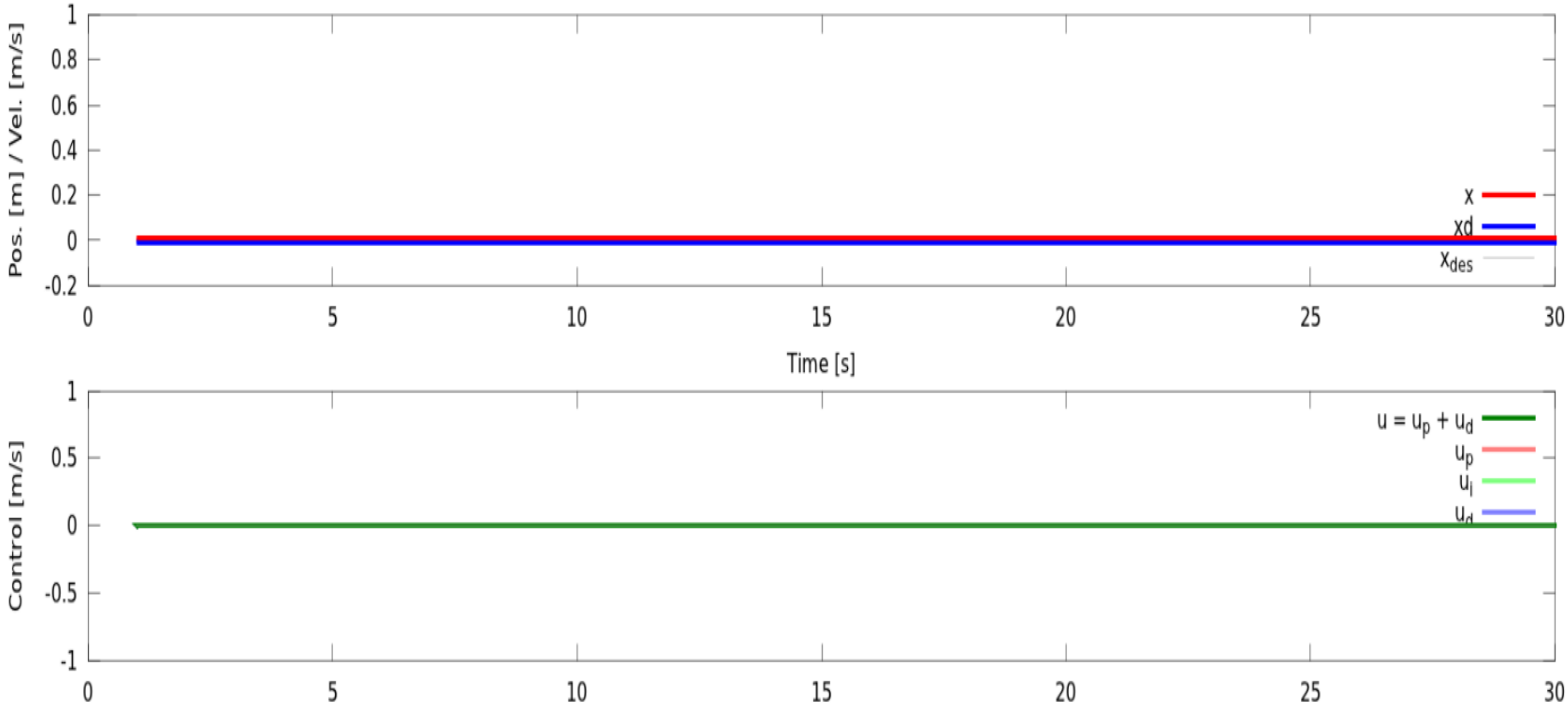


Position Control Task

- 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

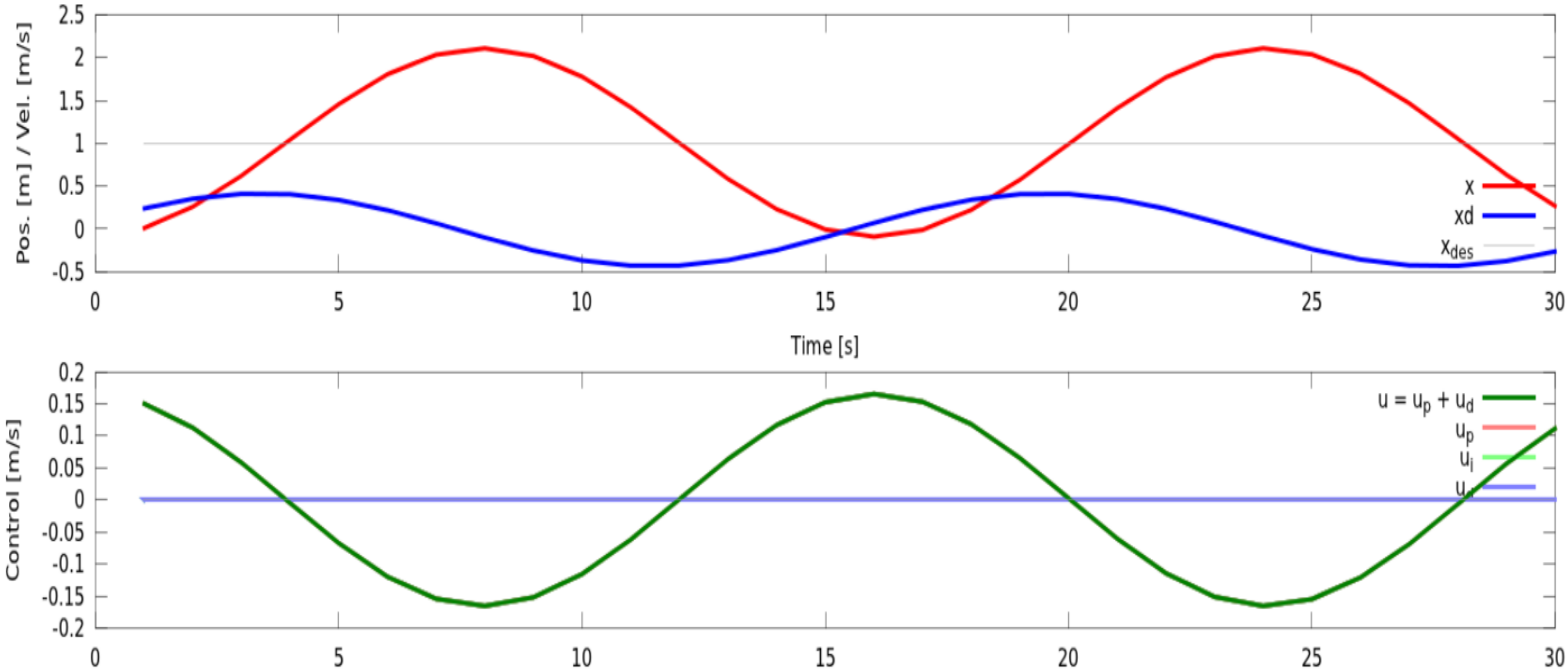
- System model : $x_t = x_{t-1} + \dot{x}\Delta t$
- Initial state: $x_0 = 0, \dot{x}_0 = 0$



P Control

- Proportional control law

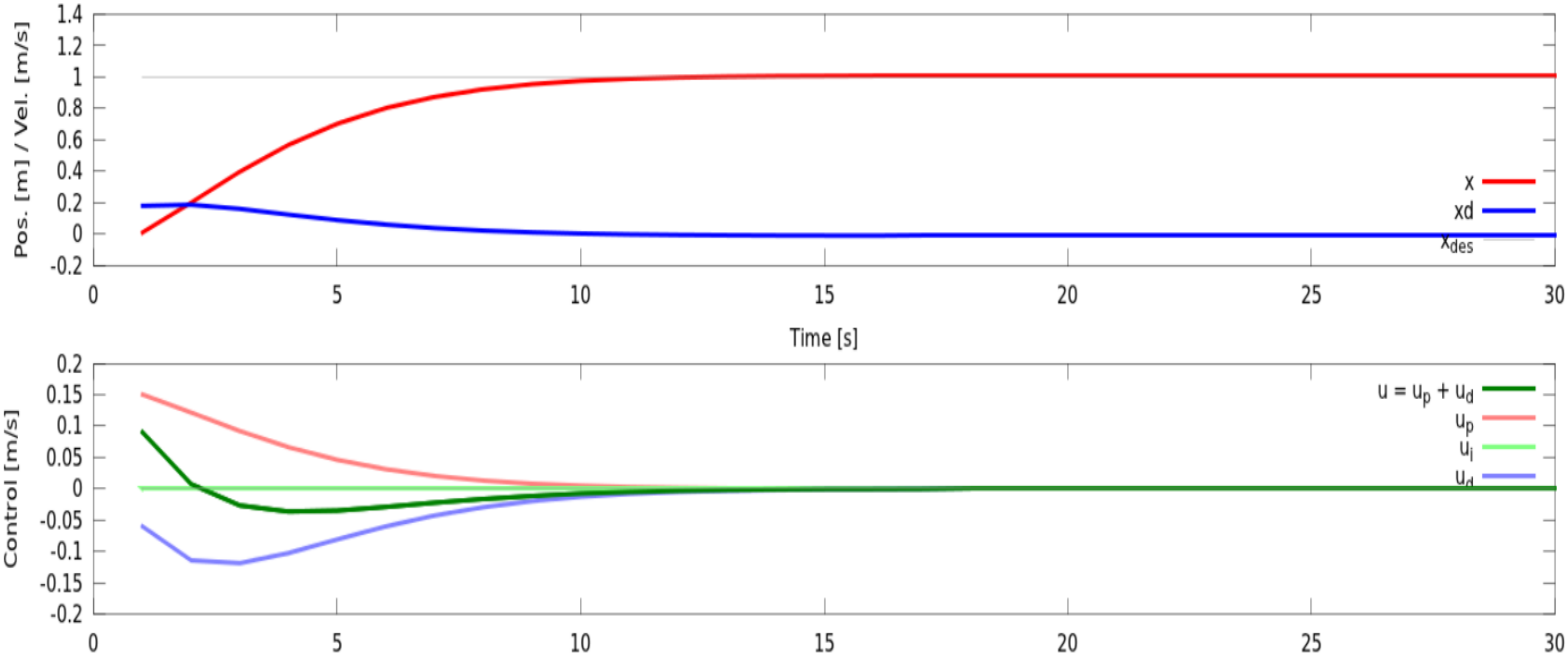
$$u_t = K_P(x_{des} - x_t)$$



PD Control

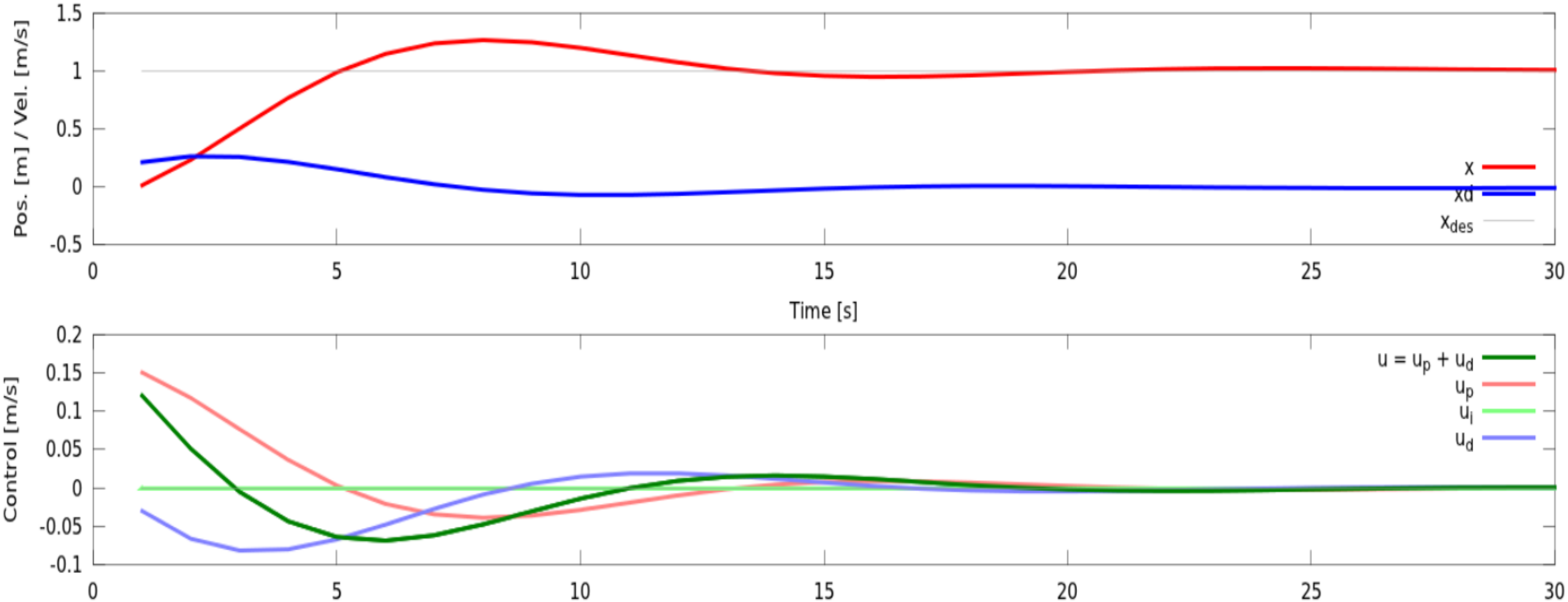
- Proportional-derivative control law

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



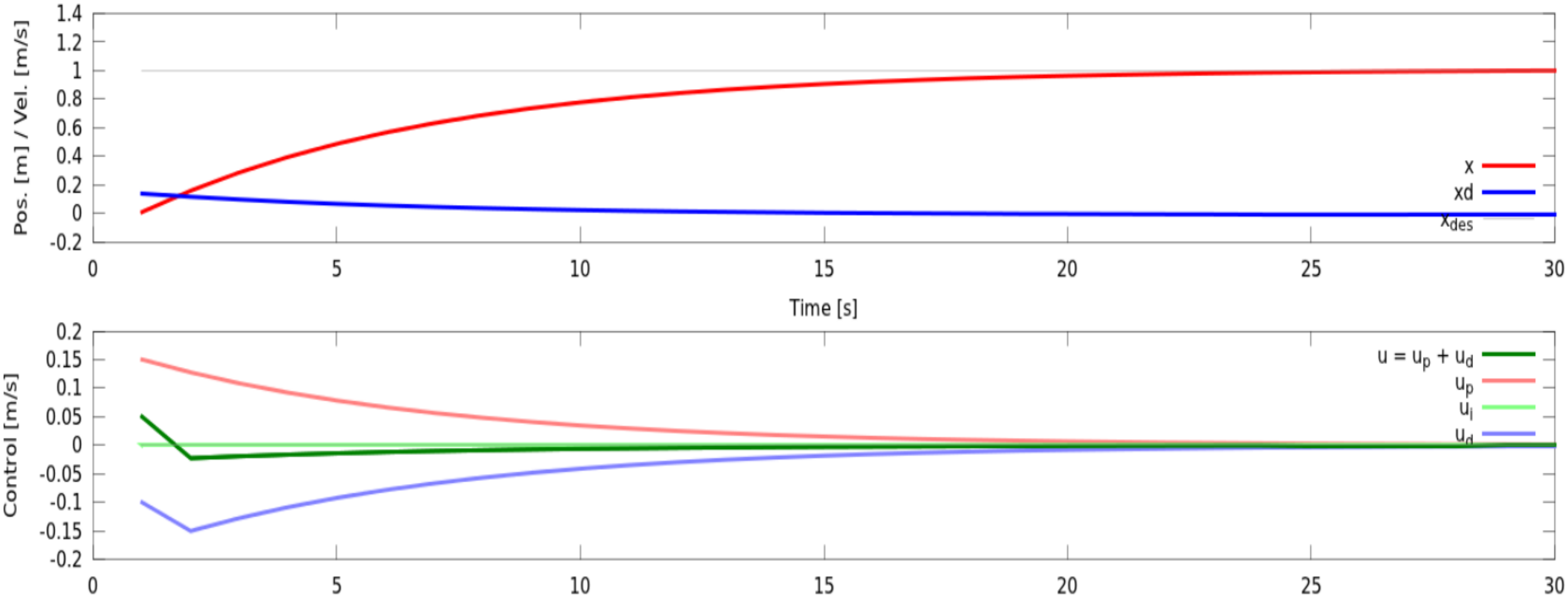
PD Control

- 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?



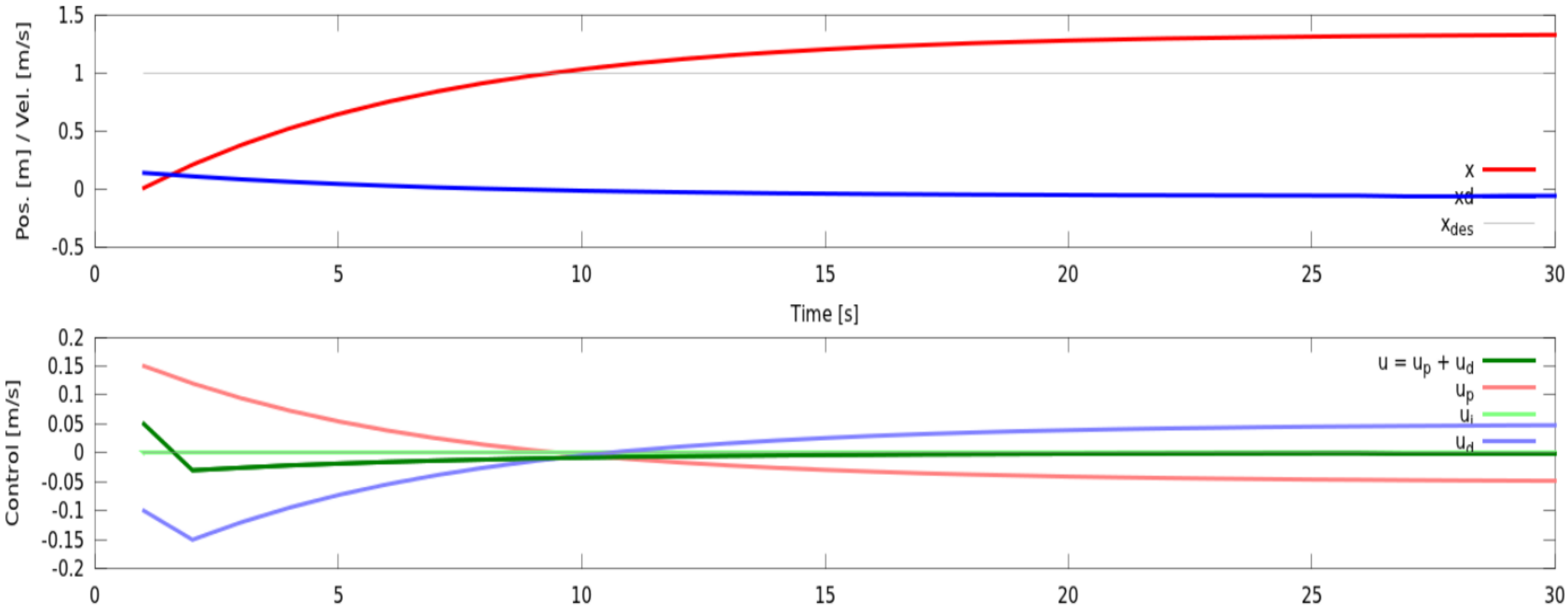
PD Control

- 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 …



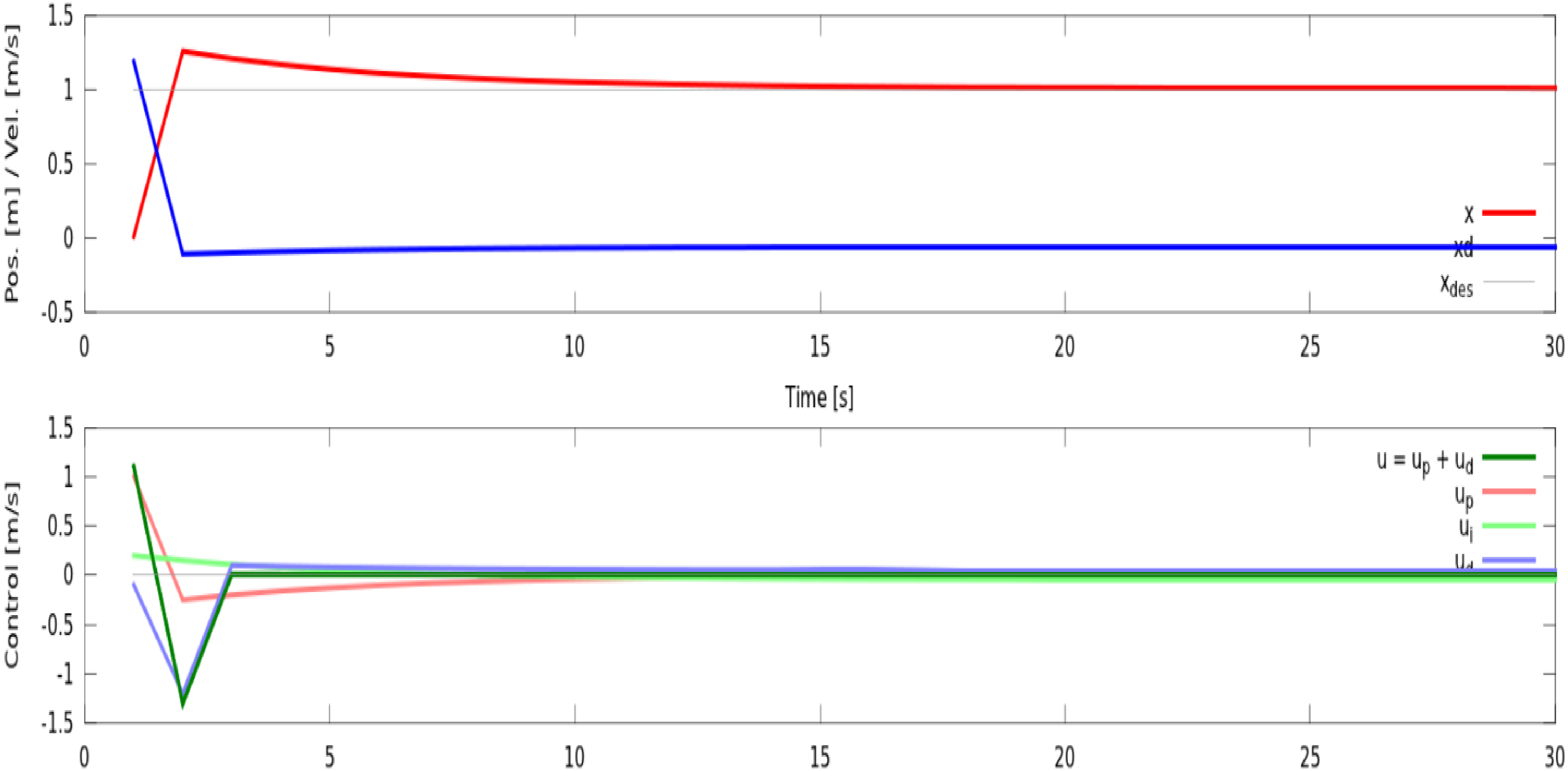
PID Control

- **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 …

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

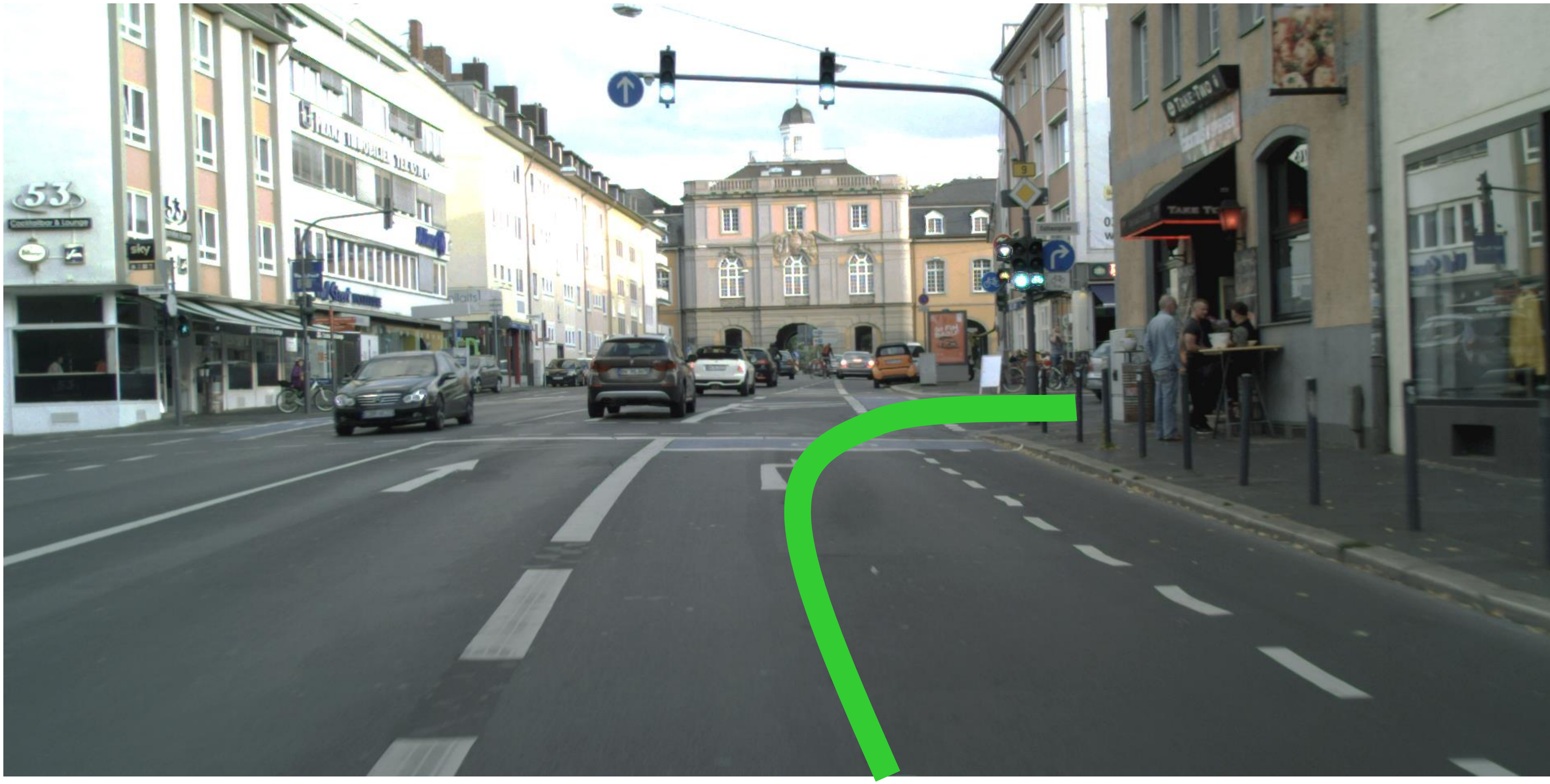
- Reasonable for steady state system
- May be dangerous to error build up (wind-up effect)

PID Control - Summary

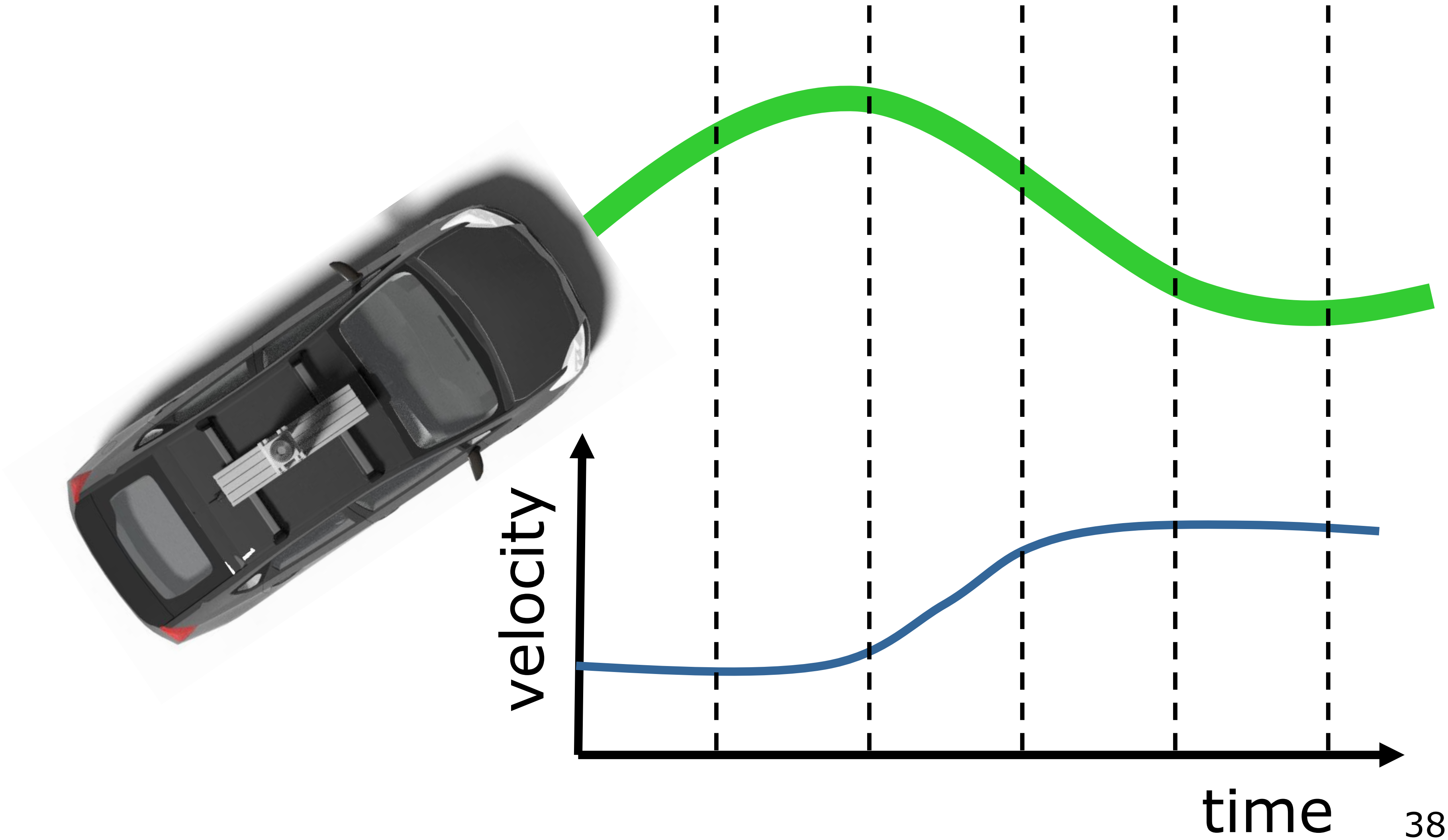
- P = simple proportional control, sufficient in most cases.
- PD = reduce overshoot (e.g. when acceleration can be controlled)
- PI = compensate for systematic error/bias
- PID = combination of the above properties.

Following A Trajectory

How to follow a trajectory?



Longitudinal Control



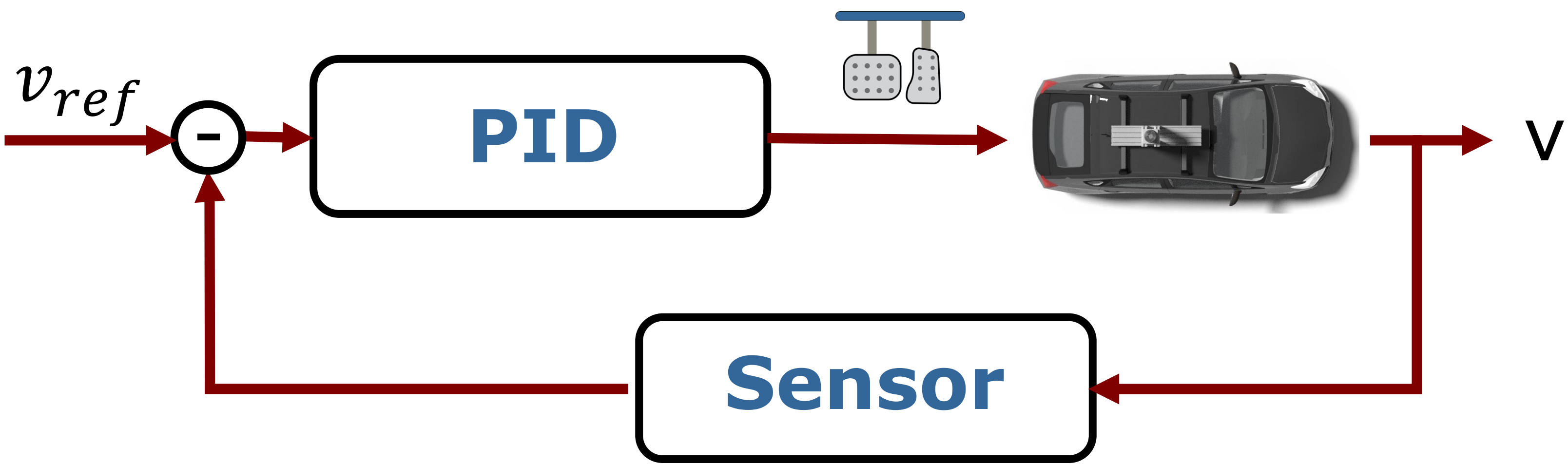
Longitudinal PID Controller

Desired acceleration

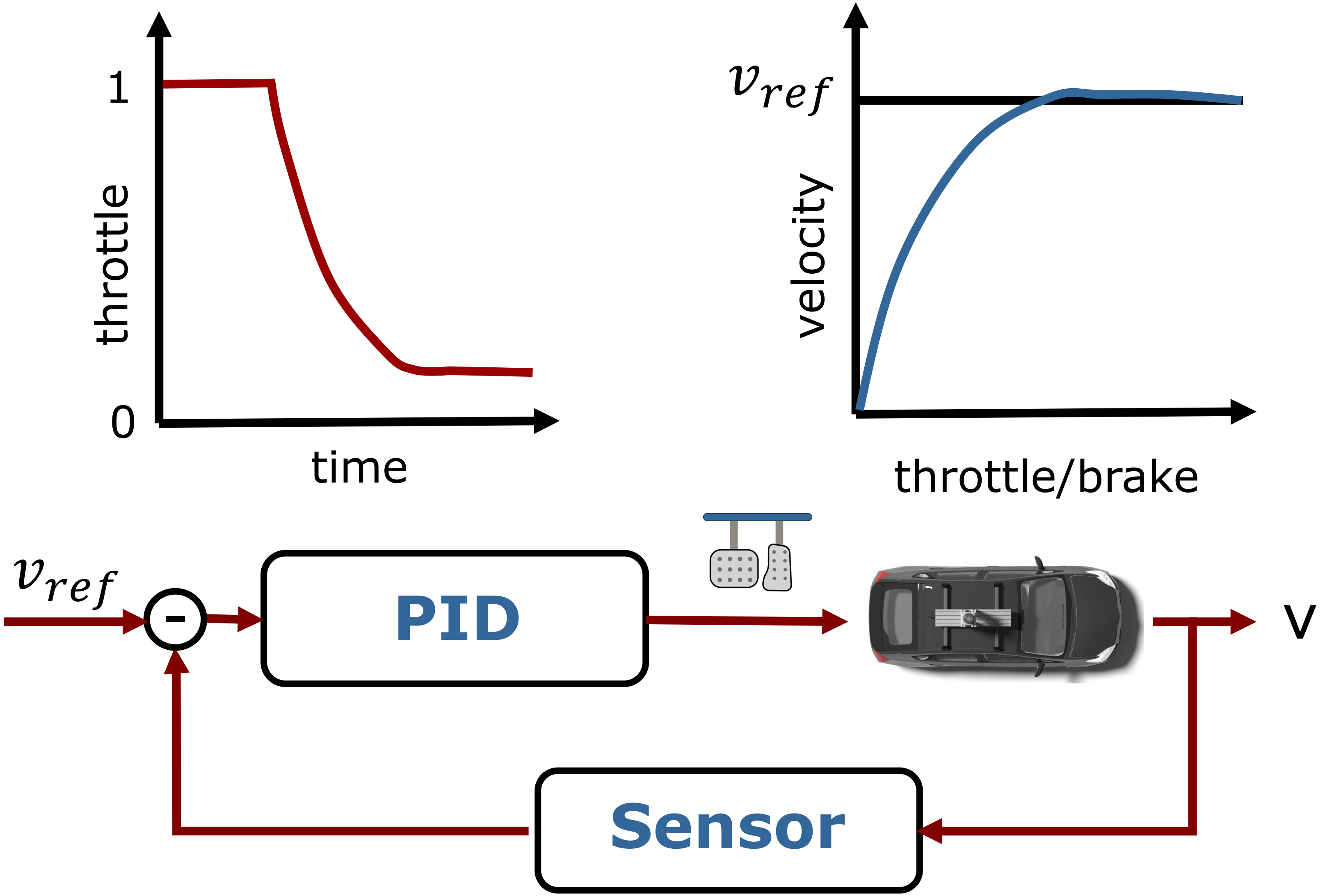
Reference velocity

car velocity

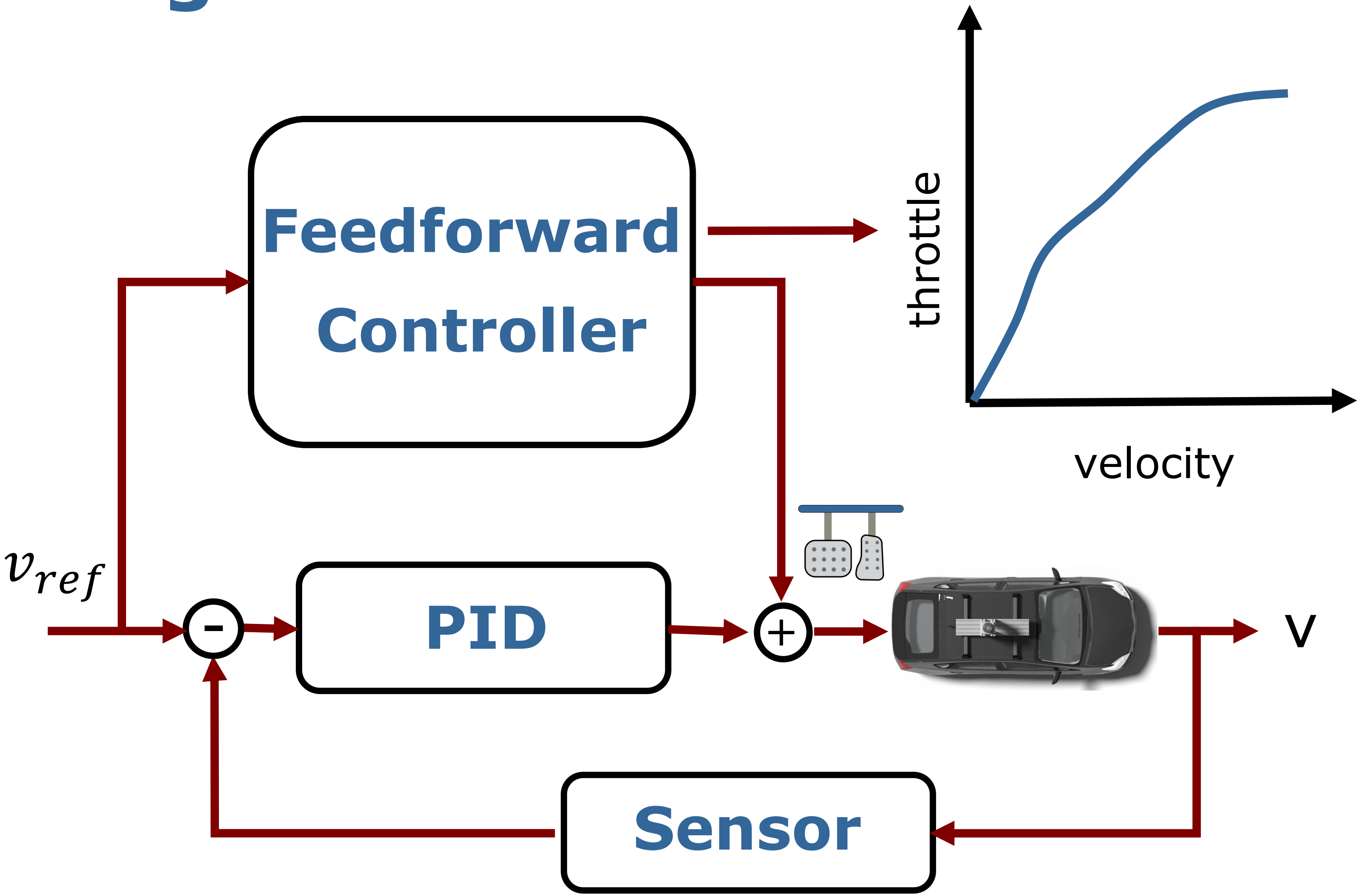
$$\ddot{x}_{des} = K_P(\dot{x}_{ref} - \dot{x}) + K_D \frac{d(\dot{x}_{ref} - \dot{x})}{dt} + K_I \int_0^t (\dot{x}_{ref} - \dot{x})dt$$



Longitudinal PID Controller

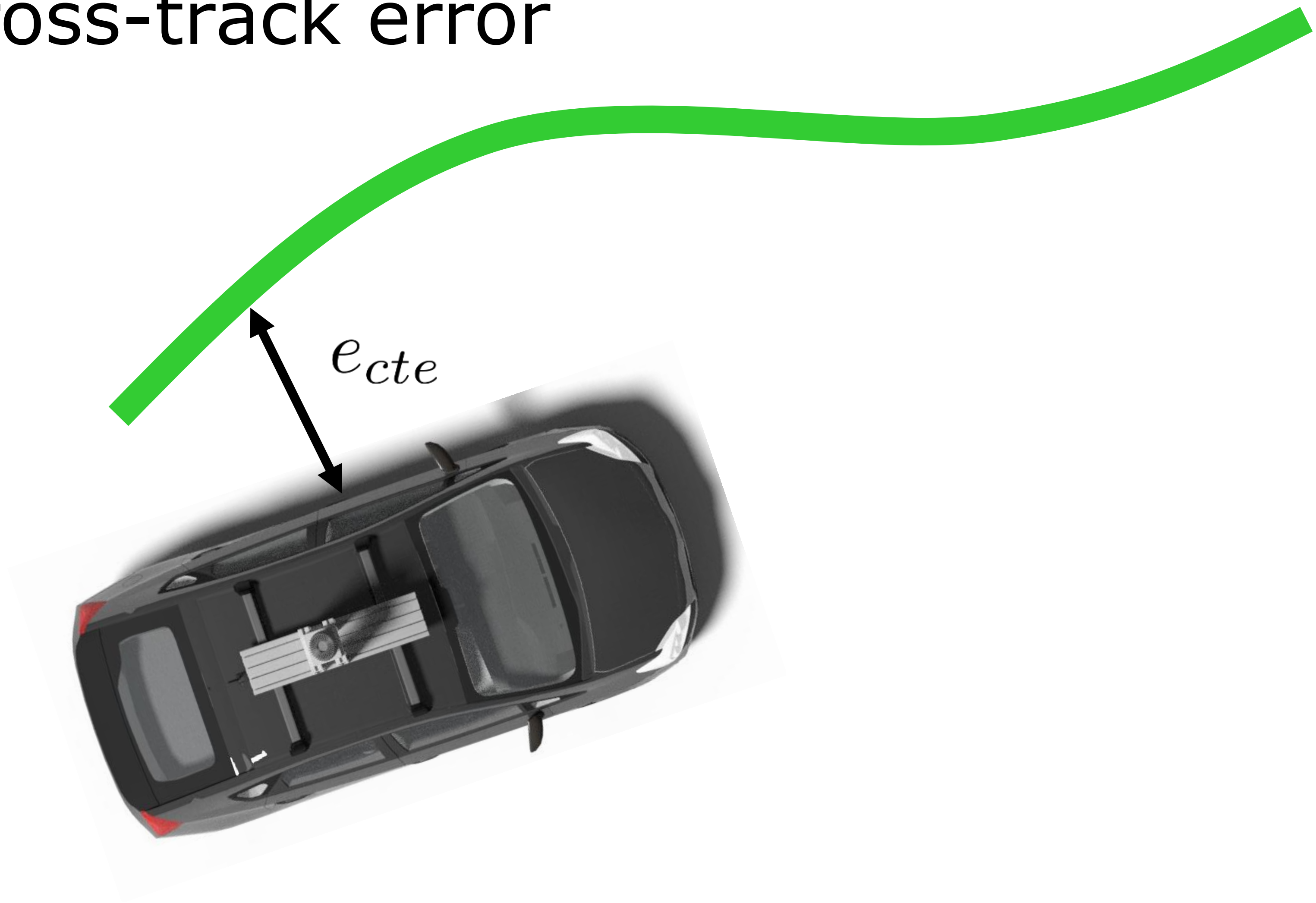


Longitudinal PID Controller



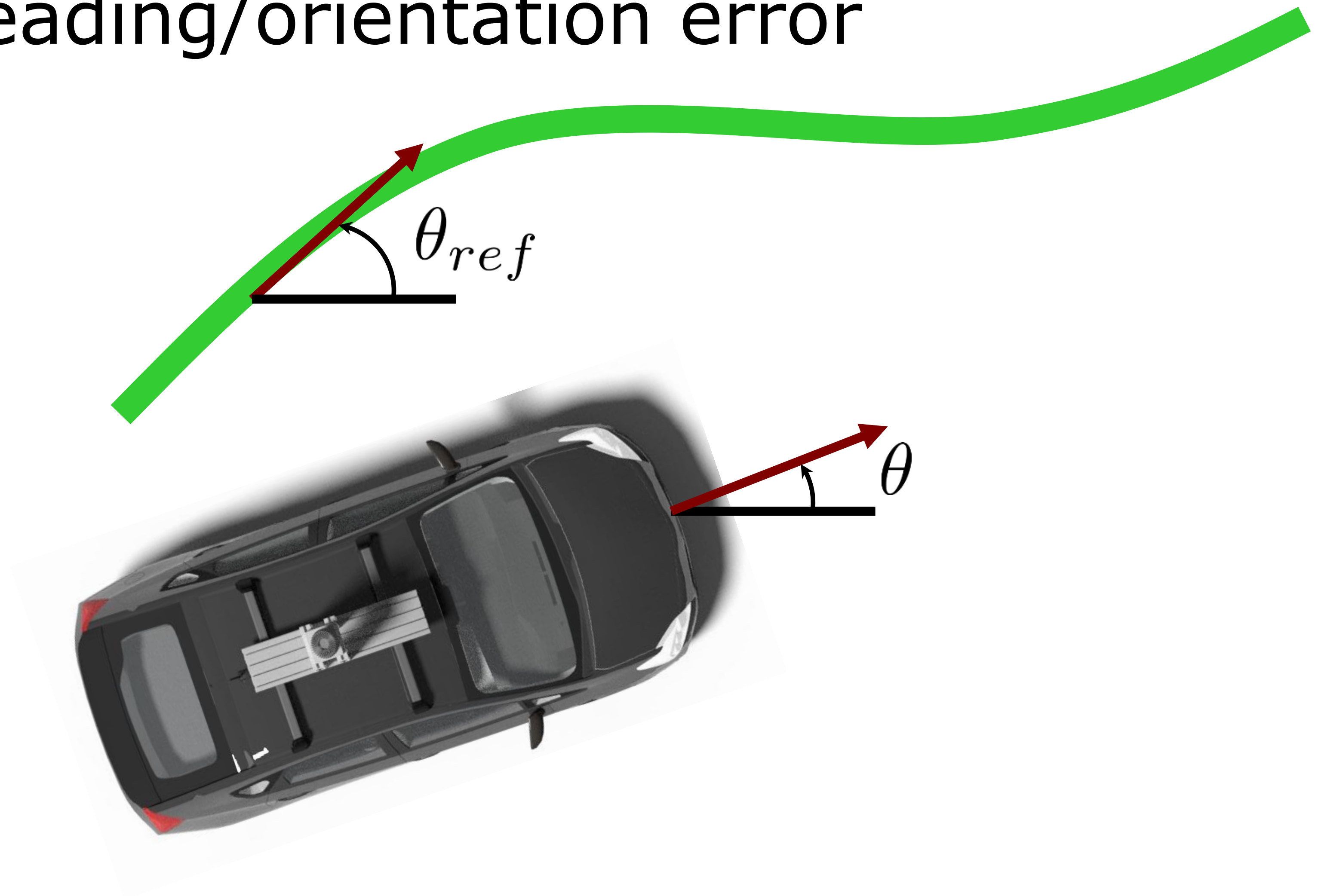
Lateral Control

- Cross-track error



Lateral Control

- Heading/orientation error

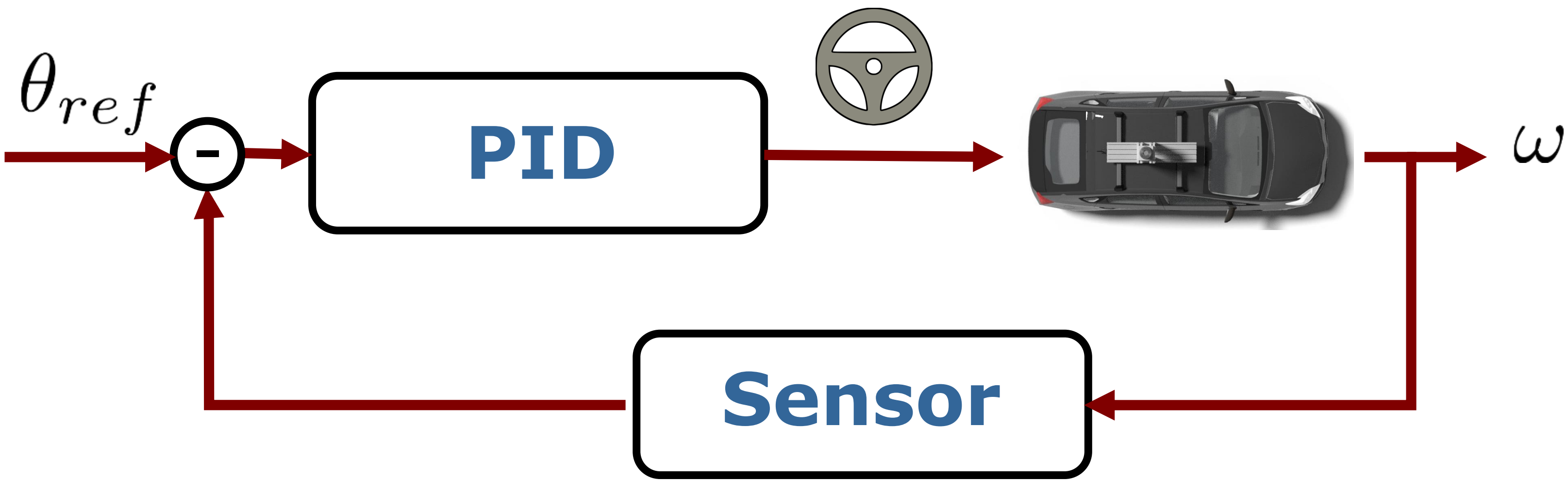


Lateral PID Controller

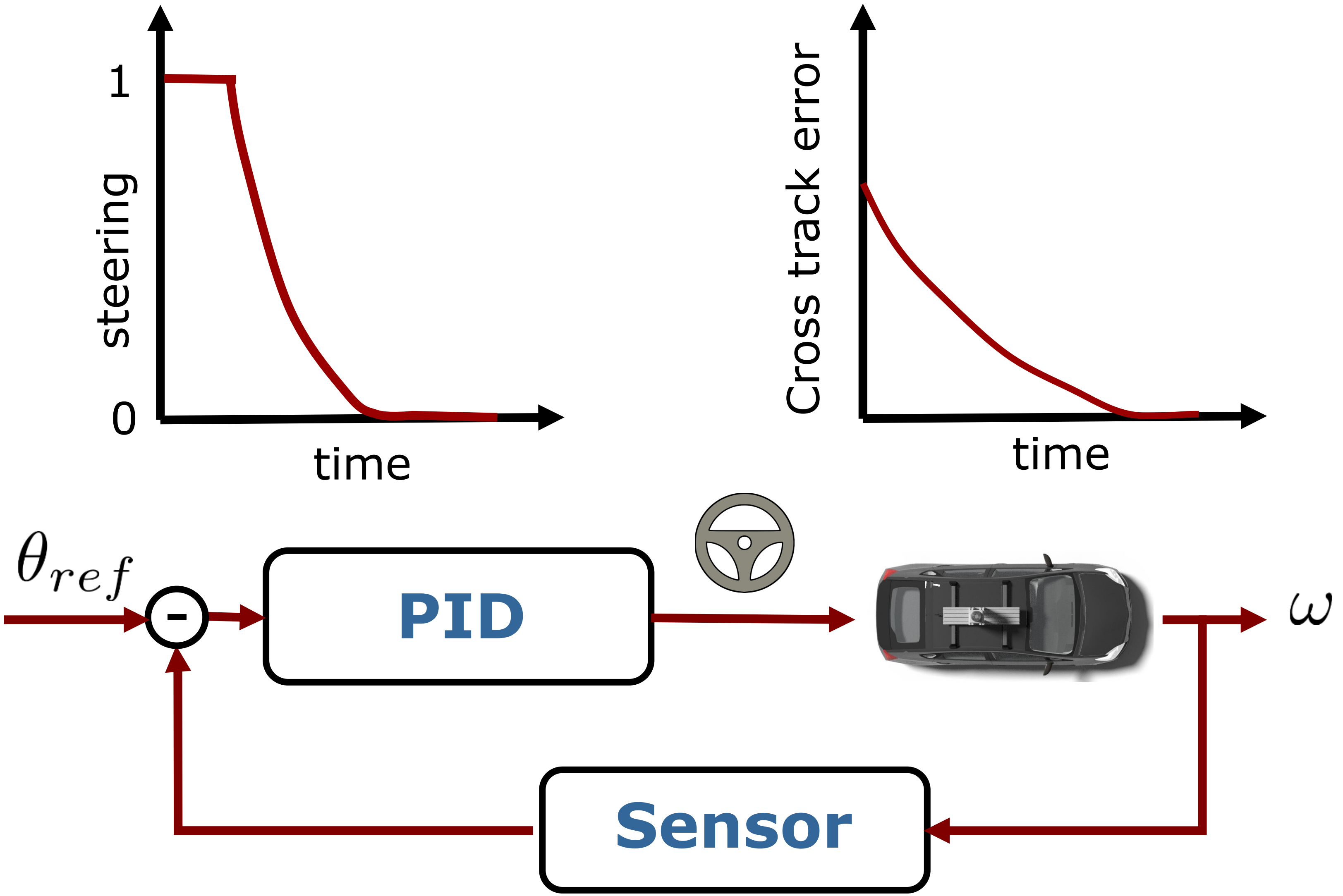
Desired steering rate

Cross-track error

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

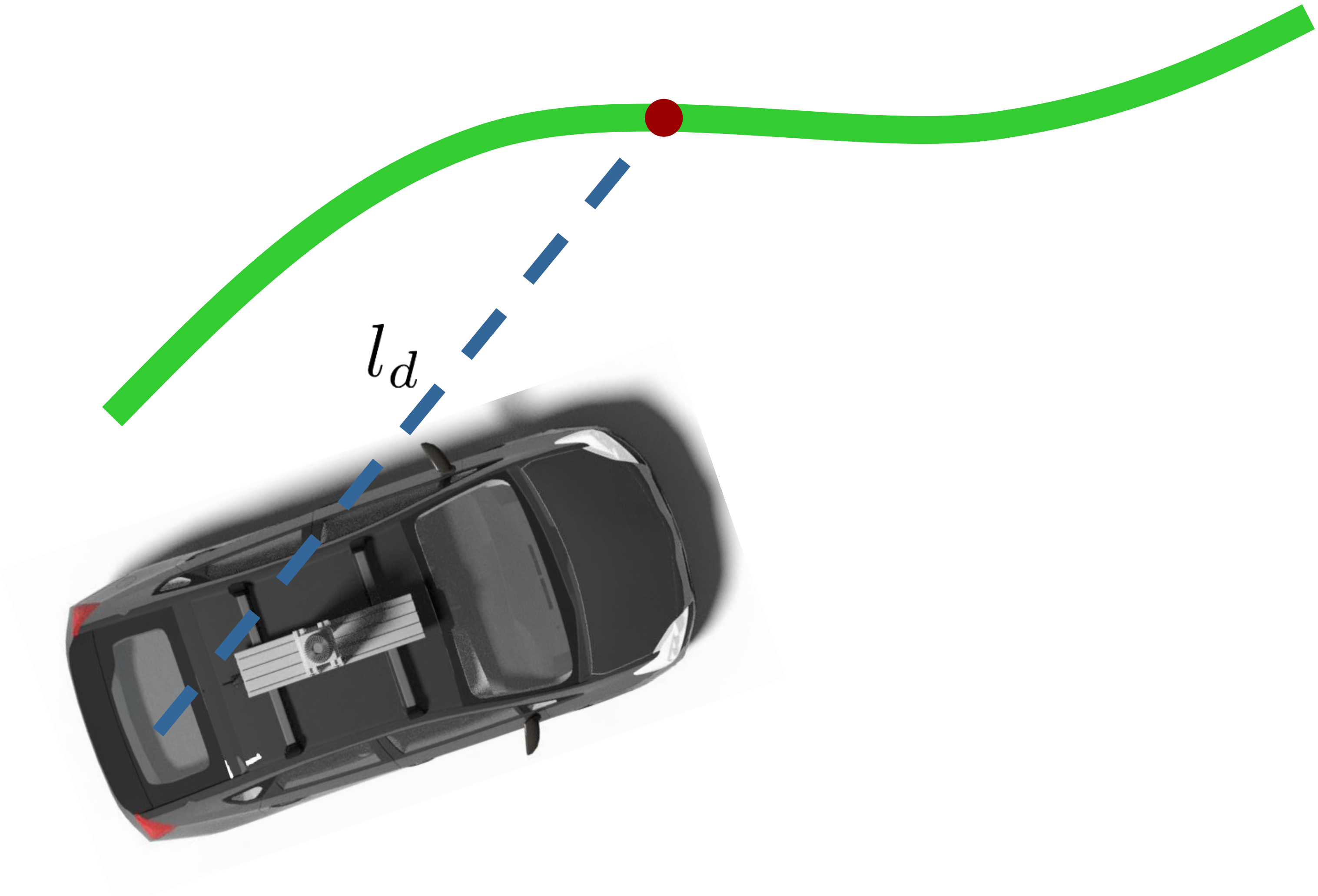


Lateral PID Controller

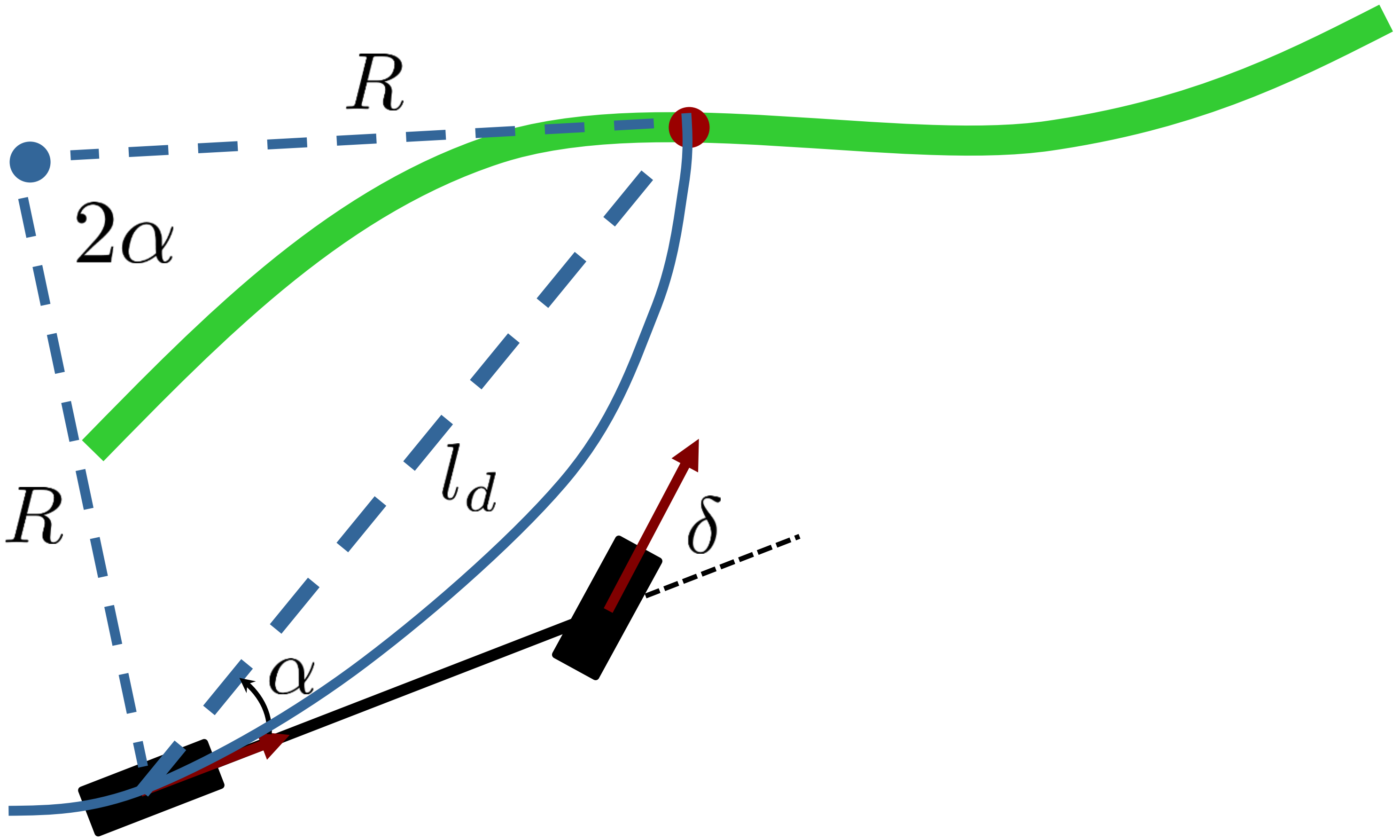


Geometric Steering Control

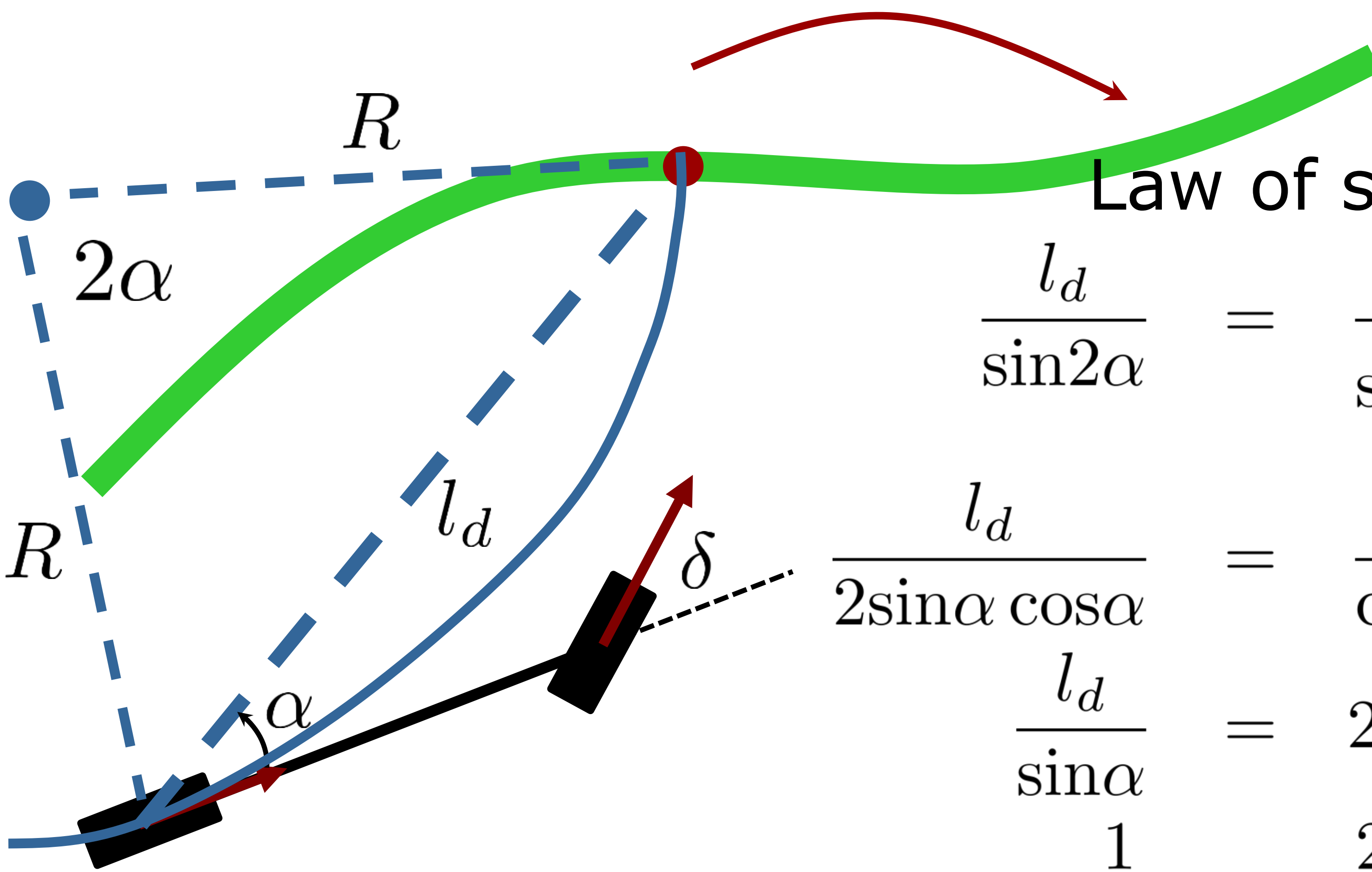
Pure Pursuit Controller



Pure Pursuit Controller



Pure Pursuit Controller



Law of sines

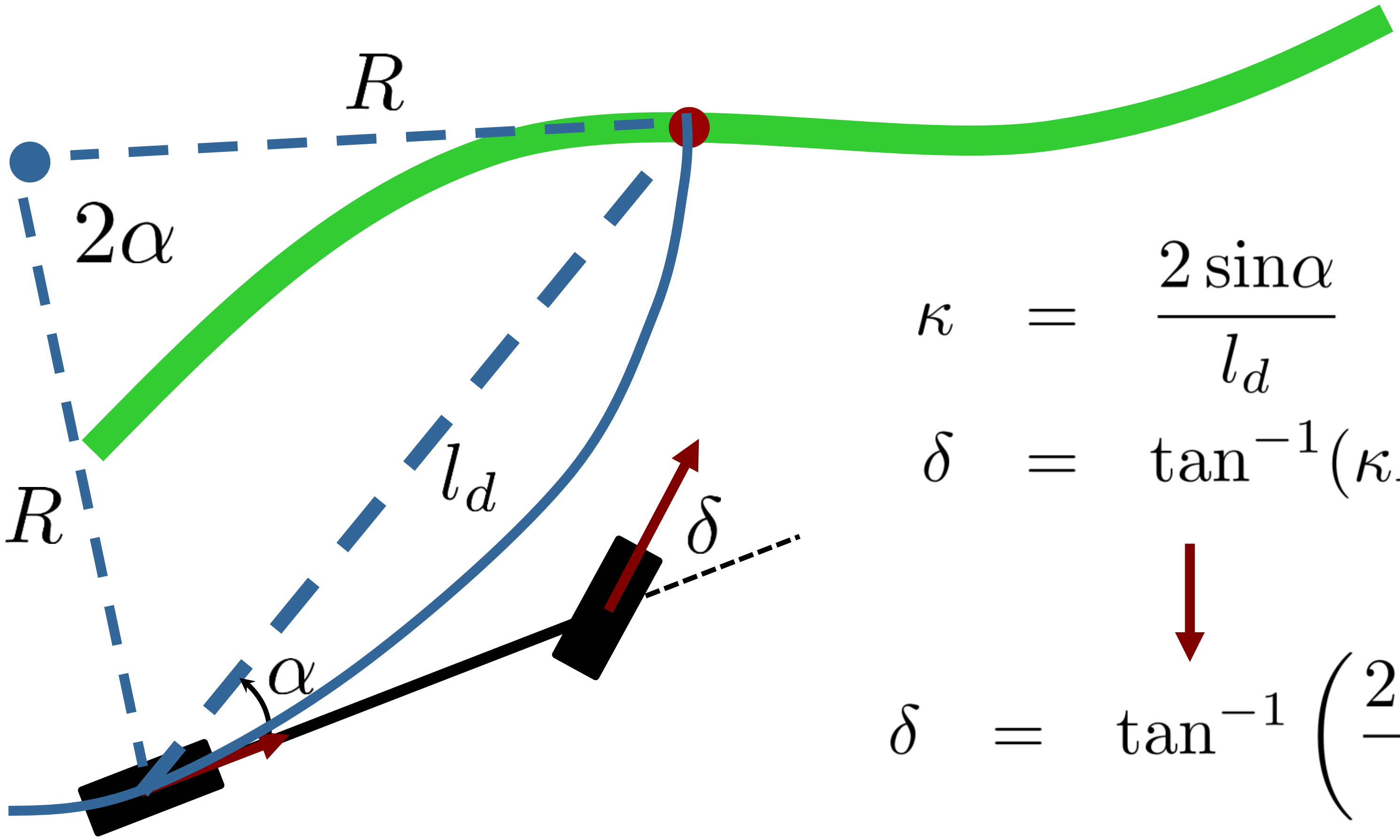
$$\frac{l_d}{\sin 2\alpha} = \frac{R}{\sin\left(\frac{\pi}{2} - \alpha\right)}$$

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

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

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

Pure Pursuit Controller



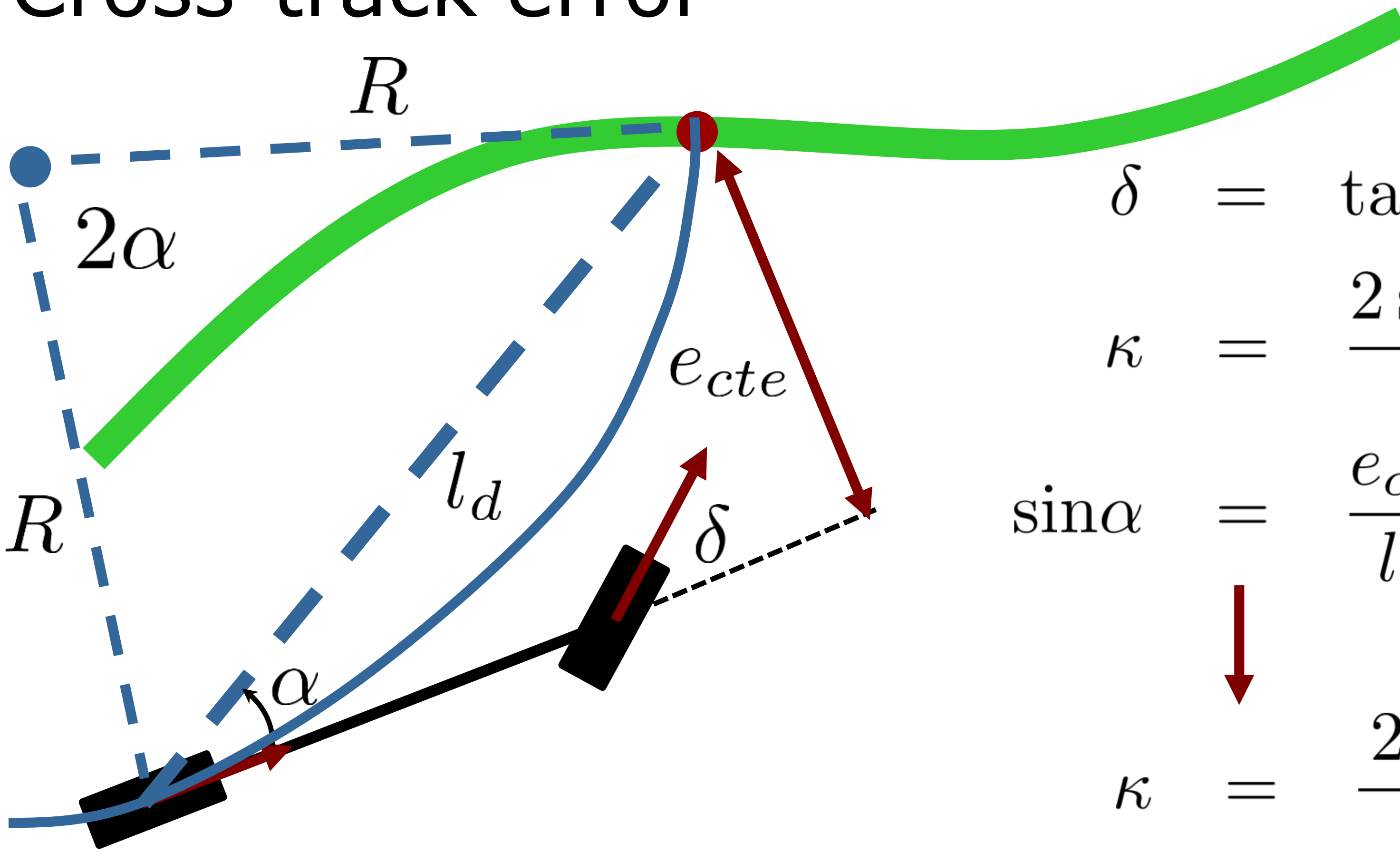
$$\kappa = \frac{2\sin\alpha}{l_d}$$

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

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

Pure Pursuit Controller

- Cross-track error



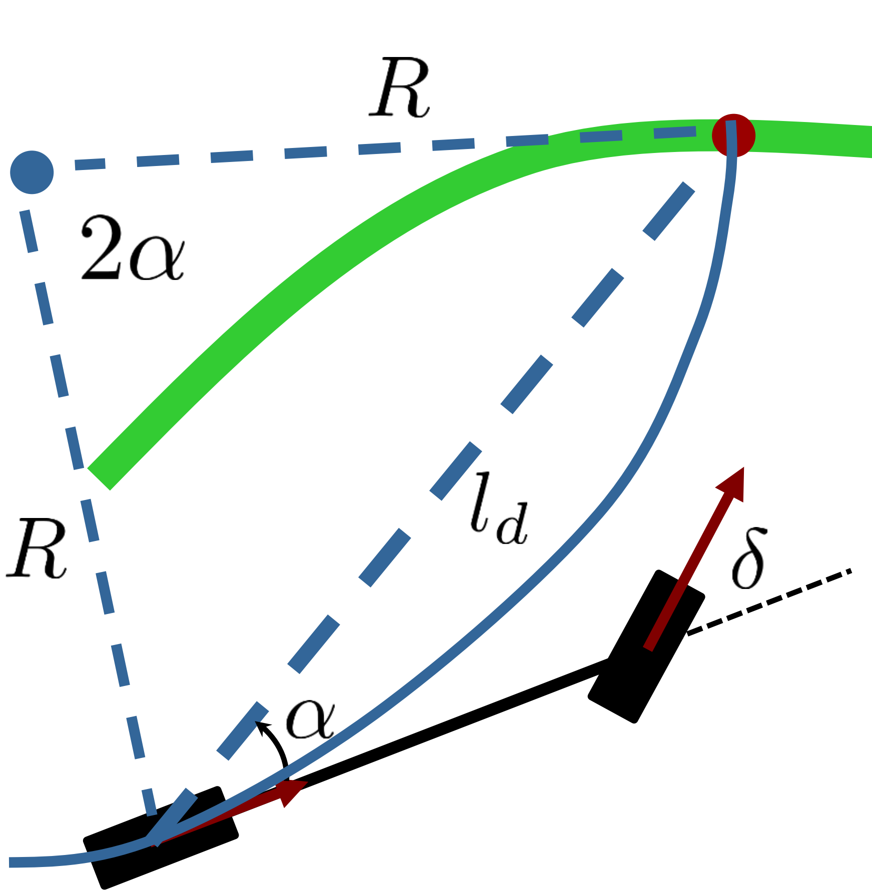
$$\delta = \tan^{-1}(\kappa L)$$

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

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

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

Pure Pursuit Controller



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

$$l_d = K_{dd}v$$

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

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:

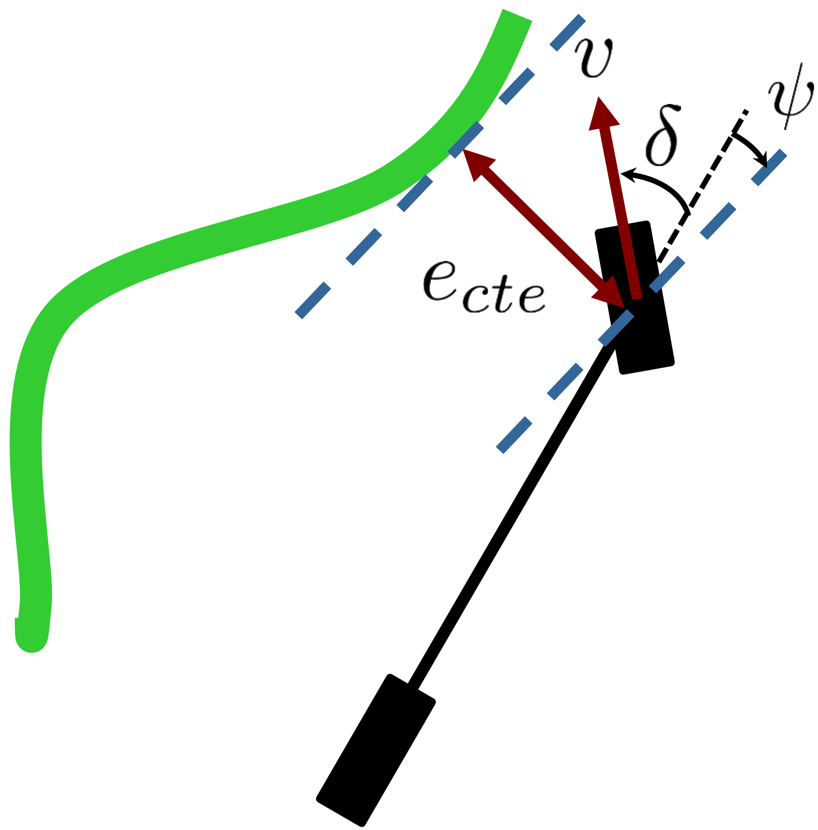
$$\delta = \psi$$

- Cross-track error:

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

- Steering limit:

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

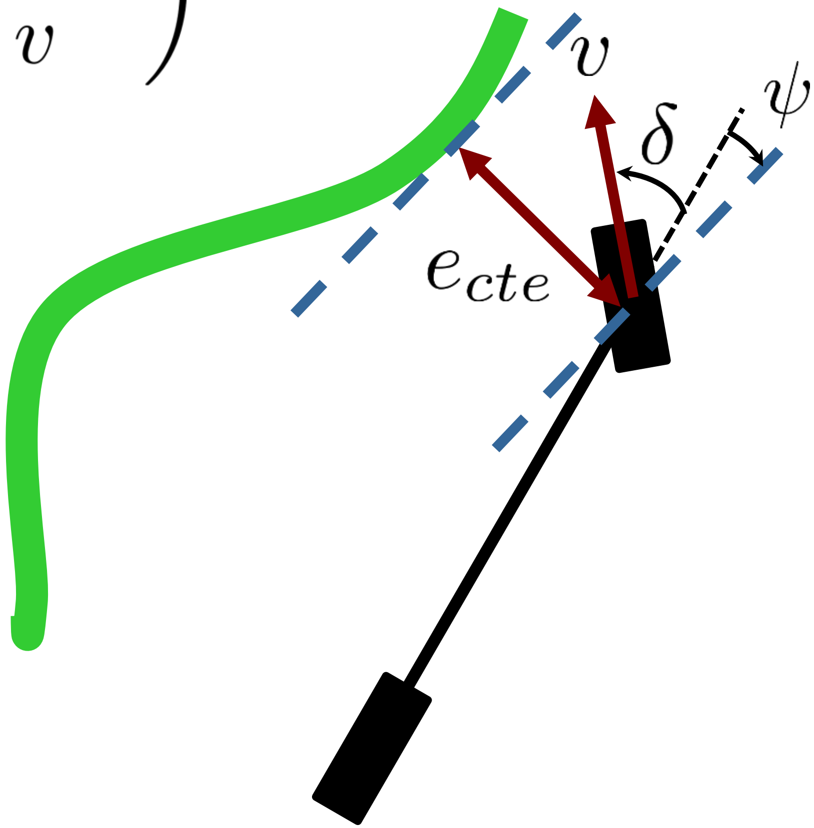


Stanley Controller

- Combined control law:

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

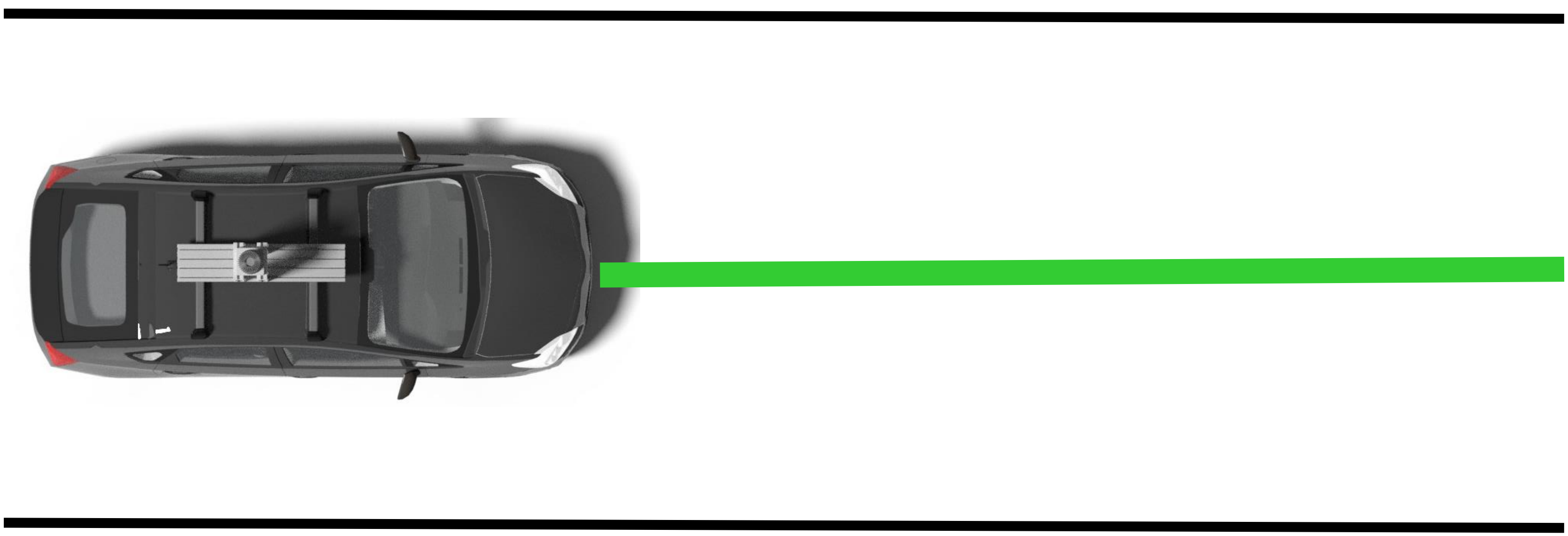
$$\delta \in [\delta_{min}, \delta_{max}]$$



Model Predictive Control

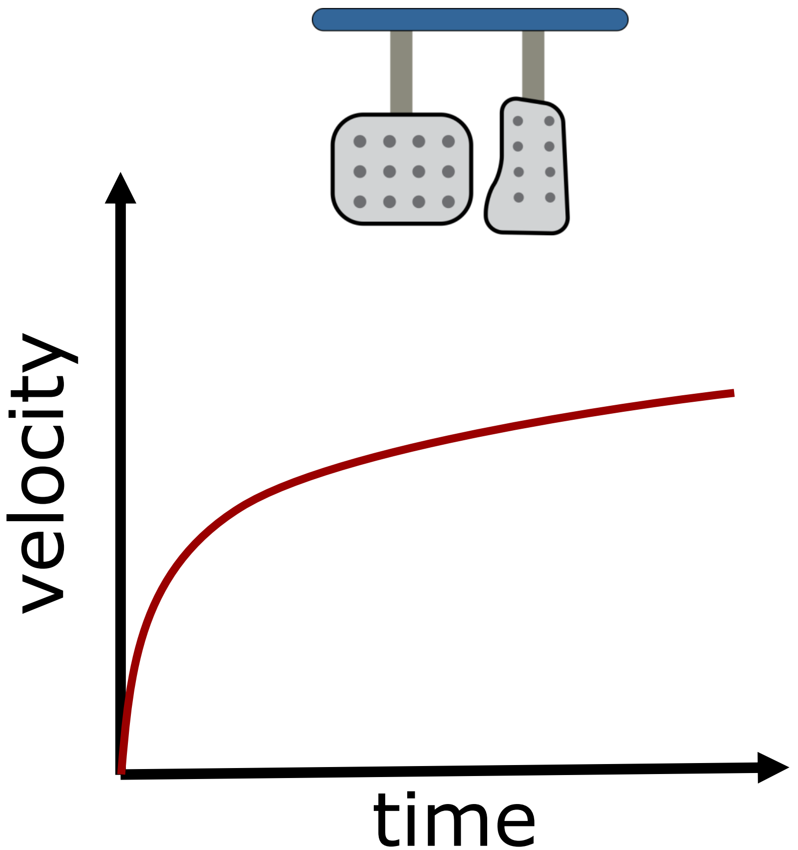
Model Predictive Control (MPC)

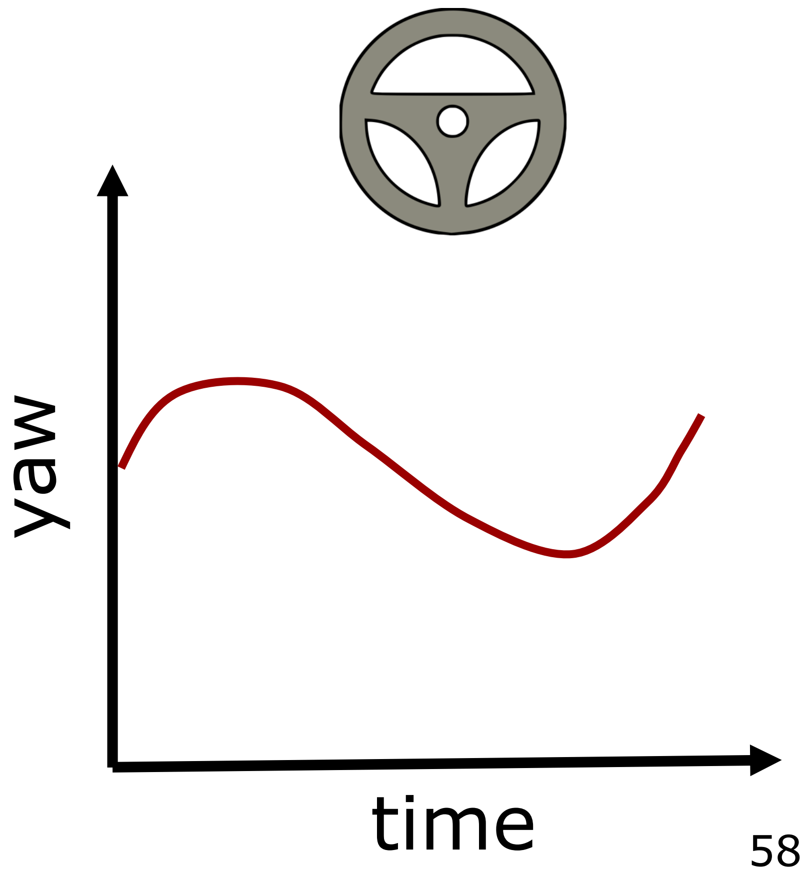
- Uses a model of the system to make future predictions for the system



Model Predictive Control (MPC)

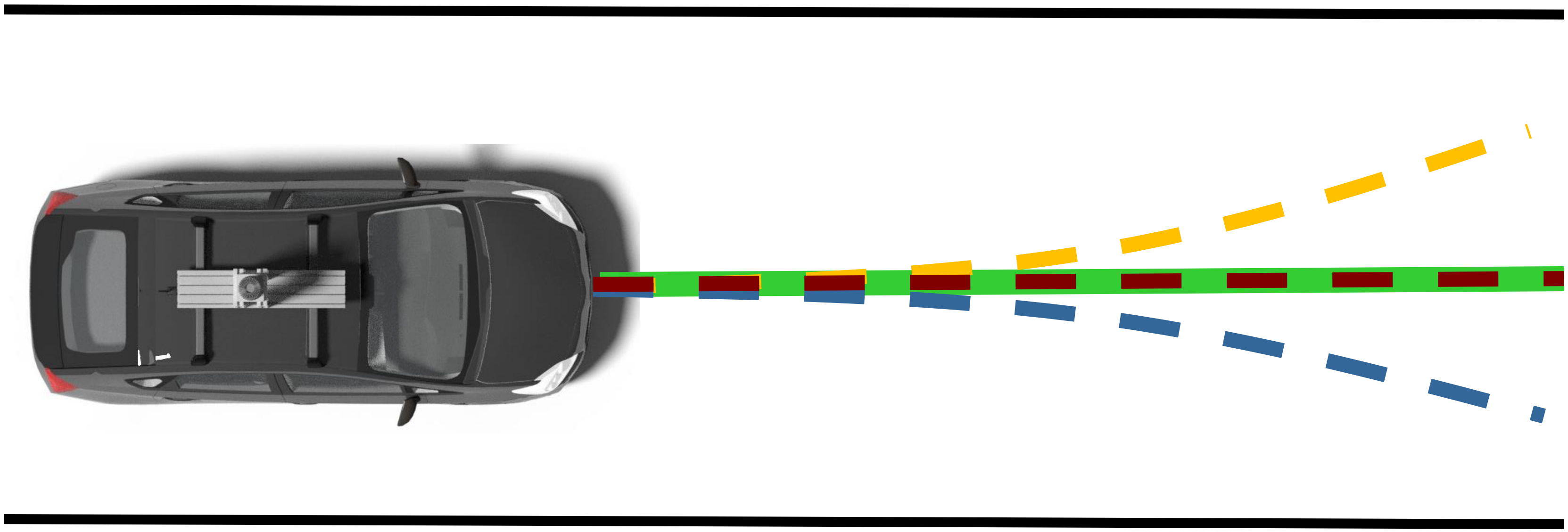
- Uses a model of the system to make future predictions for the system





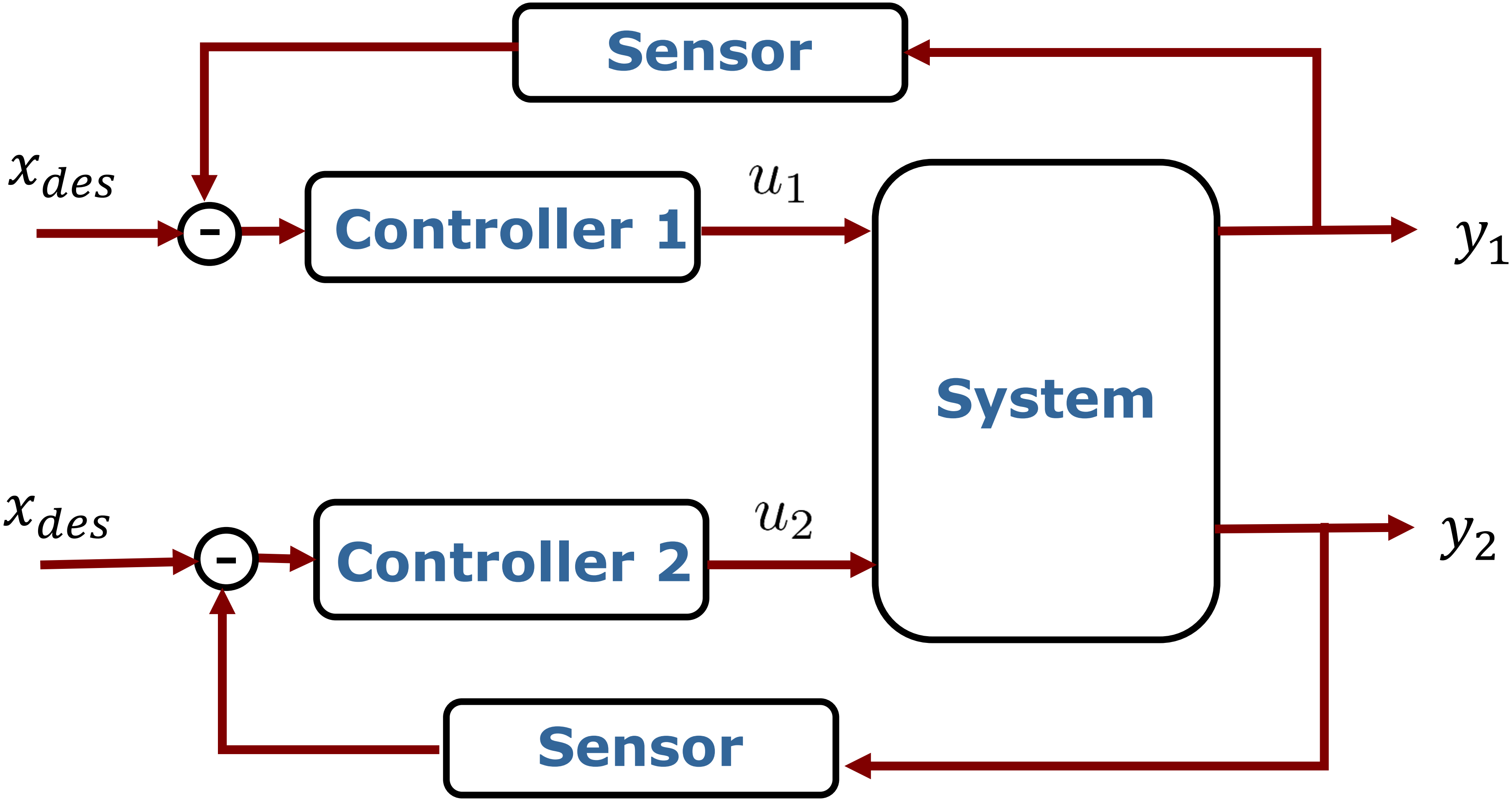
Model Predictive Control (MPC)

- Uses a model of the system to make future predictions for the system



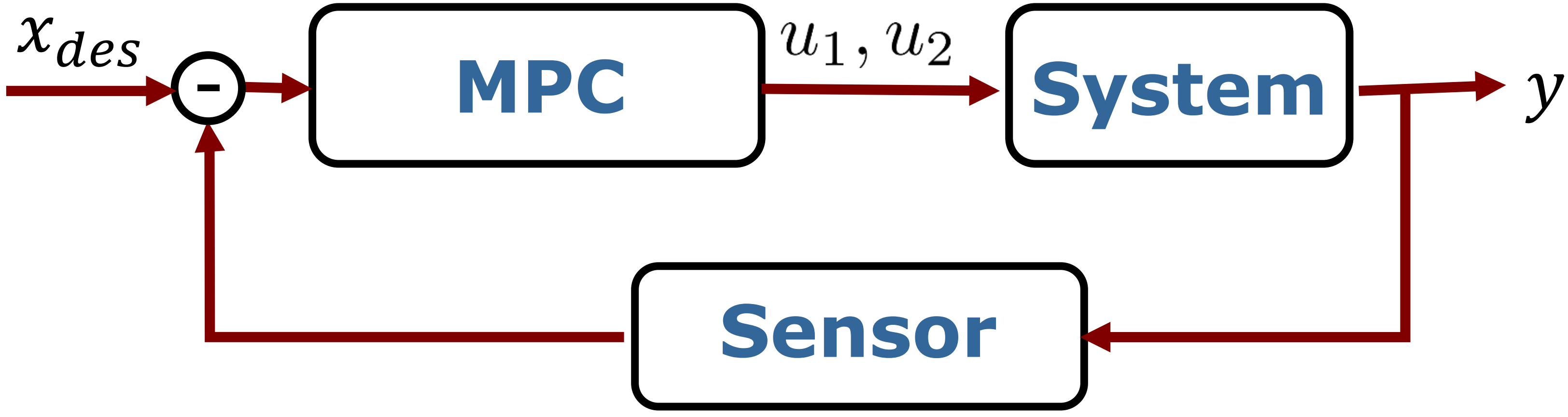
Why MPC?

- Handle multiple inputs/outputs jointly



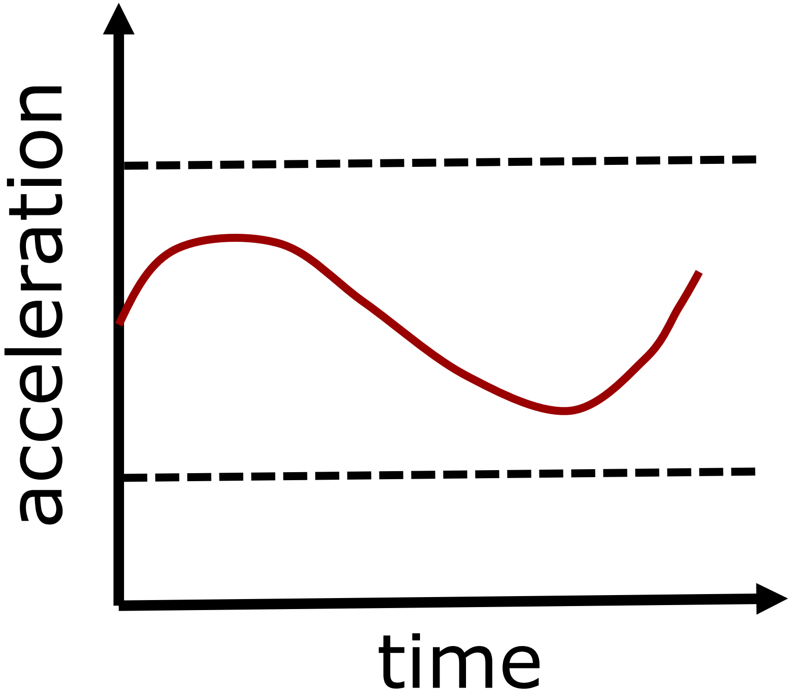
Why MPC?

- Handle multiple inputs/outputs jointly



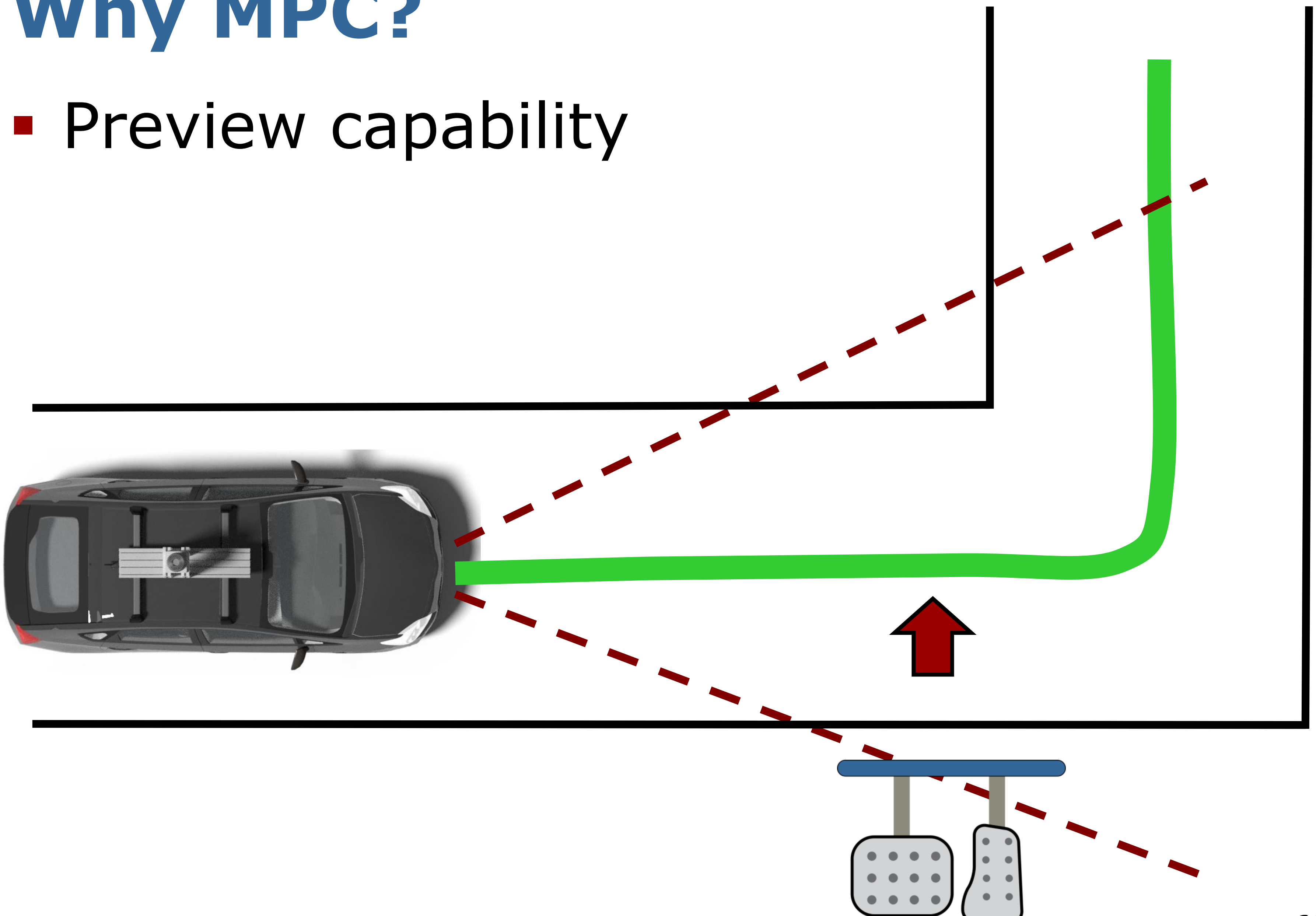
Why MPC?

- Handle Constraints



Why MPC?

- Preview capability



Control as Optimization Problem

- Find the controller that minimizes some cost function.
- How to define this cost function?
- What would be a **good cost function**?
    - Minimize the error to reference?
    - Minimize the controls necessary?
    - Combination of both?

Model Predictive Control

- Discrete-time **linear/non-linear** system

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

- **Quadratic** cost function

$$J = \sum (e_t^T Q e_t + u_t^T R u_t)$$

$$e_t = x_{des} - x_t$$

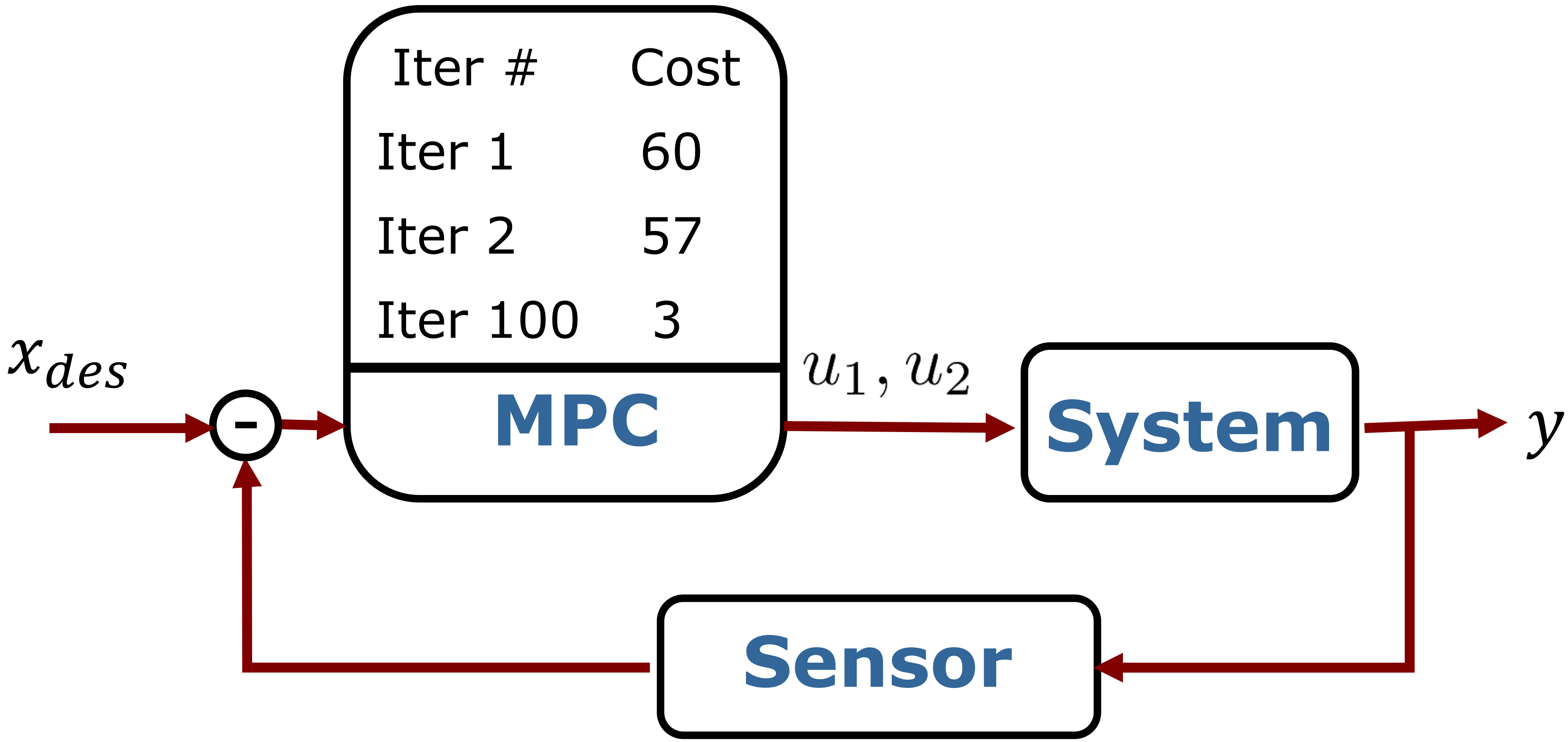
- **Goal** : Finds the control with the lowest cost.

Model Predictive Control

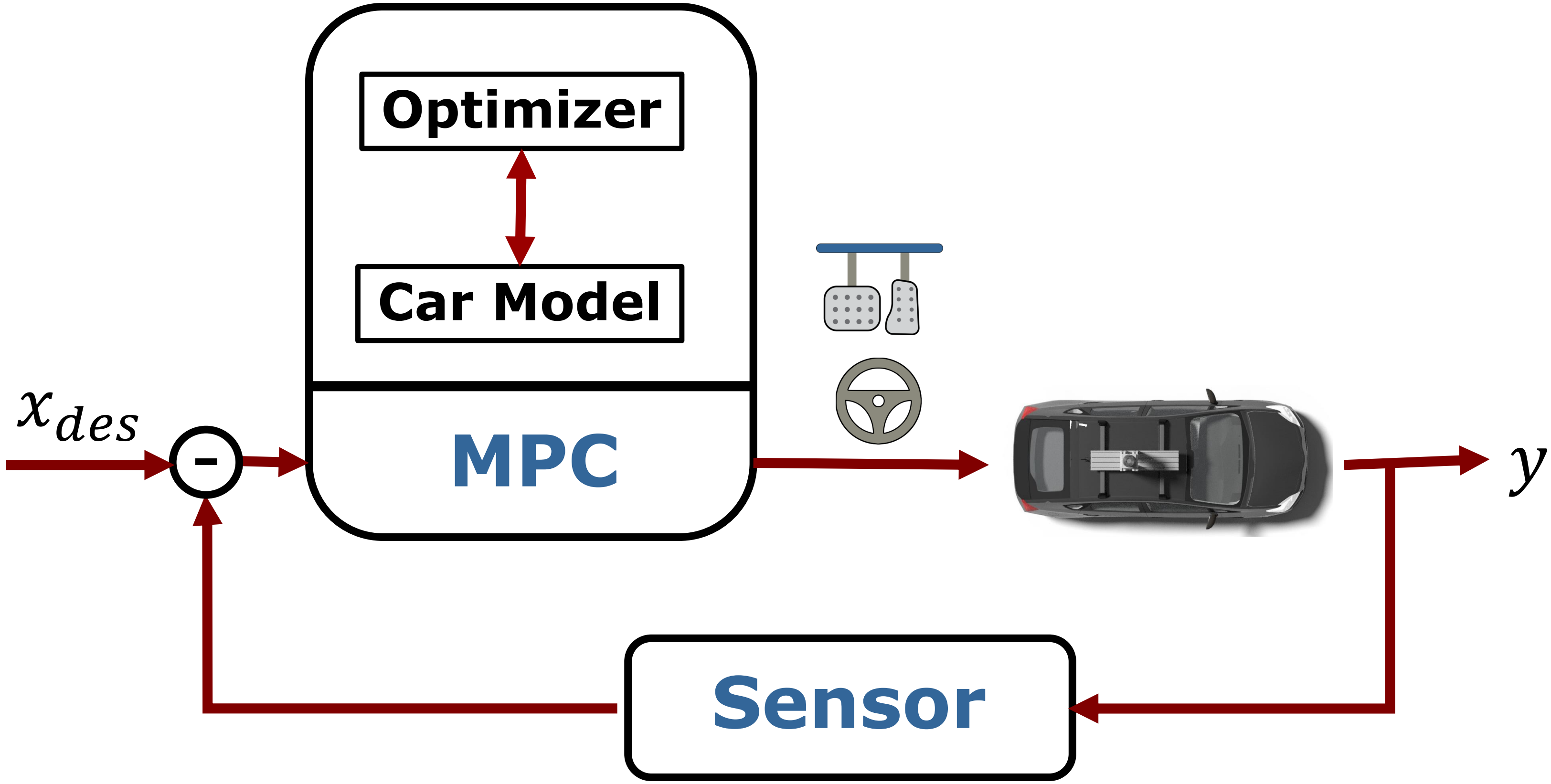
- **Pros:**
  - Cost matrix has an intuitive meaning
- **Cons:**
  - Typically no closed form solution
- Must be solved numerically
- Feasible only for small planning horizon

Requirements for MPC

- Fast processing power
- Large memory

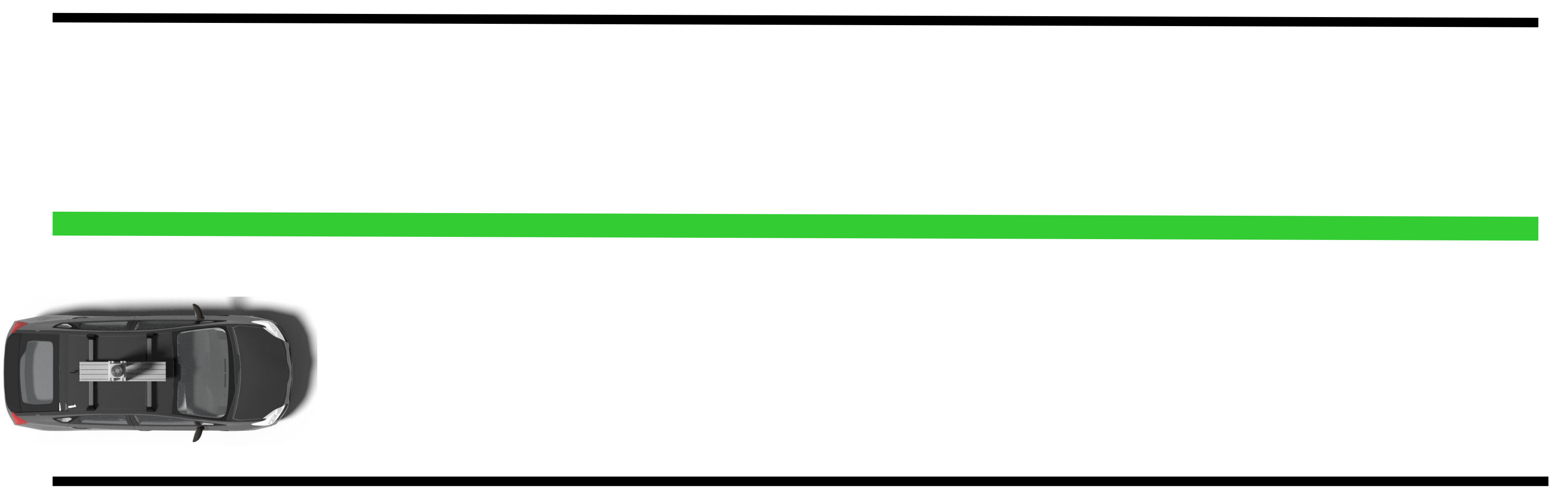


MPC for Self-Driving Cars

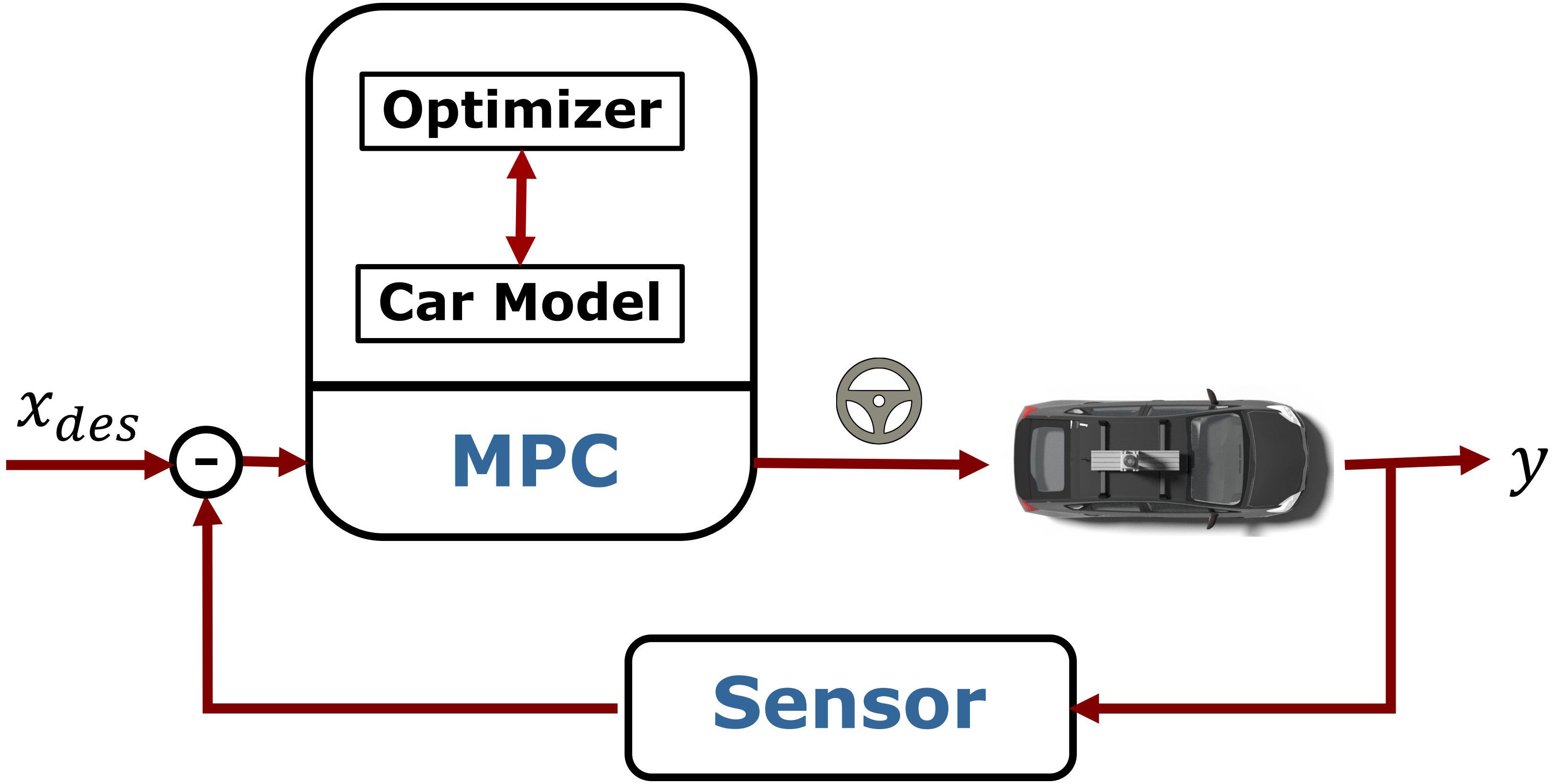


MPC for Self-Driving Car

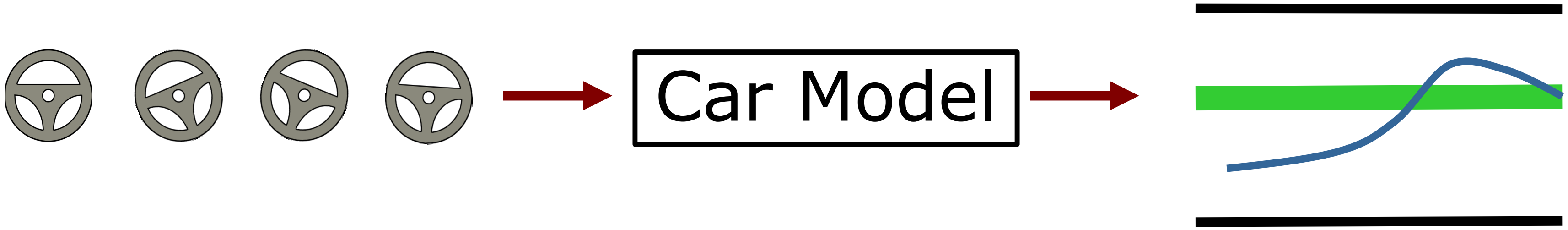
- What should be the control such that the car follows the center line?



MPC for Self-Driving Cars

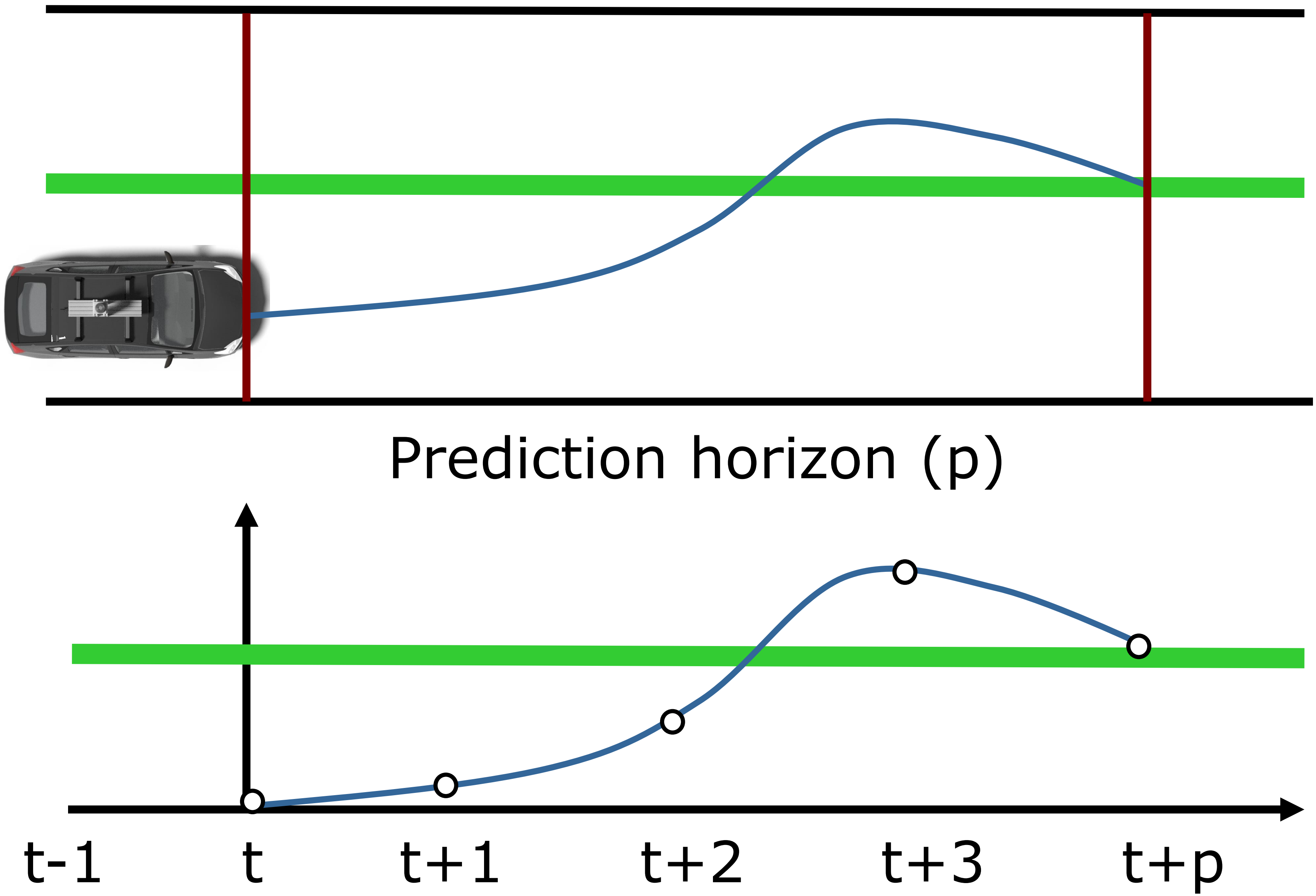


MPC for Self-Driving Cars

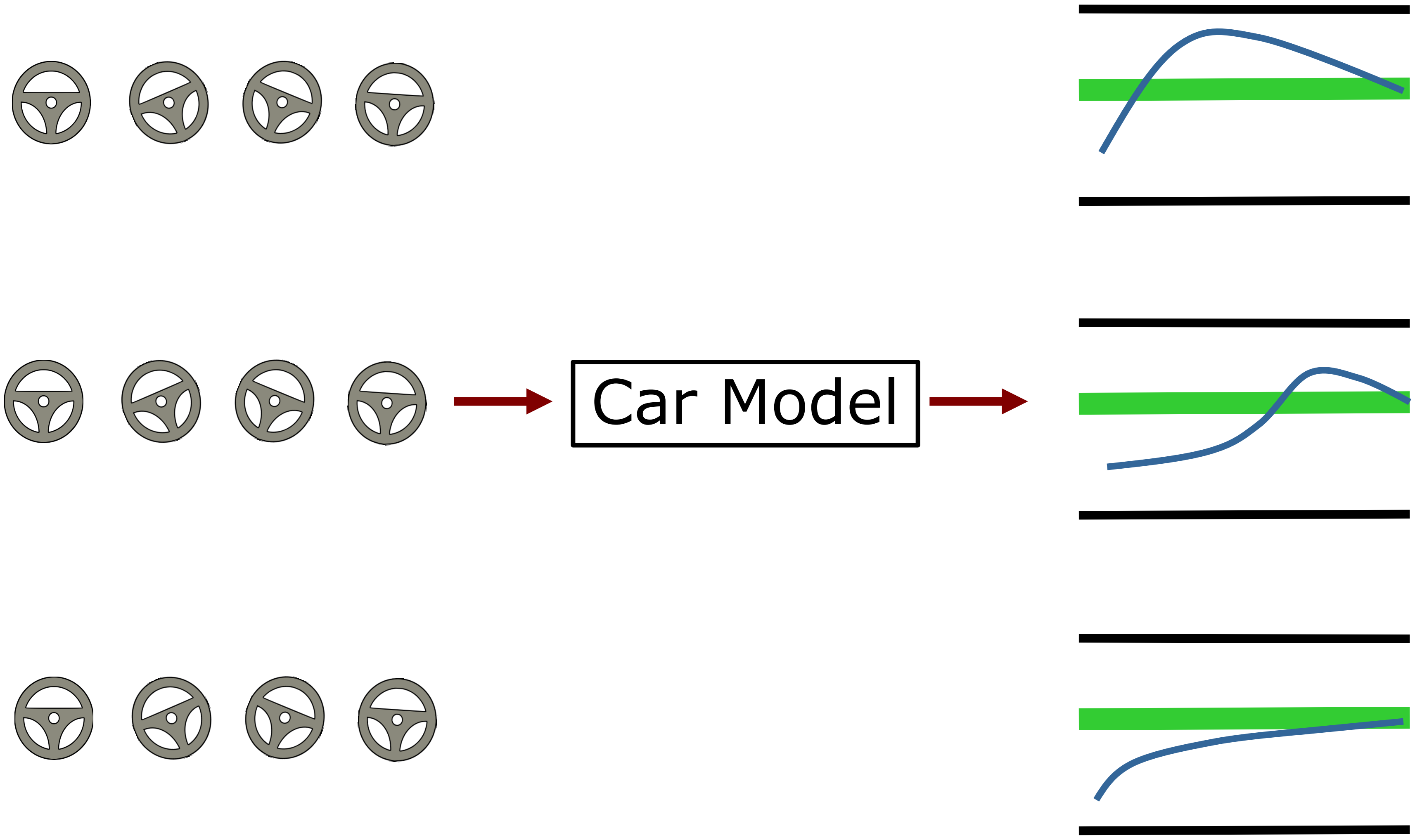


Simulation/Prediction

MPC for Self-Driving Cars

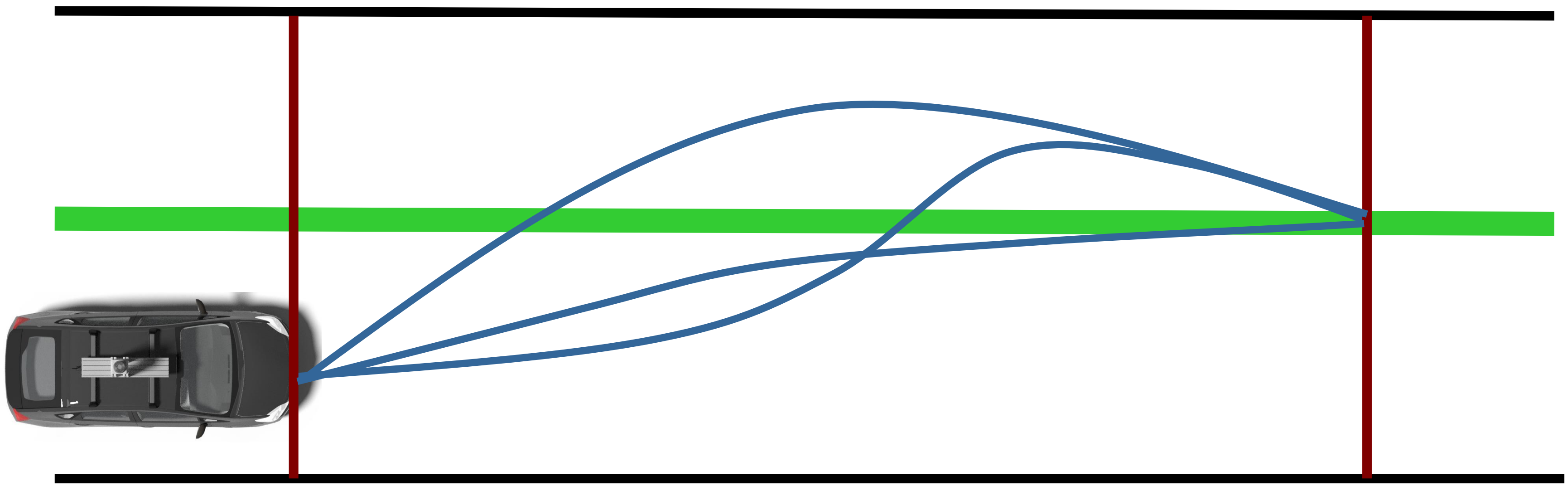


MPC for Self-Driving Cars

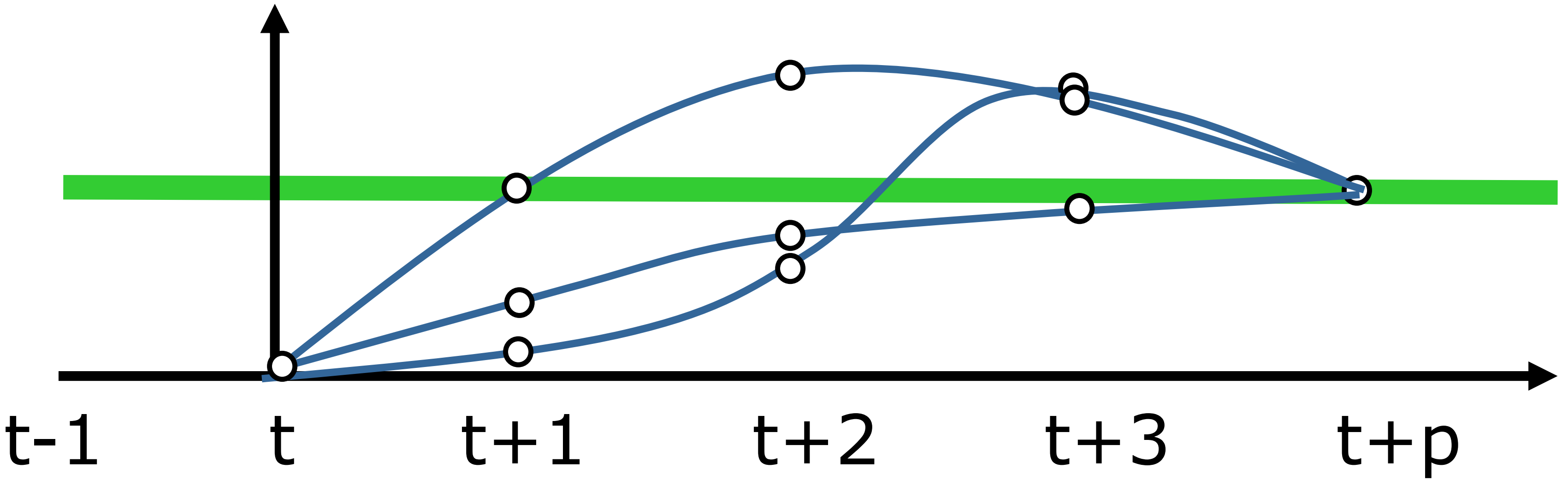


Simulation/Prediction

MPC for Self-Driving Cars



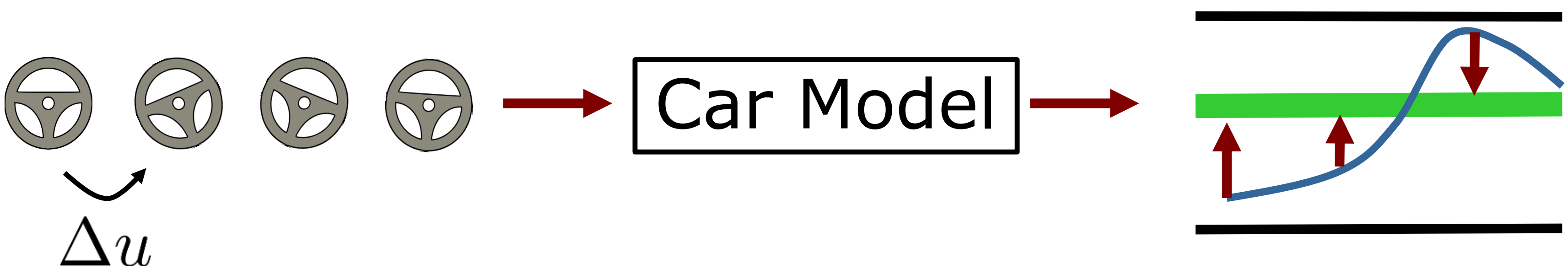
Prediction horizon (p)



MPC for Self-Driving Cars

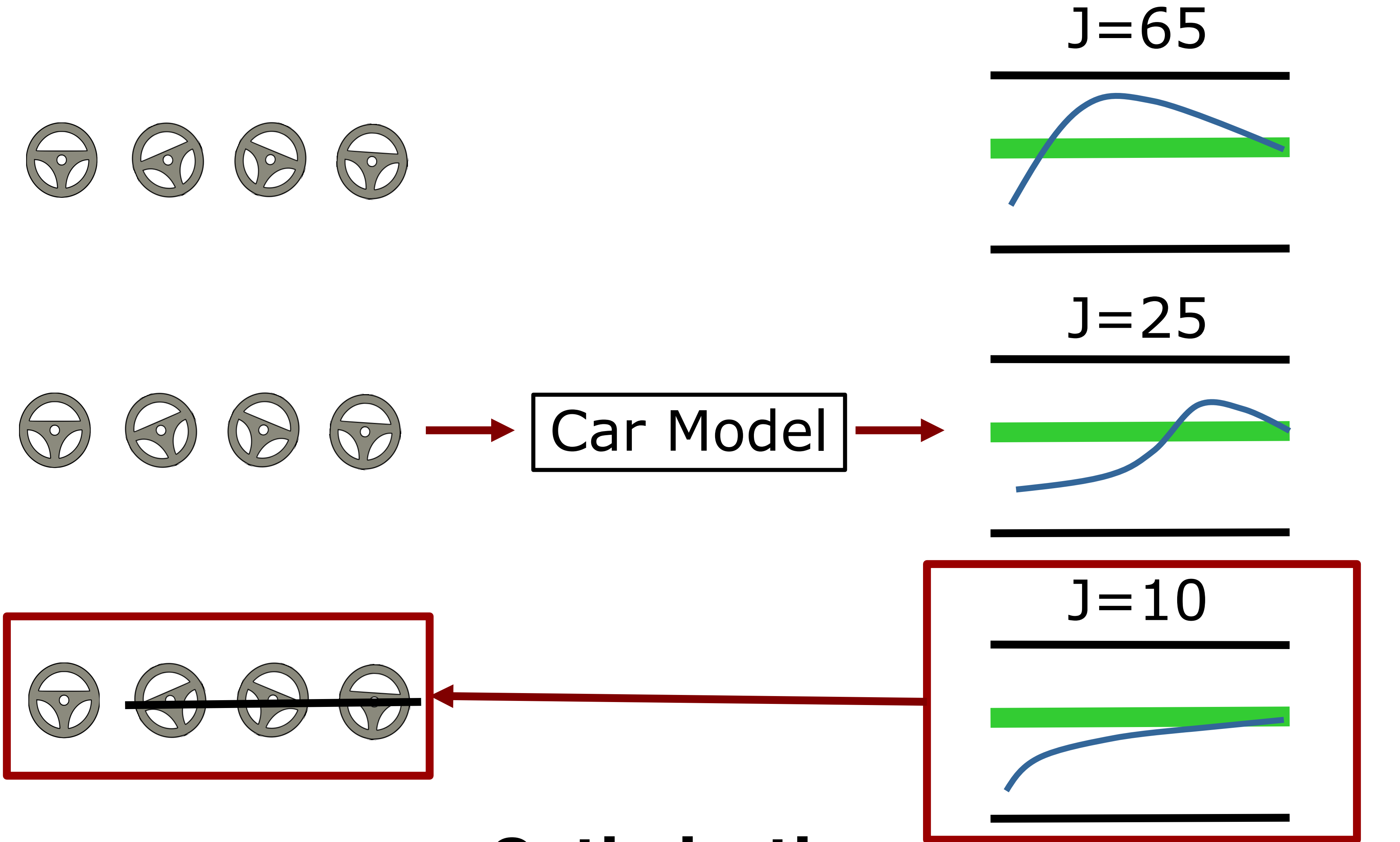
$$J = \sum_{i=1}^{p} e_t^T Q e_t + \sum_{i=0}^{p-1} \Delta u_t^T R \Delta u_t$$

s.t. All constraints are satisfied



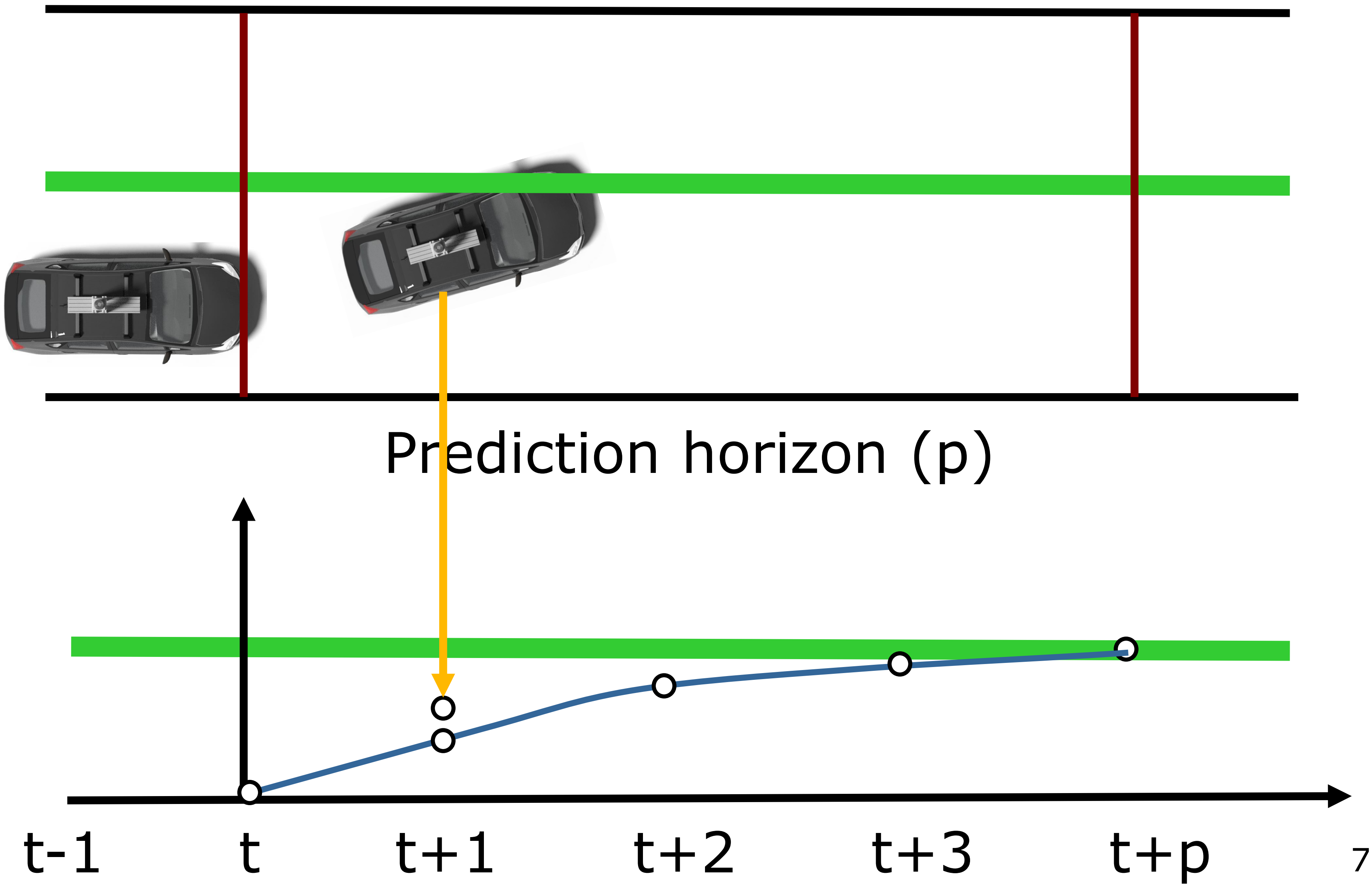
Optimization

MPC for Self-Driving Cars

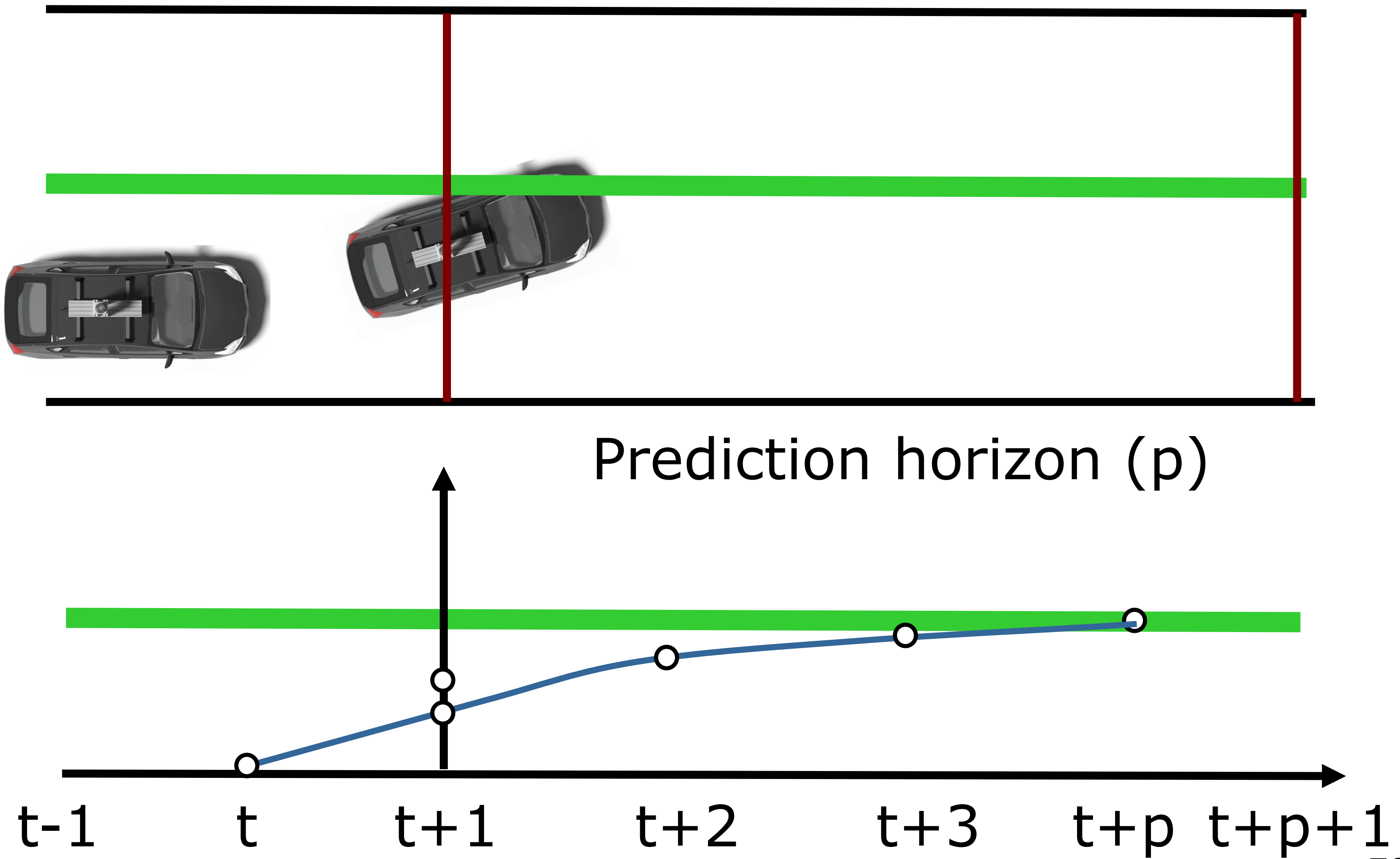


Optimization

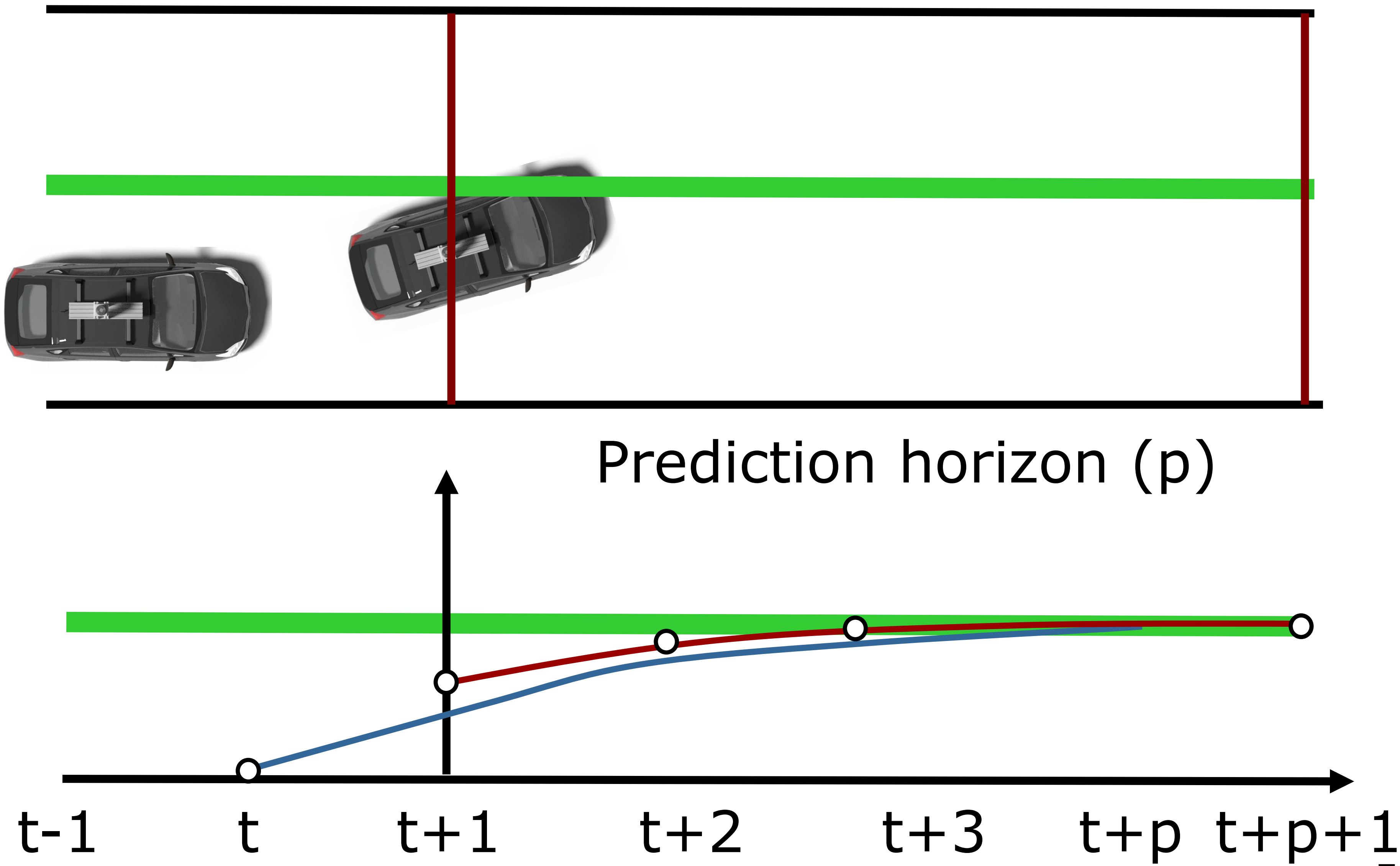
MPC for Self-Driving Cars



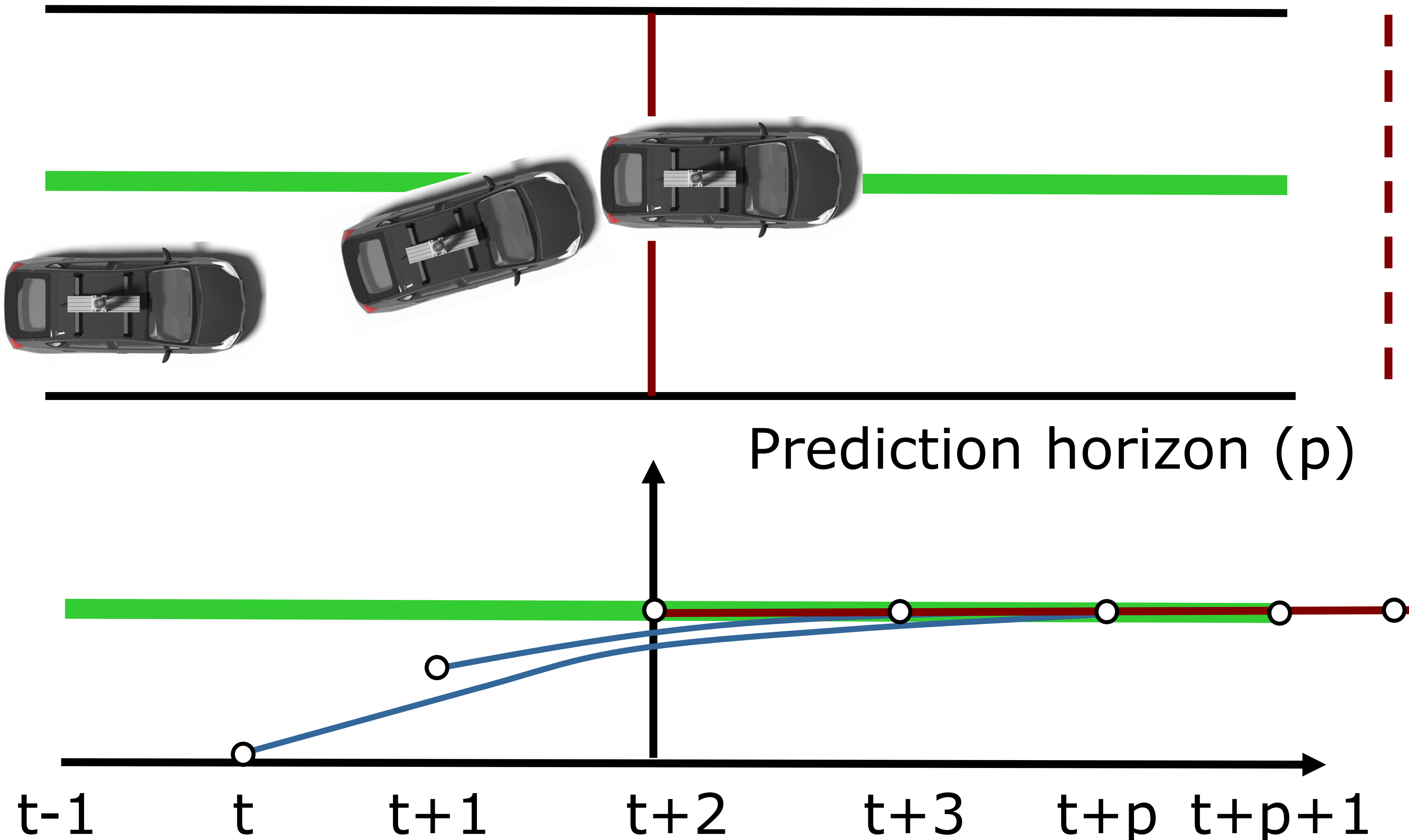
MPC for Self-Driving Cars



MPC for Self-Driving Cars

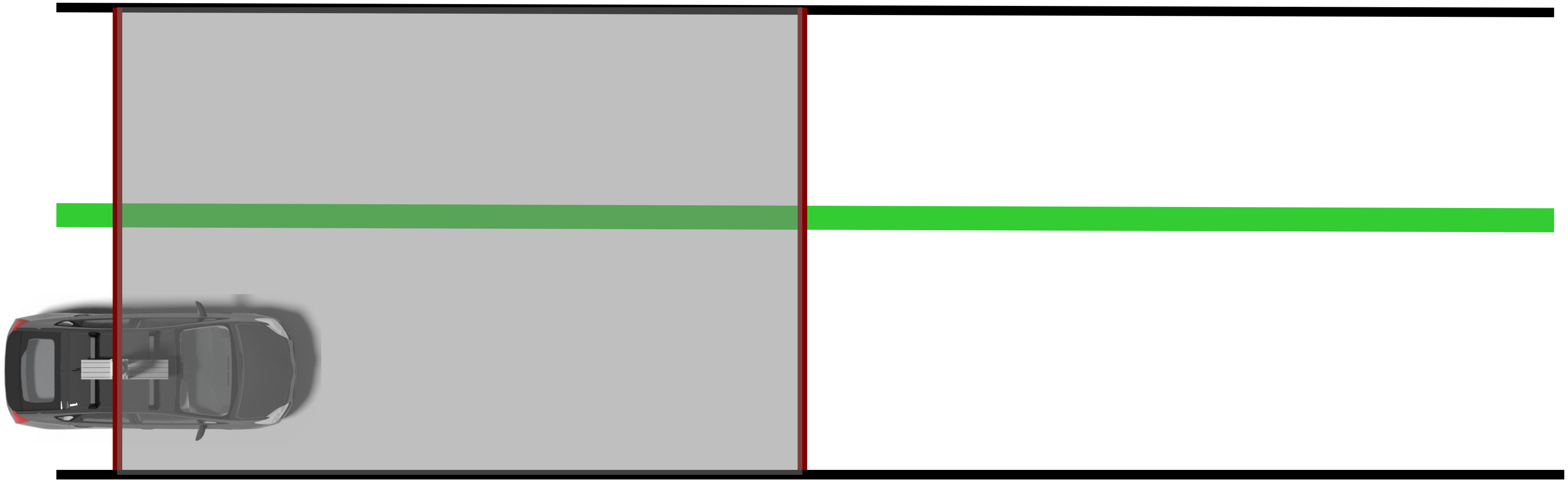


MPC for Self-Driving Cars

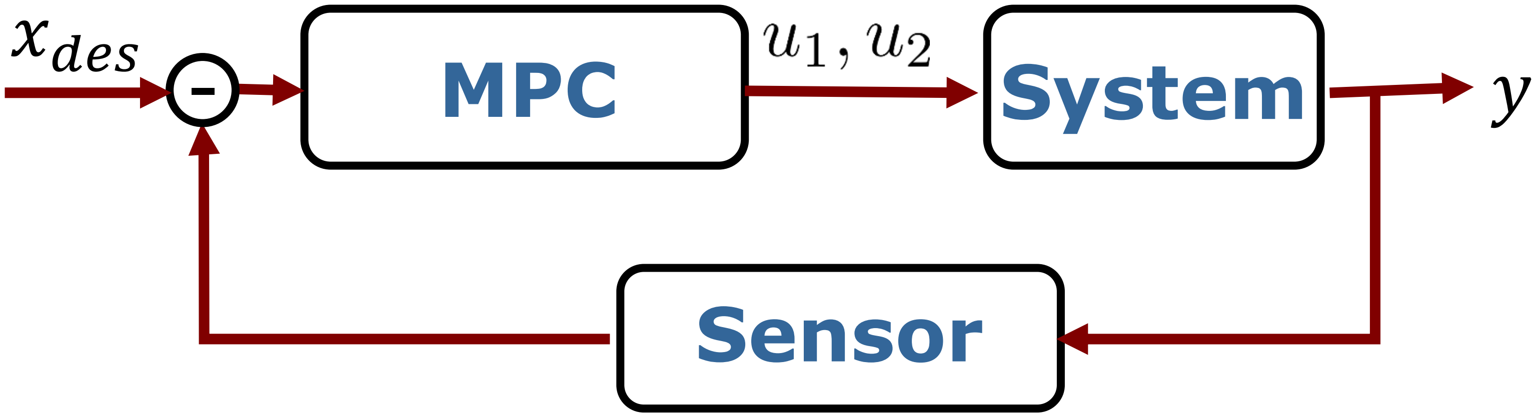


MPC for Self-Driving Cars

- Receding Horizon Control



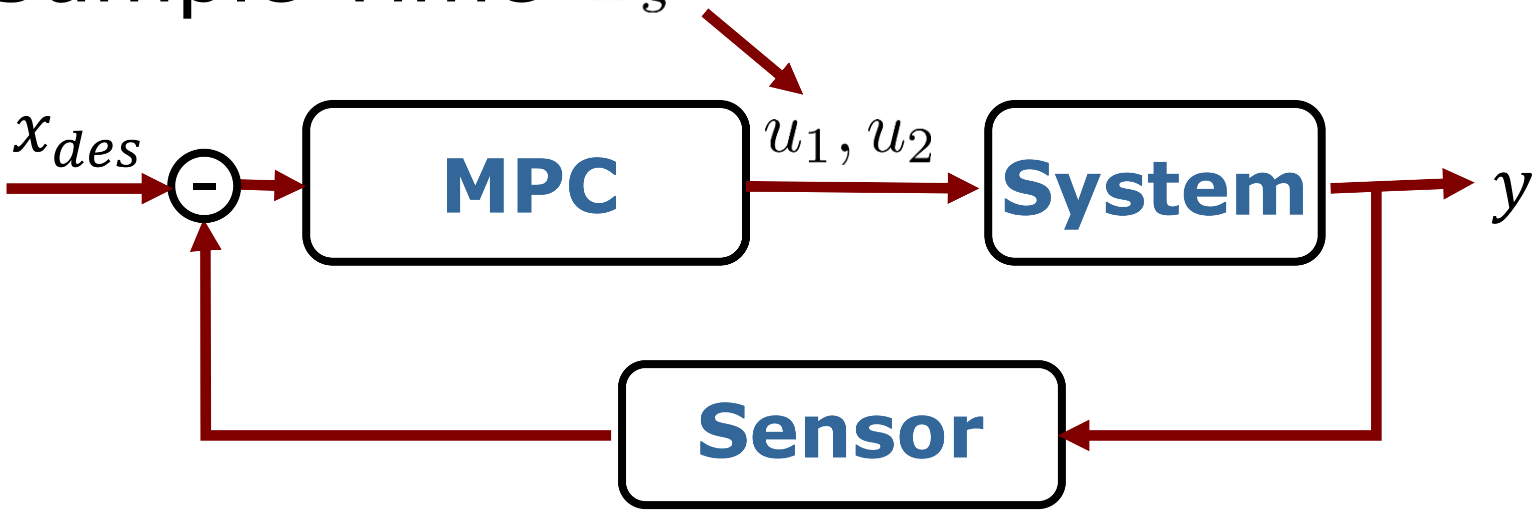
MPC Design Parameters

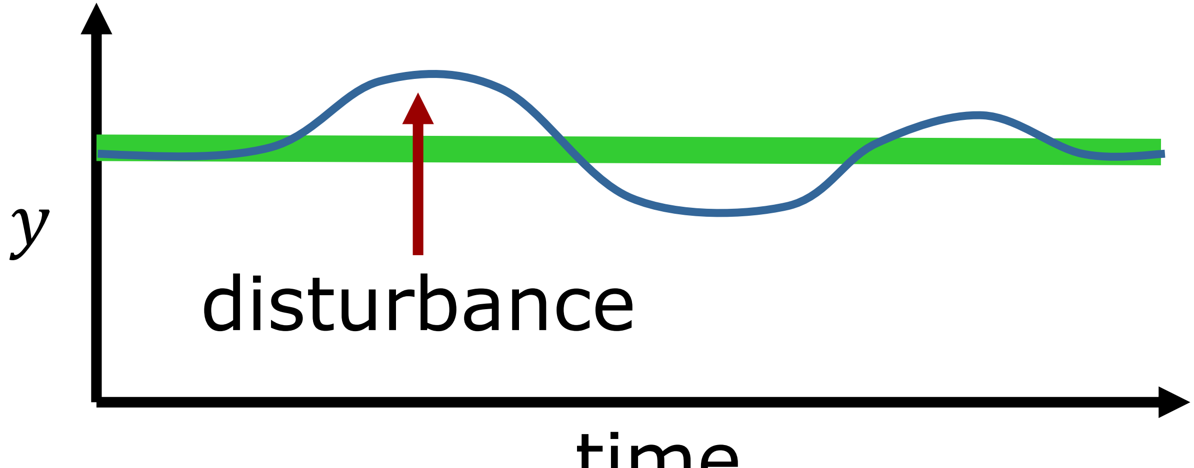


- Prediction Horizon
- Control Horizon
- Sample Time
- Constraints
- Weights

MPC Design Parameters

- Sample Time T_s

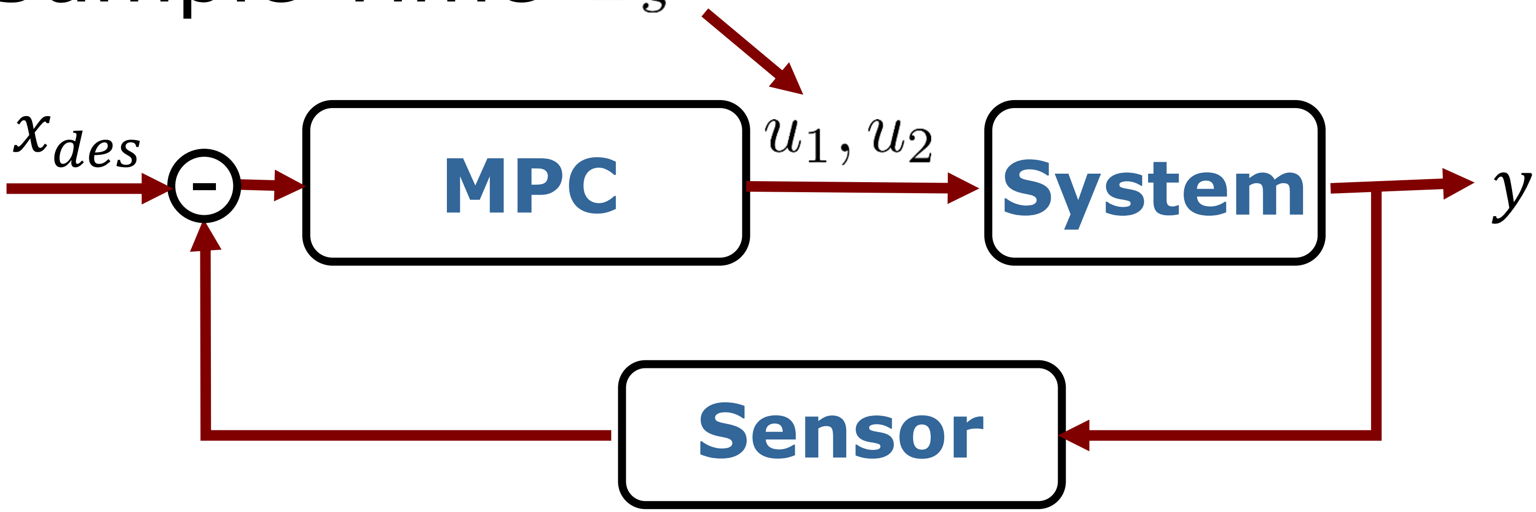


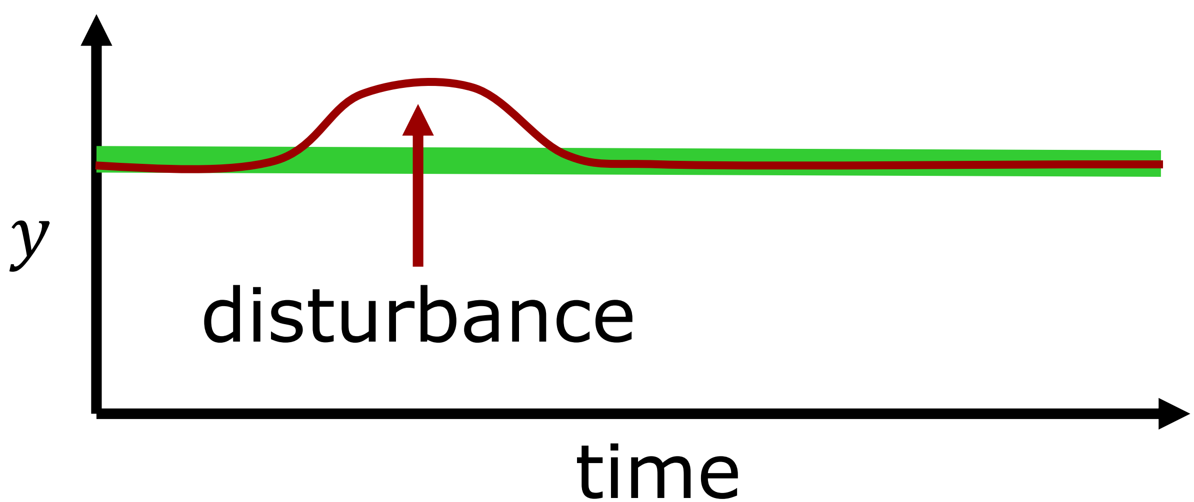


Low T_s

MPC Design Parameters

- Sample Time T_s

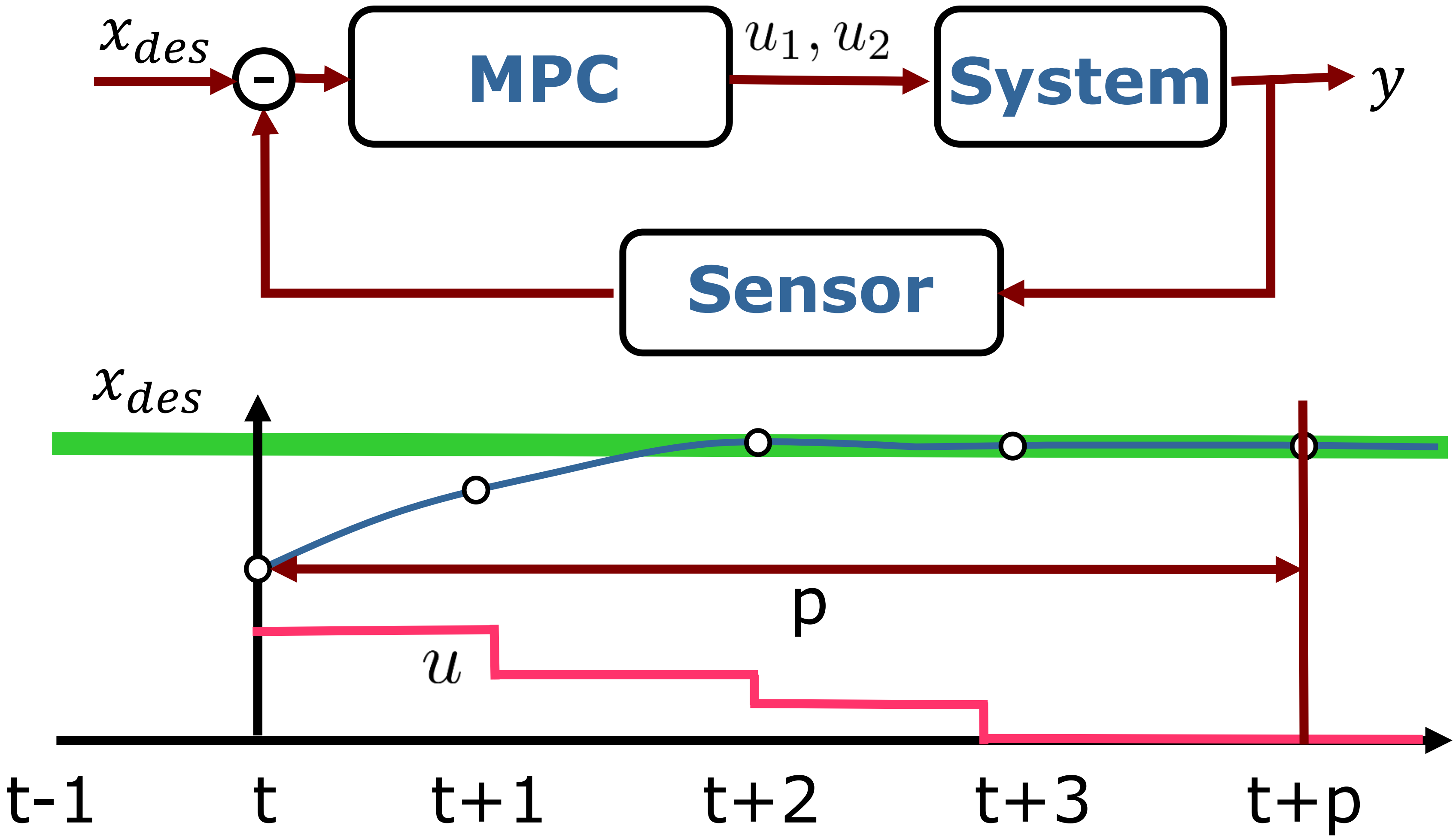




High T_s

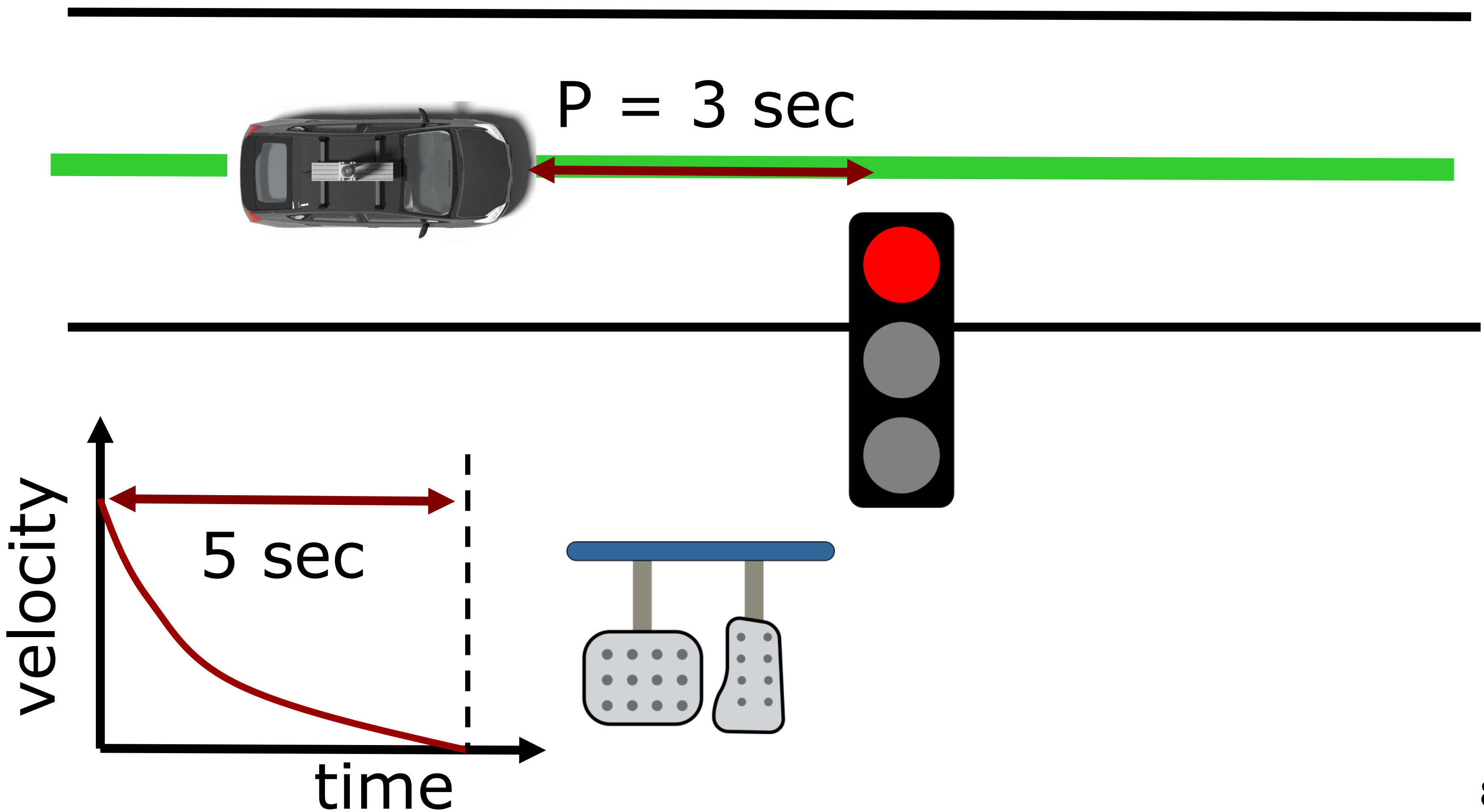
MPC Design Parameters

- Prediction Horizon



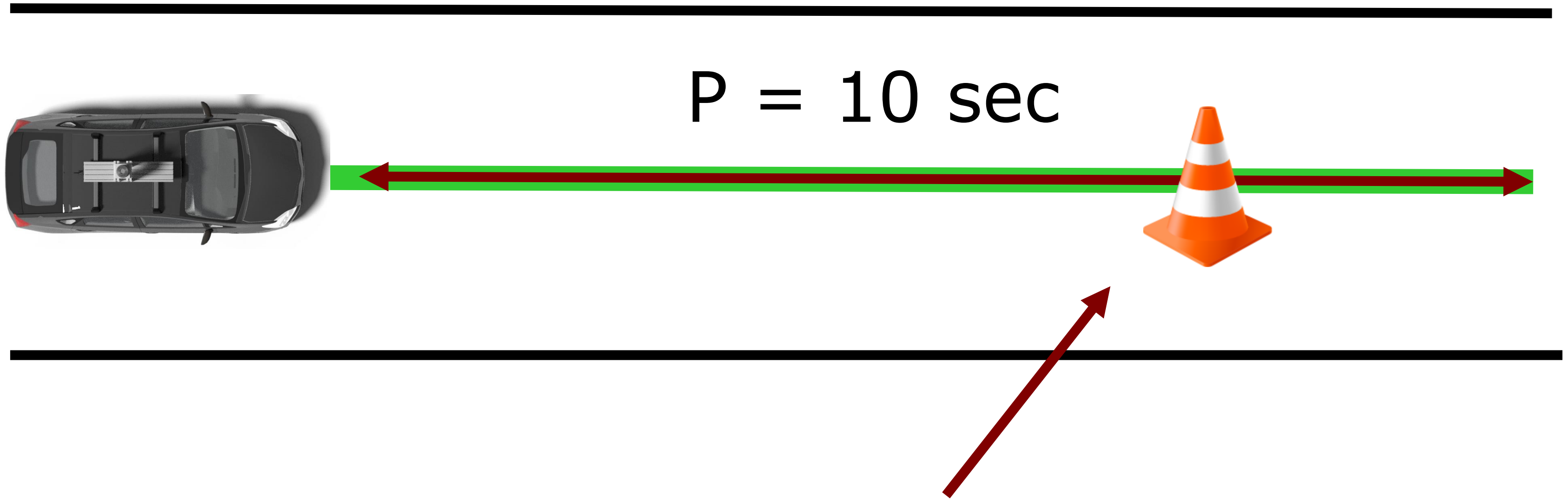
MPC Design Parameters

- Prediction Horizon



MPC Design Parameters

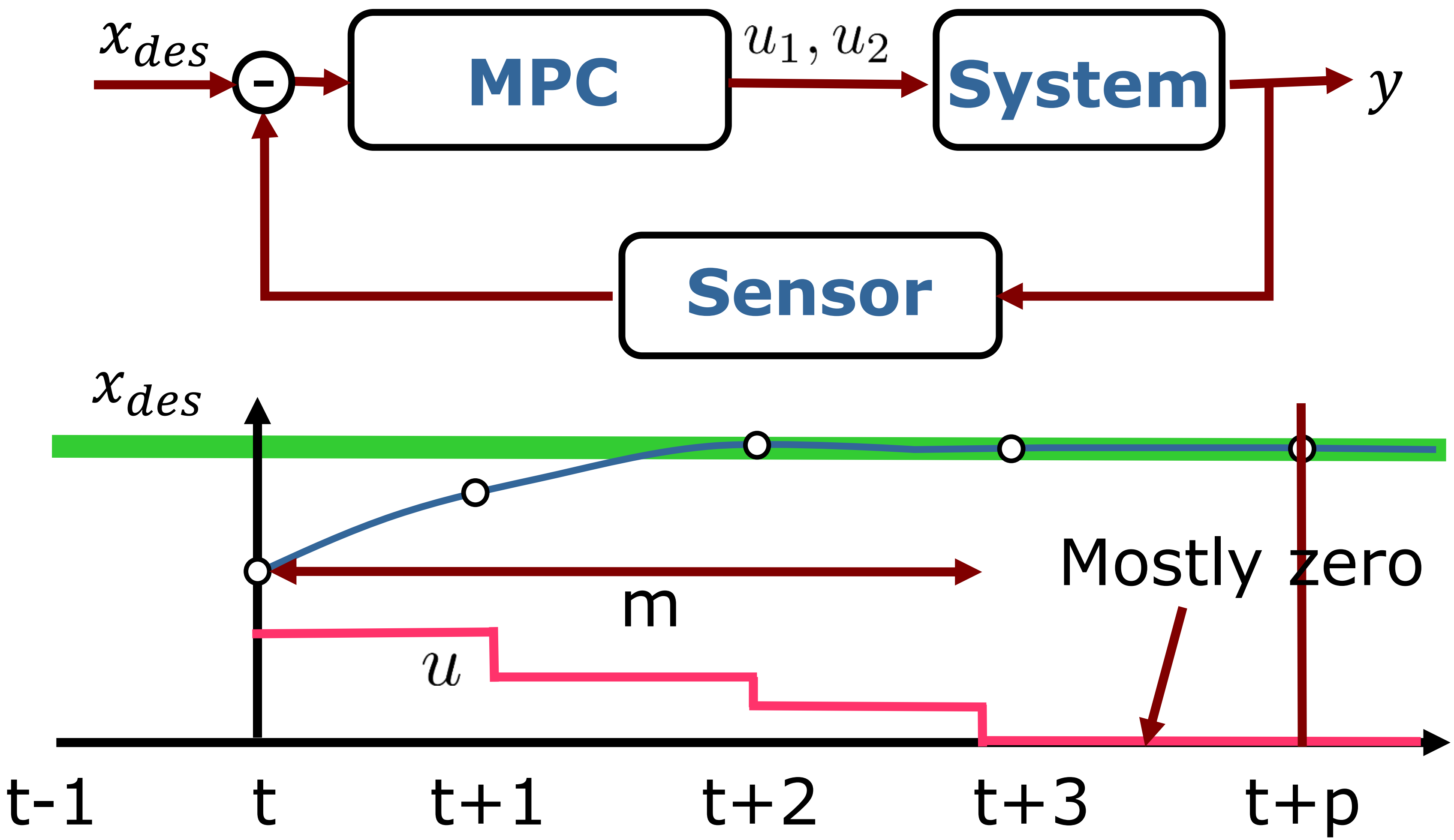
- Prediction Horizon



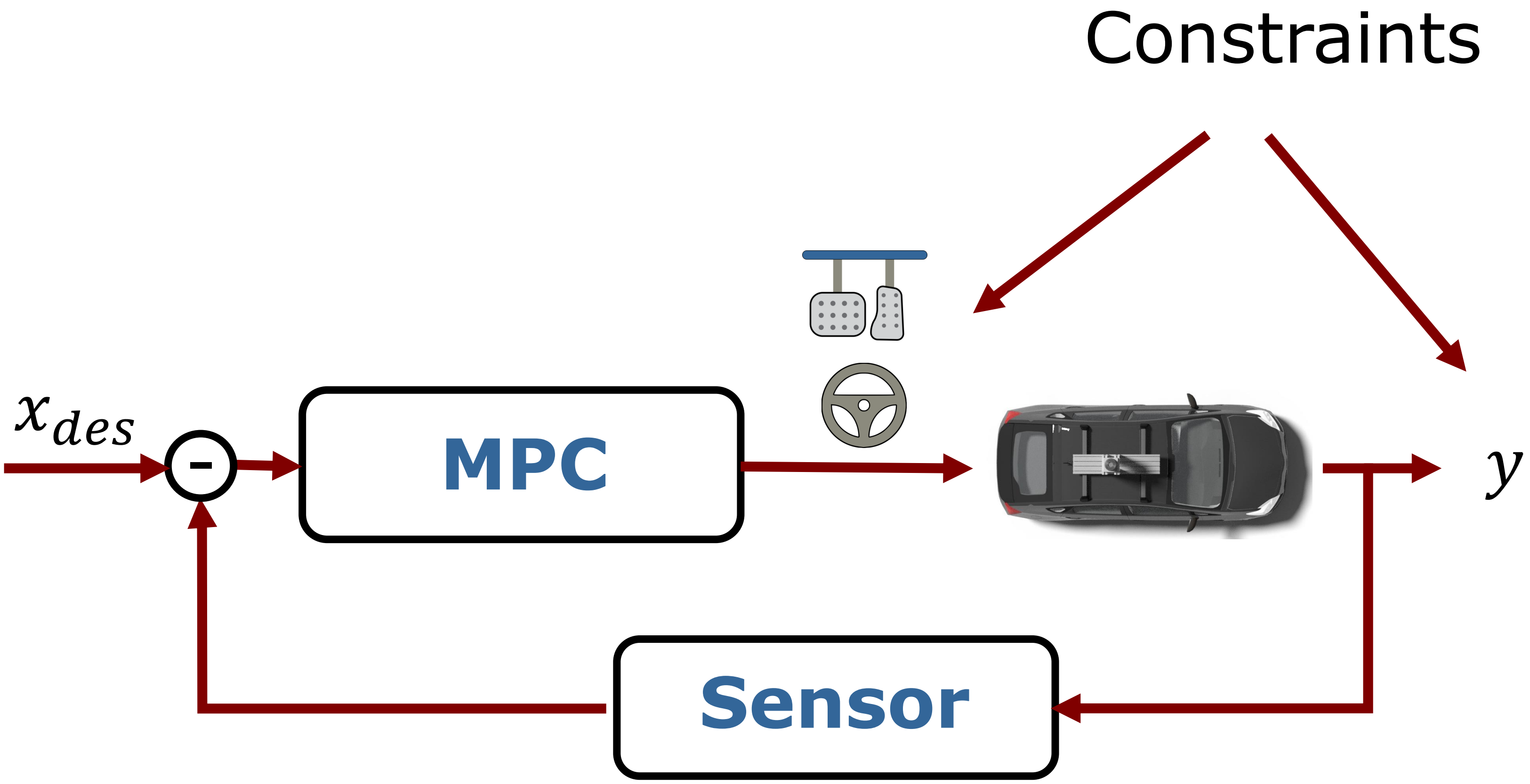
controls need to be recomputed

MPC Design Parameters

- Control Horizon



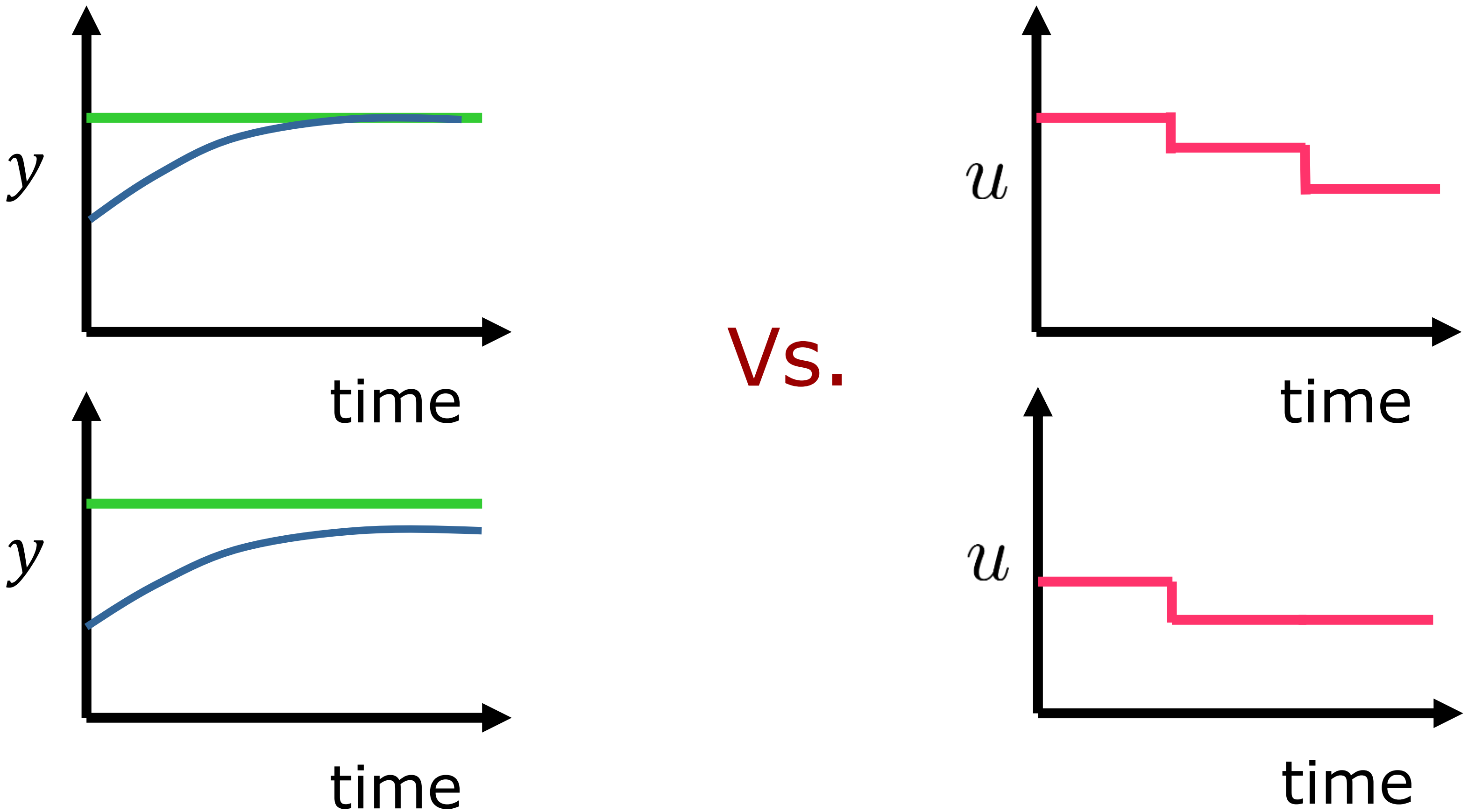
MPC Design Parameters



MPC Design Parameters

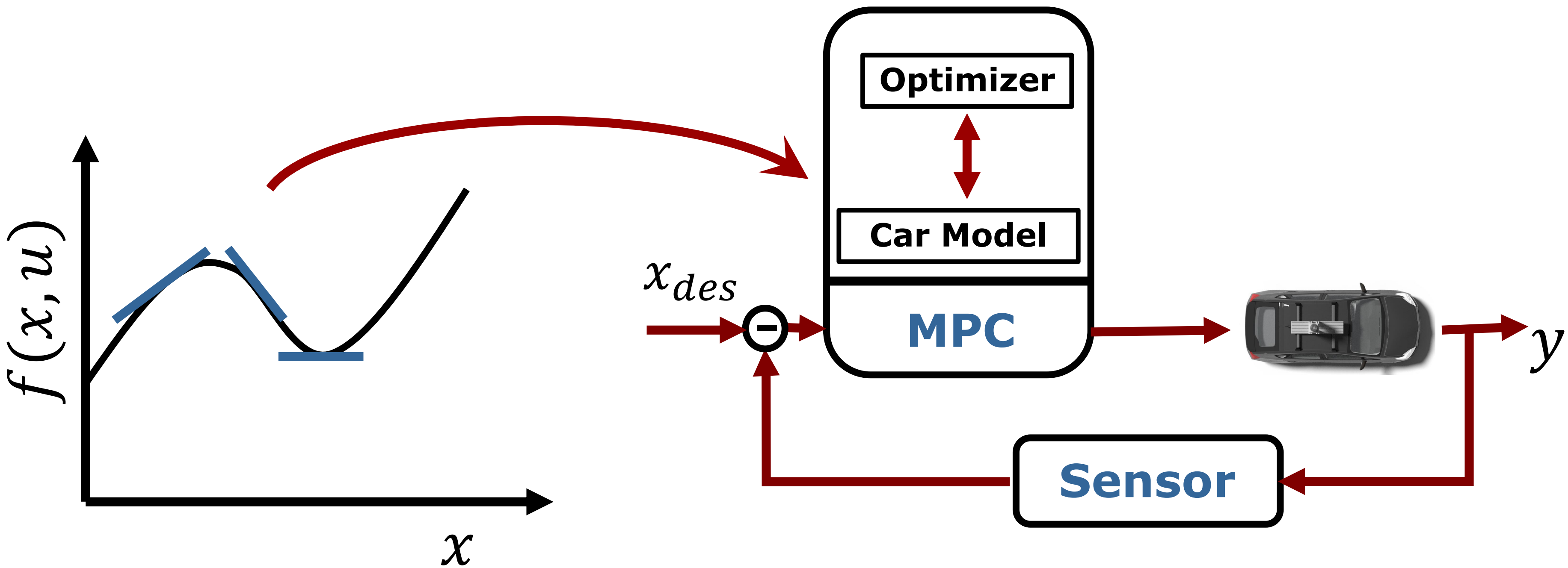
- Weights (Error vs Control)

$$J = \sum (e_t^T Q e_t + u_t^T R u_t)$$



Adaptive (Linearized) MPC

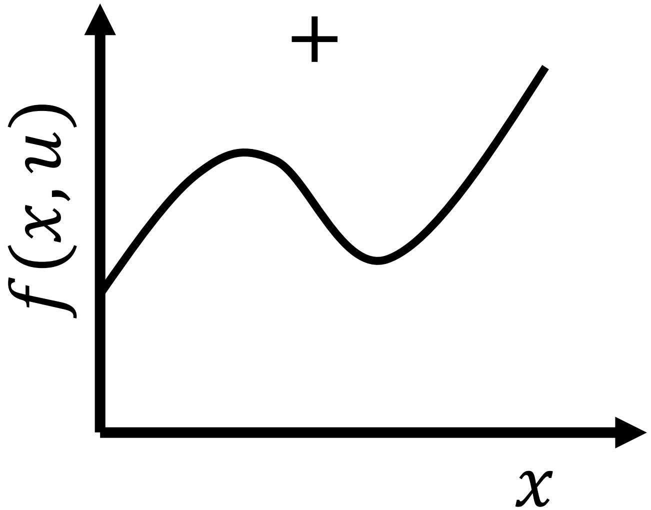
- What if the system has non-linear dynamics?

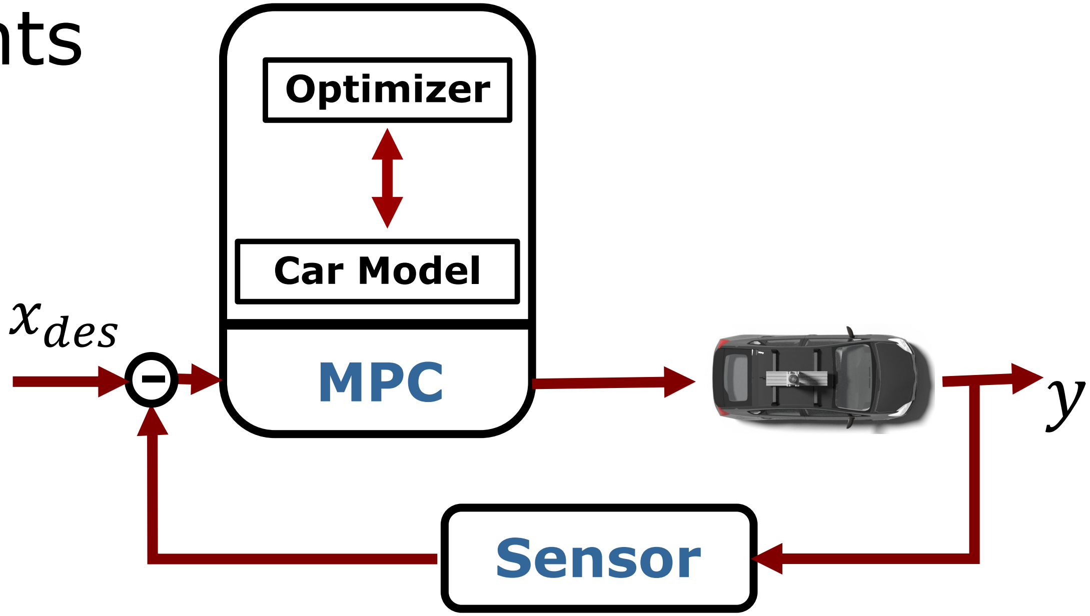


Non-linear MPC

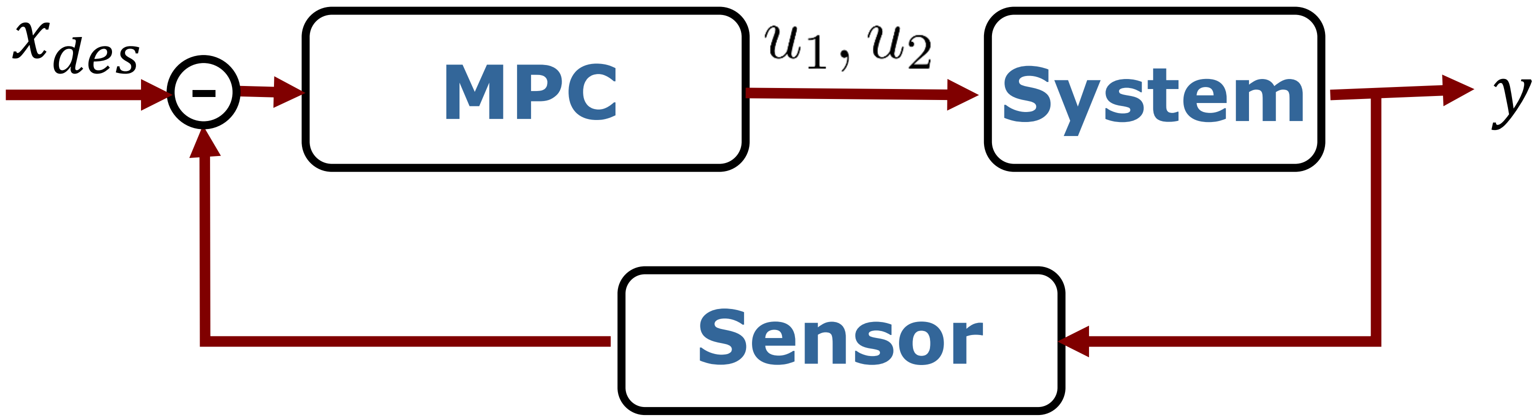
- What if the constraints are also non-linear?

Non-linear constraints

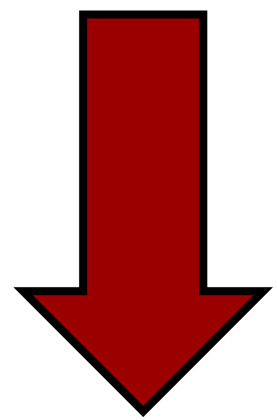




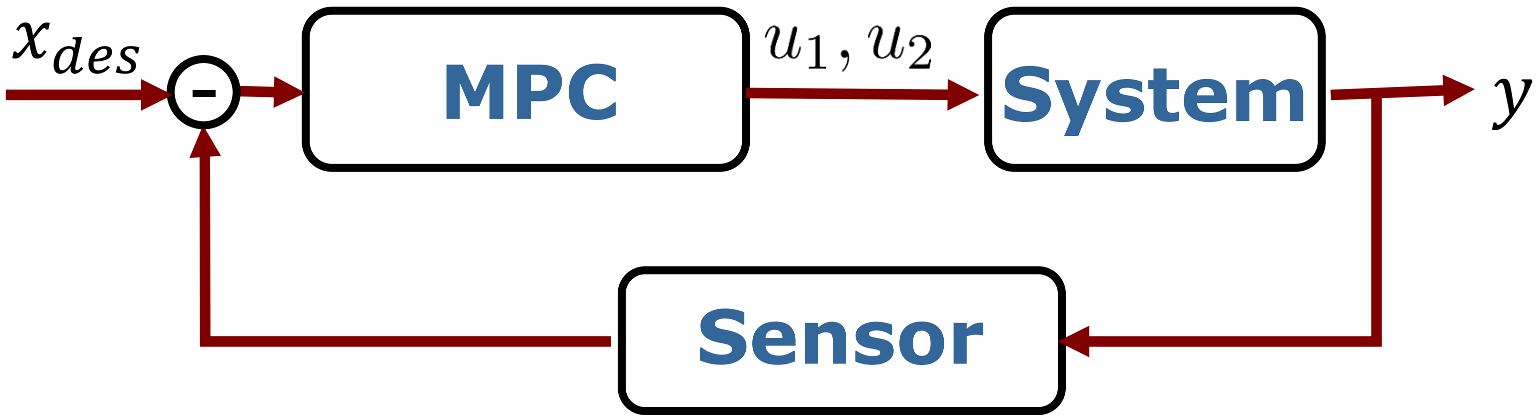
Running MPC Faster

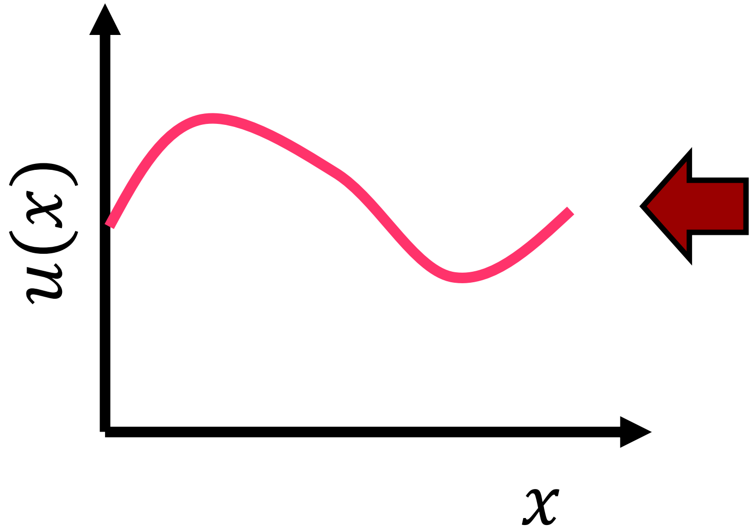


- Prediction Horizon
- Control Horizon
- Sample Time
- Constraints
- Number of iterations



Explicit (Offline) MPC





Perform offline optimization

Lookup for best control

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)**

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

Thank you for your attention