

# ROBOT DESIGN AND APPLICATIONS



## ROBOT TASARIMI VE UYGULAMALARI

Robot Tasarımı ve Uygulamaları Dersi | 2025-2026 Güz Dönemi

**Ders Sorumlusu:** Furkan HANİLÇİ

**Bölüm:** Bilgisayar Mühendisliği

## İletişim Kaynakları



**LinkedIn:**  
**furkanhanilci**

Autonomous Driving Software  
Engineer



**GitHub:**  
**furkanhanilci**

[github.com/furkanhanilci](https://github.com/furkanhanilci)



**E-posta**

[furkan.hanilci@ext.karsan.com.tr](mailto:furkan.hanilci@ext.karsan.com.tr)



# Agenda

**6.1** Introduction

**6.2** Black Box Control

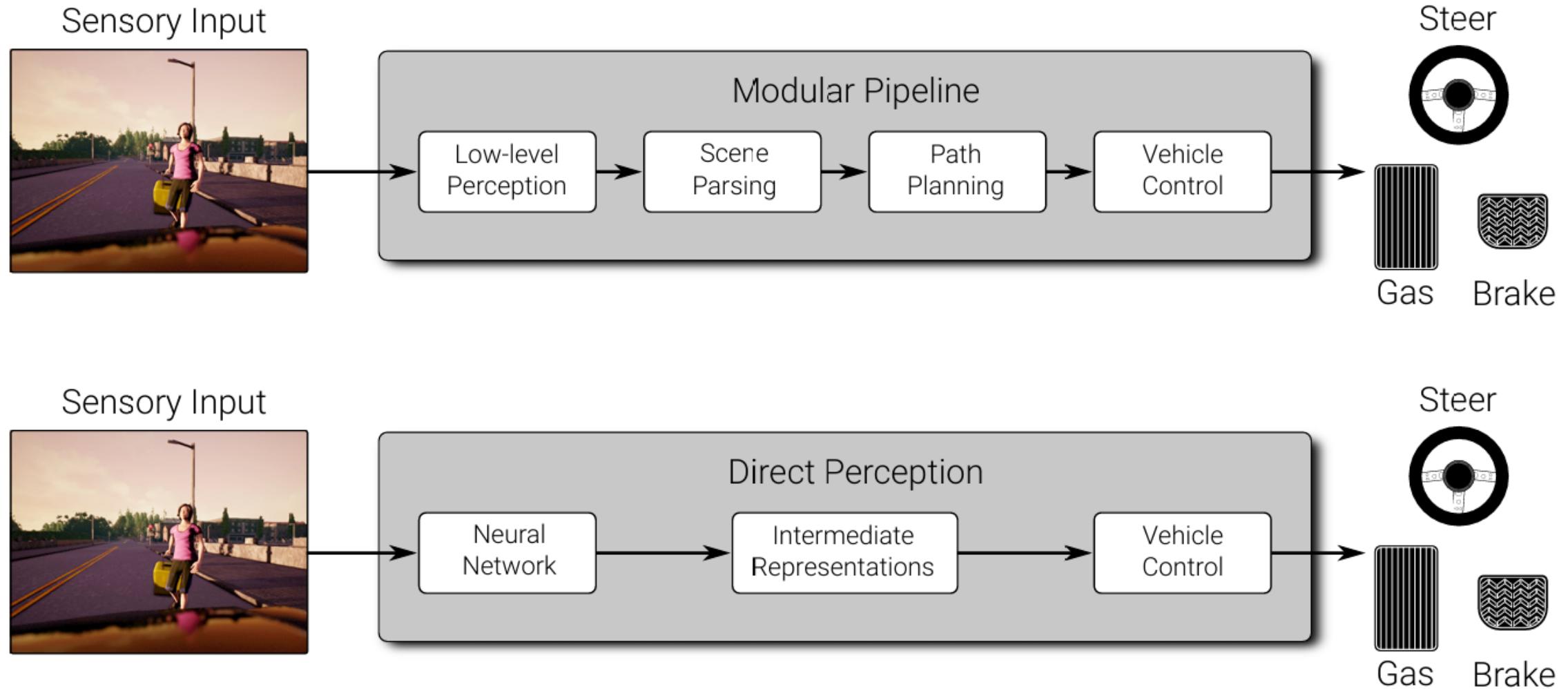
**6.3** Geometric Control

**6.4** Optimal Control

# 6.1

## Introduction

# Approaches to Self-Driving

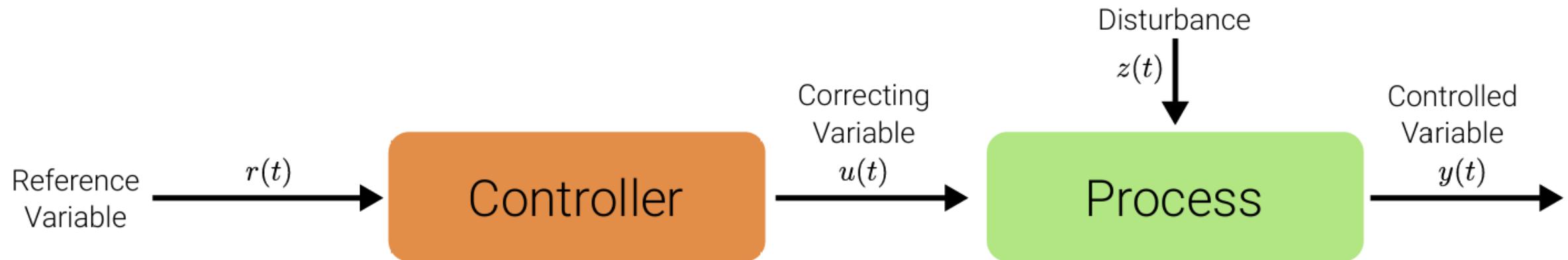


- Self-driving cars and driver assistance systems require **vehicle control**

# Brief History of Driver Assistance Systems

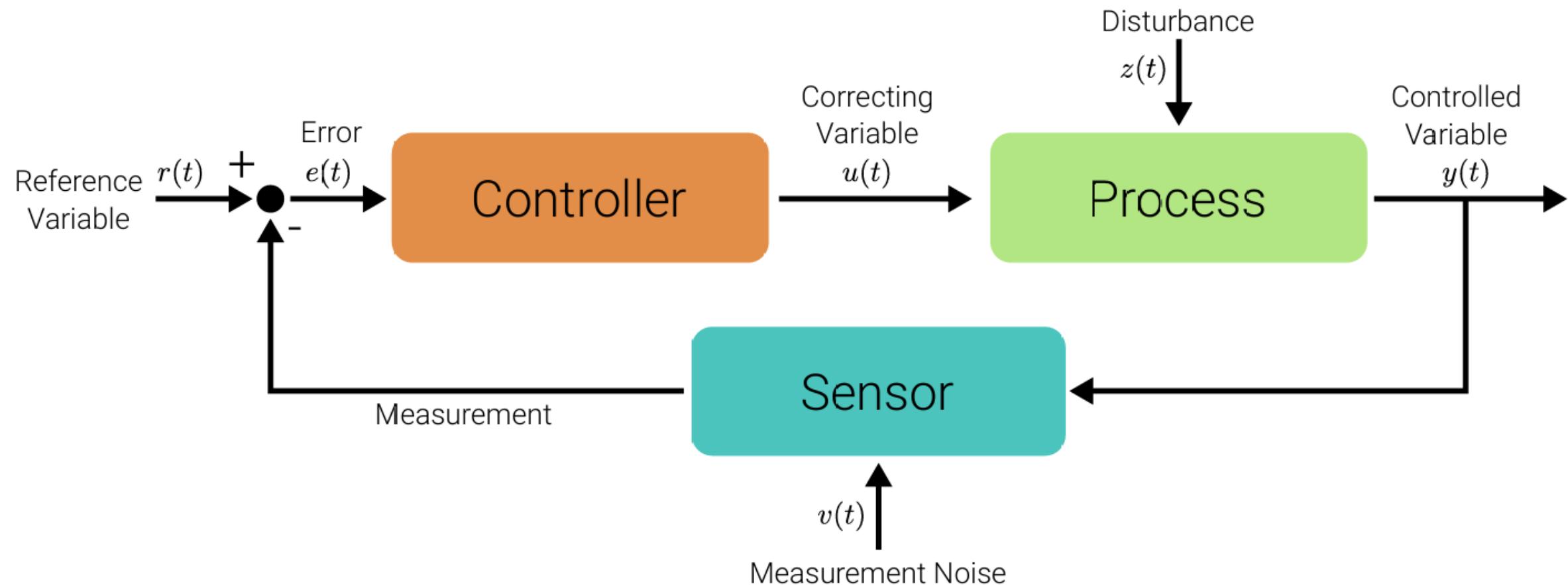
- ▶ 1926: Servo braking (Pierce-Arrow)
- ▶ 1951: Servo steering (Chrysler)
- ▶ 1958: Cruise control (Chrysler)
- ▶ 1978: Anti-lock braking system ABS (Bosch)
- ▶ 1986: Traction control system ASR (Bosch)
- ▶ 1995: Electronic stability program ESP (Bosch/BMW)
- ▶ 2000: Adaptive cruise control ACC (Mitsubishi/Toyota/Bosch)
- ▶ 2002: Emergency brake assistant (Mercedes Benz)
- ▶ 2003: Lane-keeping assistant (Honda)
- ▶ 2007: Automatic park assistant (Valeo)

# Open-Loop Control



- ▶ Requires **precise knowledge** of the plant and the influence factors
- ▶ **No feedback** about the controlled variable
- ▶ Cannot handle unknown disturbances, resulting in **drift**

# Closed-Loop Control



- ▶ Exploit feedback loop to minimize error between reference and measurement

# Closed-Loop Control

- ▶ We will be considering **closed-loop control** in this lecture
- ▶ A vehicle needs to be controlled both **longitudinally** and **laterally**
- ▶ We consider 3 different types of controllers:
  - ▶ **Black box controllers** don't require knowledge about the process
  - ▶ **Geometric controllers** exploit geometric relationships between the vehicle and the path, resulting in compact control laws for path tracking
  - ▶ **Optimal controllers** use knowledge of the system and minimize an objective function over future time steps

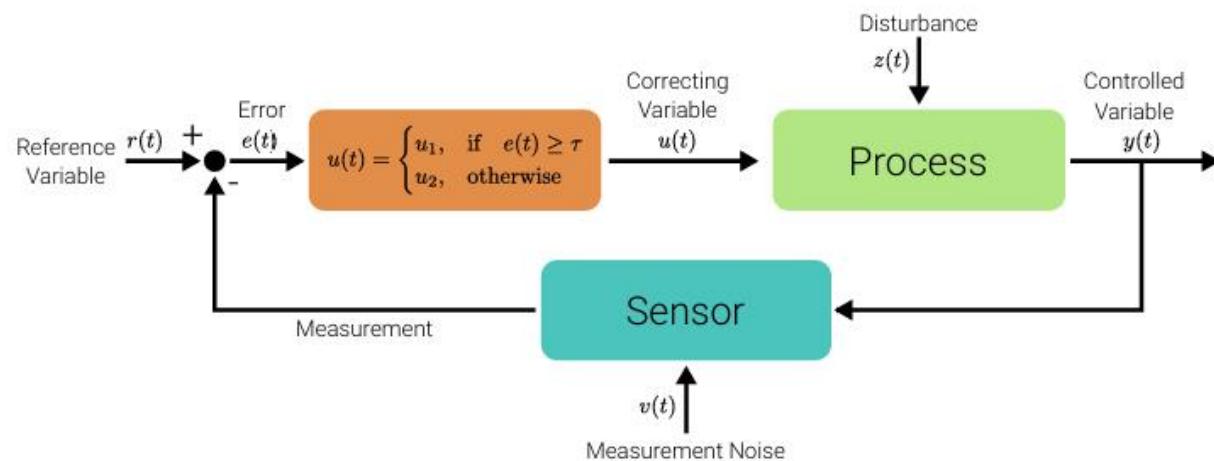
## 6.2

# Black Box Control

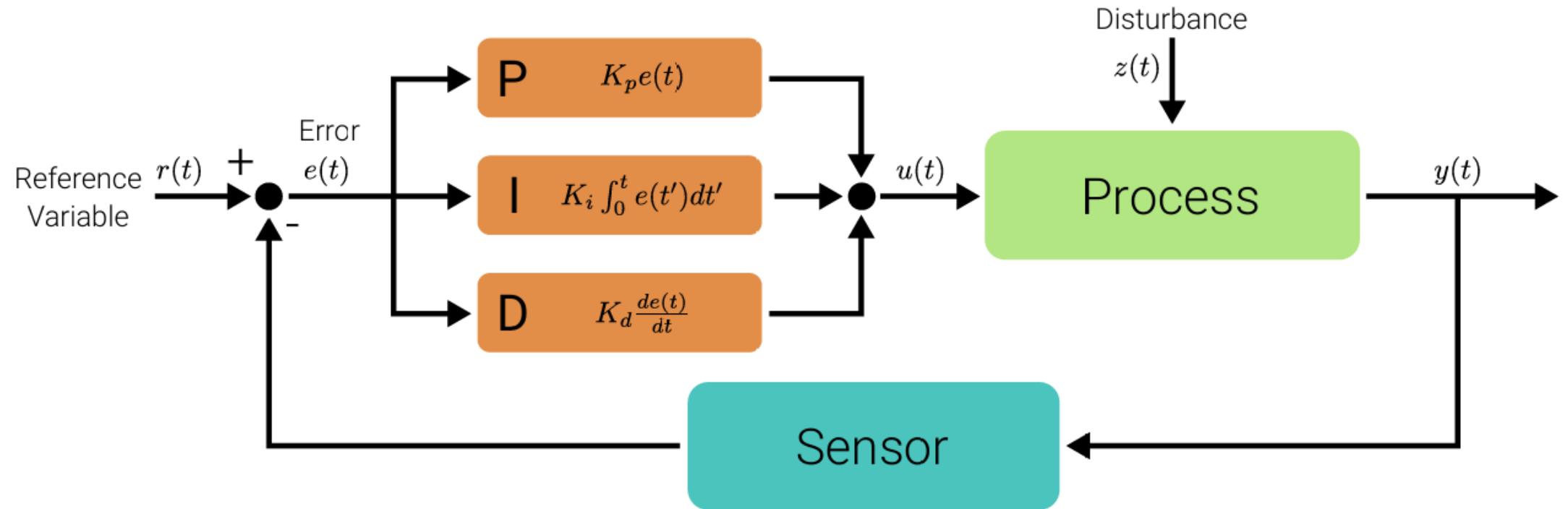
# Bang-Bang Control

- ▶ Also called: hysteresis controller
- ▶ Often applied, e.g. in household thermostats
- ▶ Switches abruptly between two states
- ▶ Mathematical formulation:

$$u(t) = \begin{cases} u_1, & \text{if } e(t) \geq \tau \\ u_2, & \text{otherwise} \end{cases}$$

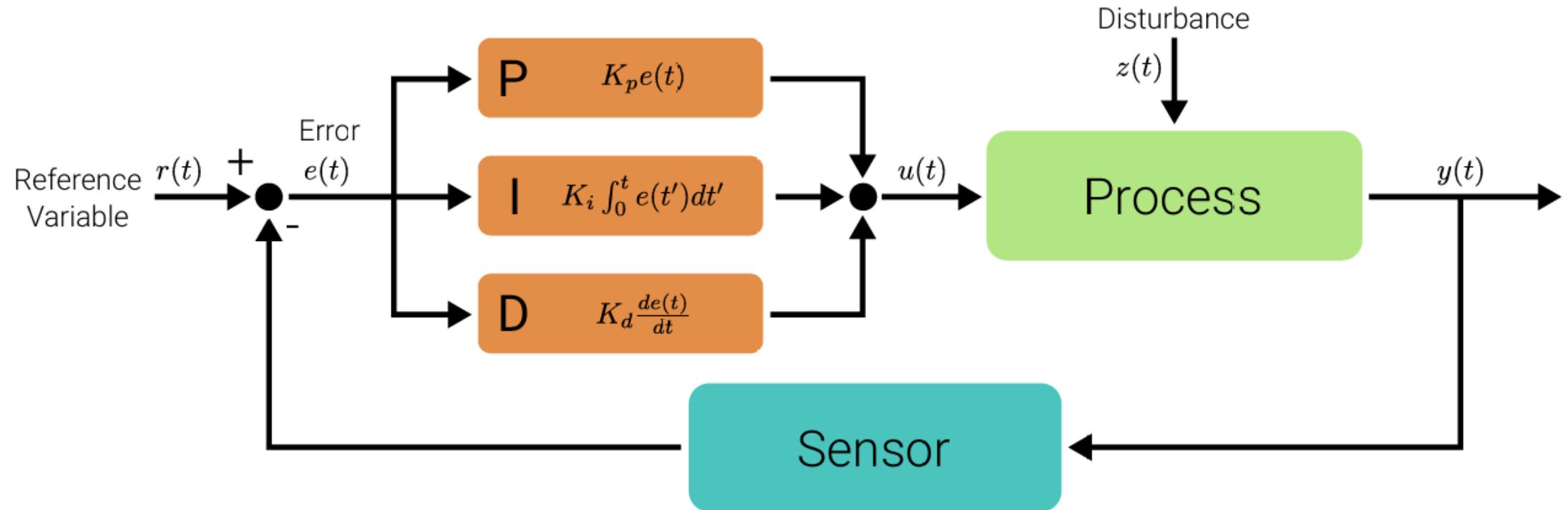


# PID Control



- Most common controller in current industrial applications
- Doesn't require knowledge about plant/process to be controlled
- Comprises proportional (**P**), integration (**I**) and differential (**D**) element
- Reference variable  $r(t)$ , correcting variable  $u(t)$ , controlled variable  $y(t)$ , error  $e(t)$

# PID Control

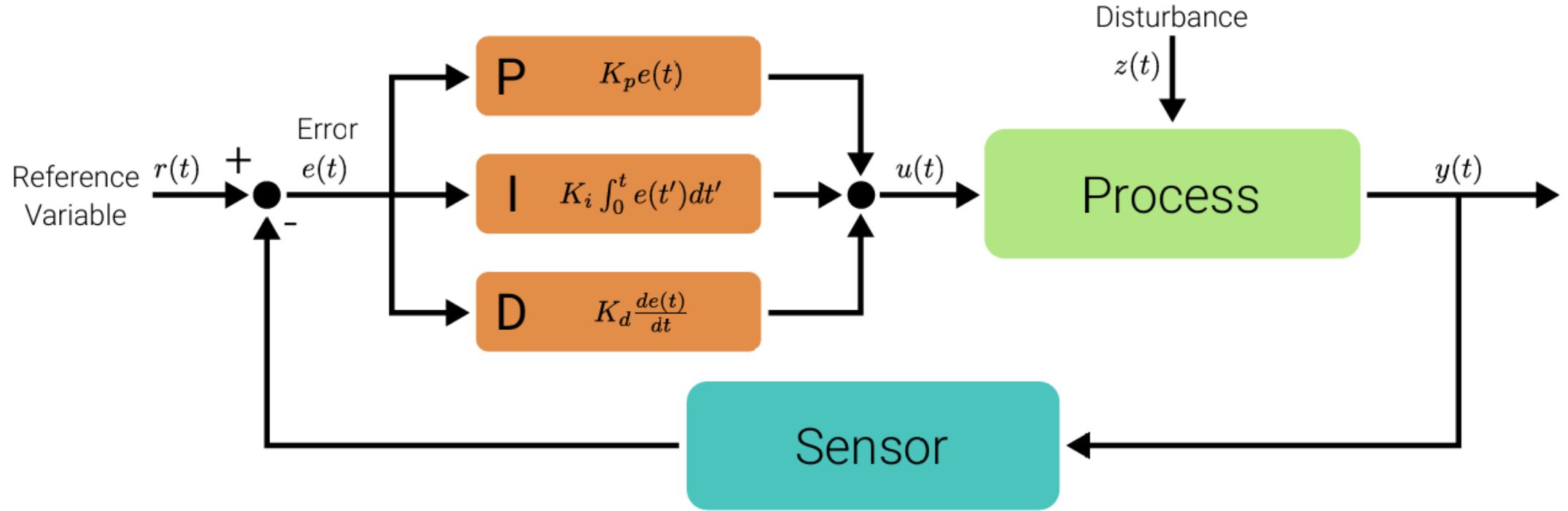


Mathematical formulation:

$$u(t) = K_p e(t) + K_i \int_0^t e(t') dt' + K_d \frac{de(t)}{dt}$$

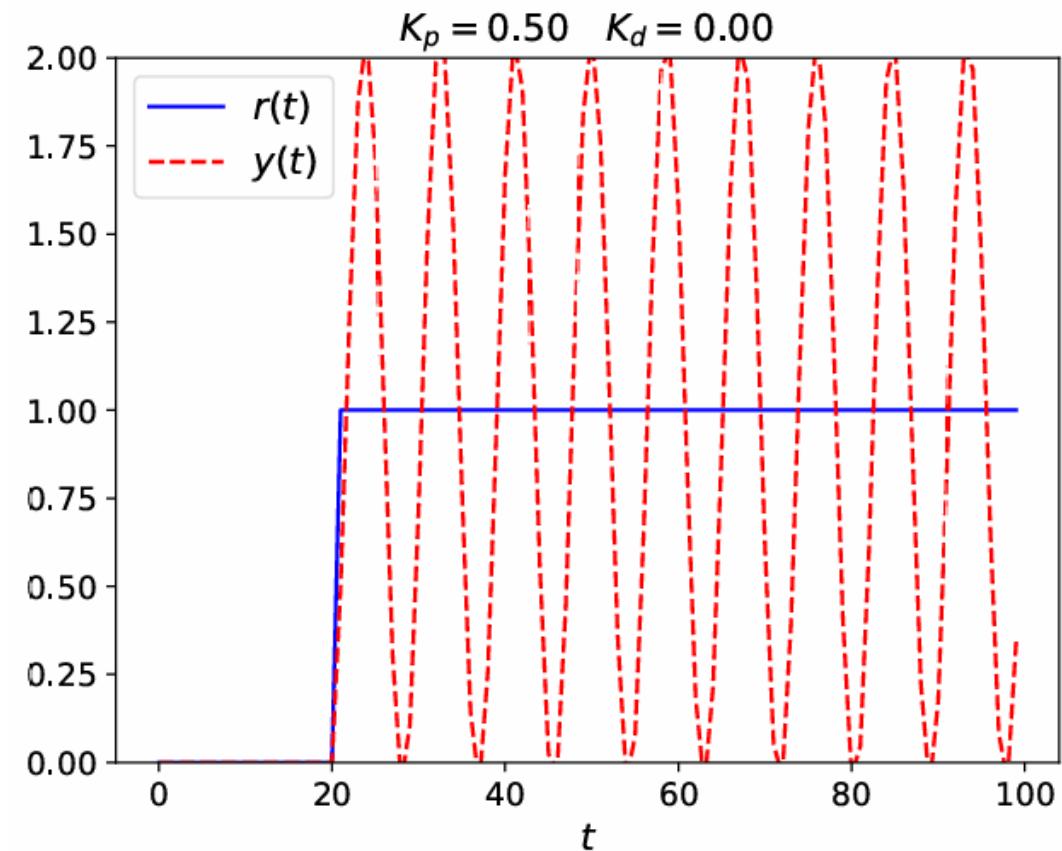
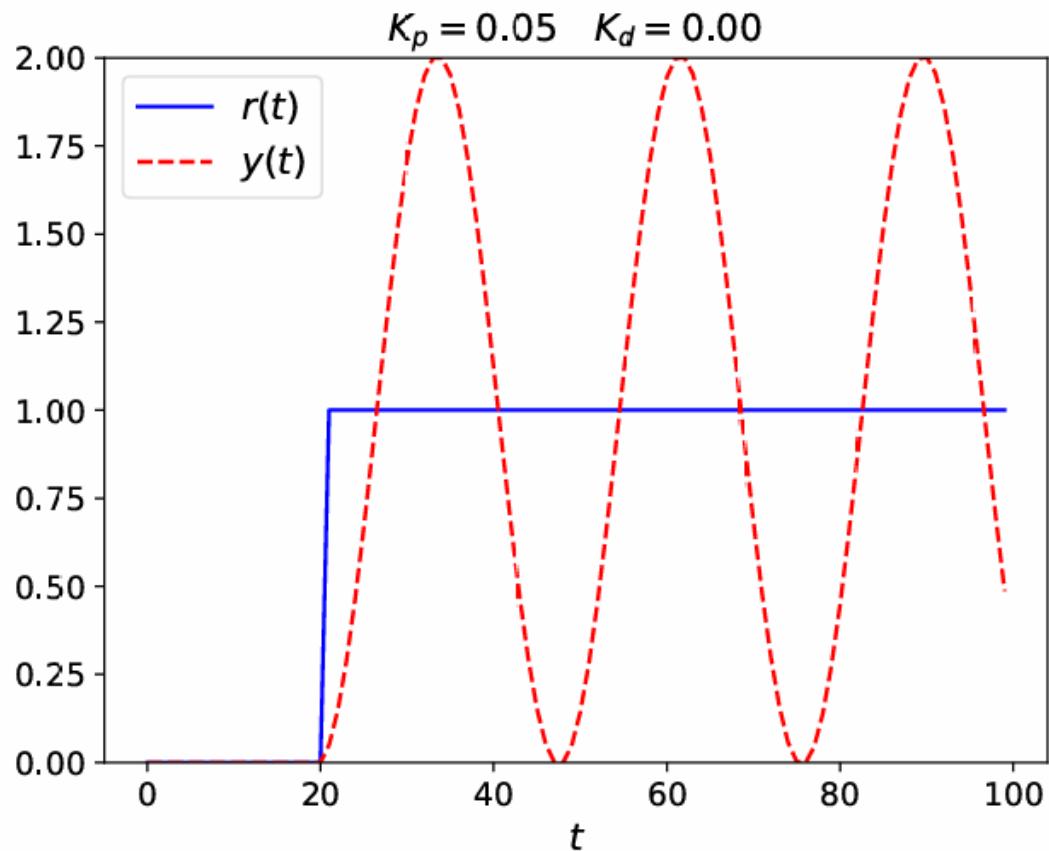
with parameters  $K_p$ ,  $K_i$  and  $K_d$

# PID Control



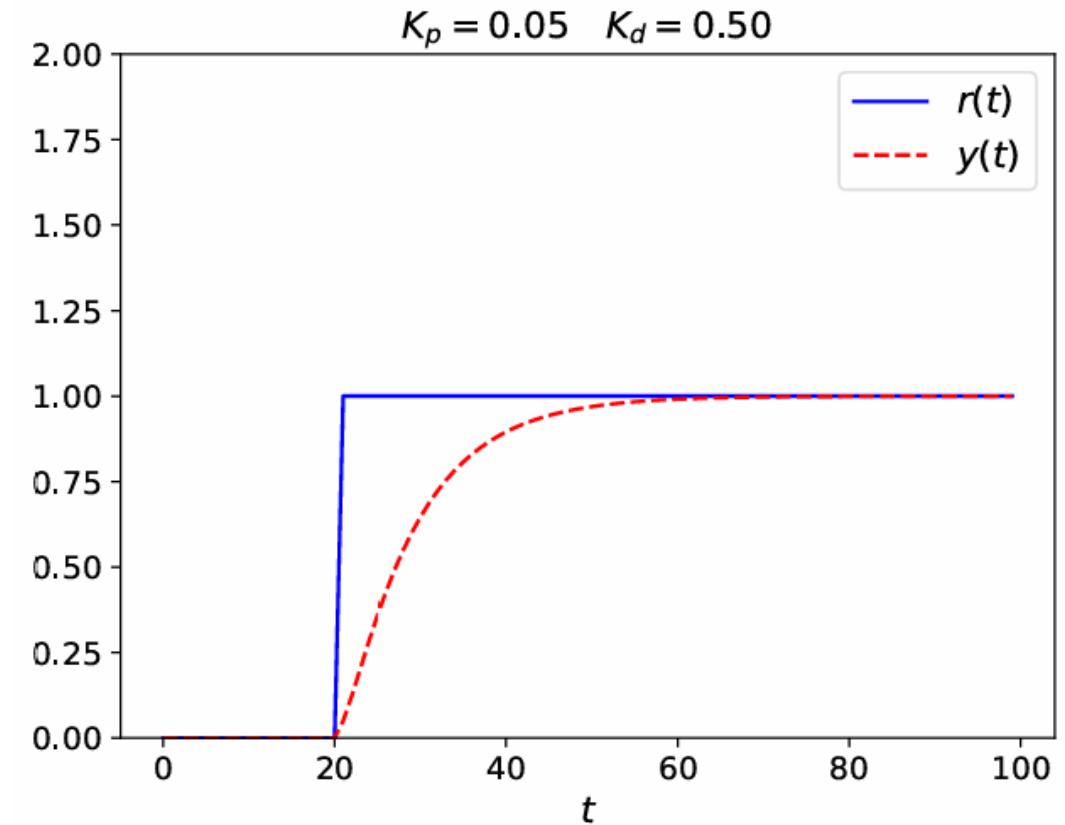
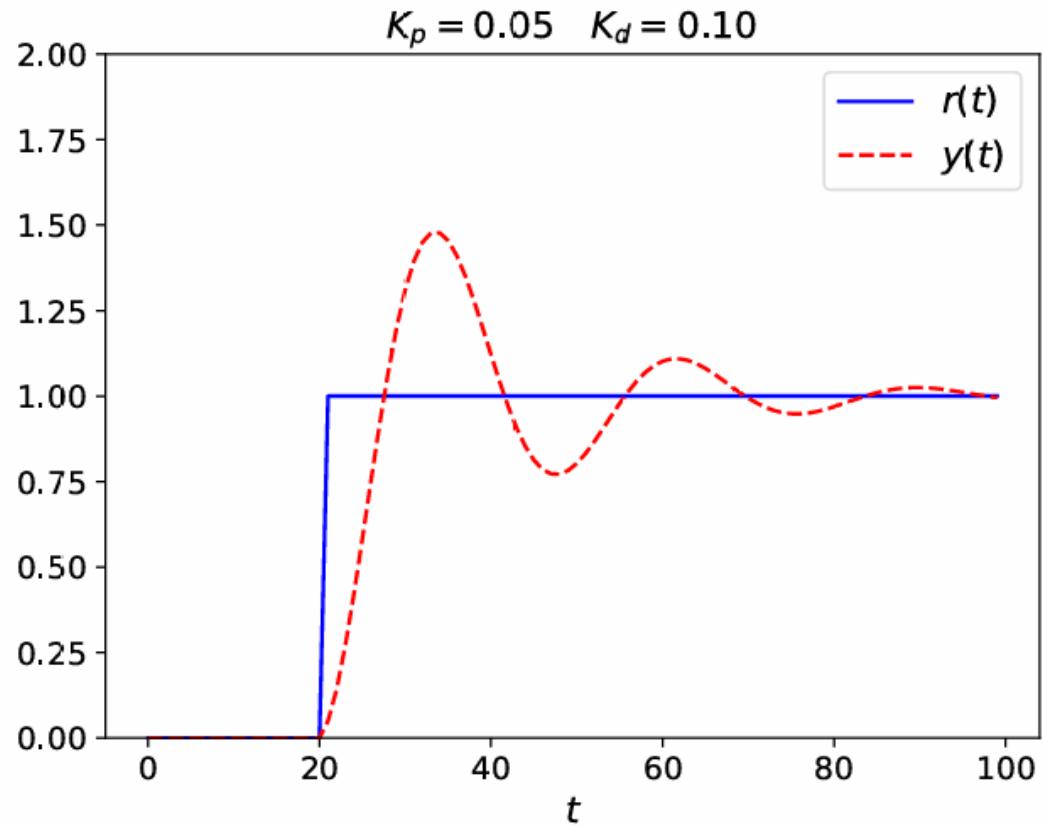
- ▶ Using the **P** element alone leads to overshooting / oscillation
- ▶ Adding a **D** element alleviates this problem by introducing damping behavior
- ▶ The **I** element corrects residual errors by integrating past error measurements

# P Control



- ▶ Controlled Variable: Position  $y(t) = x(t)$
- ▶ Correcting Variable: Acceleration  $u(t) = a(t) = \ddot{x}(t)$

# PD Control



- ▶ Controlled Variable: Position  $y(t) = x(t)$
- ▶ Correcting Variable: Acceleration  $u(t) = a(t) = \ddot{x}(t)$

# PID Control

As PID controllers are often used to control black box processes, various heuristics have been proposed for setting the parameters. **Ziegler-Nichols** is most common:

- ▶ Set  $K_i = K_d = 0$
- ▶ Increase  $K_p$  until the ultimate gain  $K_p = K_u$  where the system oscillates
- ▶ Measure the oscillation period  $T_u$  at  $K_u$
- ▶ Set  $K_p = 0.6K_u$ ,  $K_i = 1.2K_u/T_u$  and  $K_d = 3K_uT_u/40$

These heuristics have been derived empirically using computer simulations. While they provide a good starting point, manual fine-tuning is often required in practice.

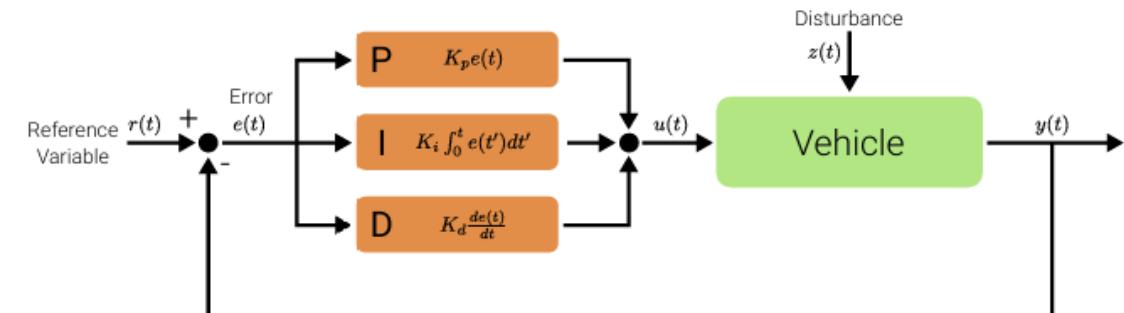
Note: Ziegler-Nichols does not apply to double integrators as in the previous example.

# PID Control

## Example: Longitudinal Vehicle Control

$$v(t) = v_{\max} (1 - \exp(-\theta_1 d(t) - \theta_2))$$

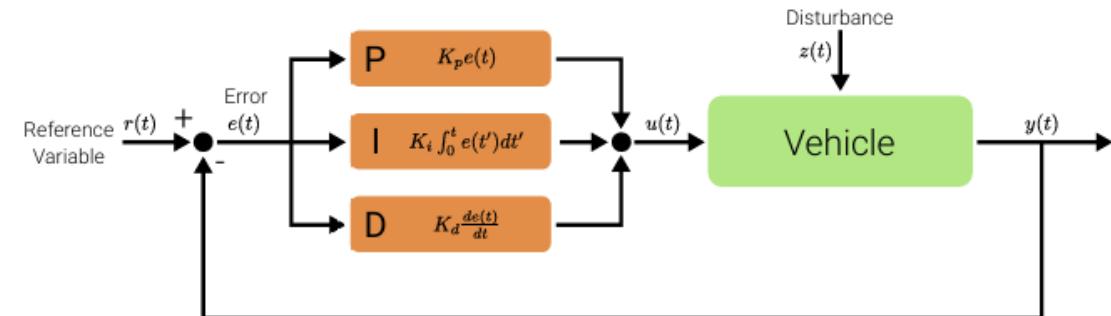
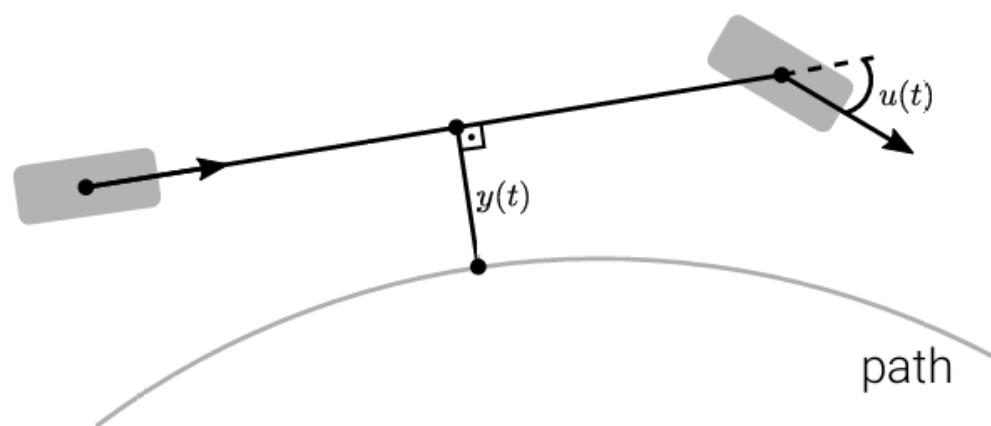
- ▶  $v(t)$ : target velocity at time  $t$
- ▶  $d(t)$ : distance to preceding car



- ▶ Reference variable:  $r(t) = v(t) = \text{target velocity}$
- ▶ Correcting variable:  $u(t) = \text{gas/brake pedal}$
- ▶ Controlled variable:  $y(t) = \text{current velocity}$
- ▶ Error:  $e(t) = v(t) - y(t)$

# PID Control

## Example: Lateral Vehicle Control



- ▶ Reference variable:  $r(t) = 0$  = no cross track error
- ▶ Correcting variable:  $u(t) = \delta$  = steering angle
- ▶ Controlled variable:  $y(t)$  = cross track error
- ▶ Error:  $e(t) = -y(t)$  = cross track error

# PID Control

The **steady-state error** is defined as the difference between the input (command) and the output of a system in the limit as time goes to infinity (i.e., when the response has reached steady state). Steady-state error analysis is only useful for **stable** systems (a linear system is stable if its output will stay bounded for any bounded input).

# PID Control

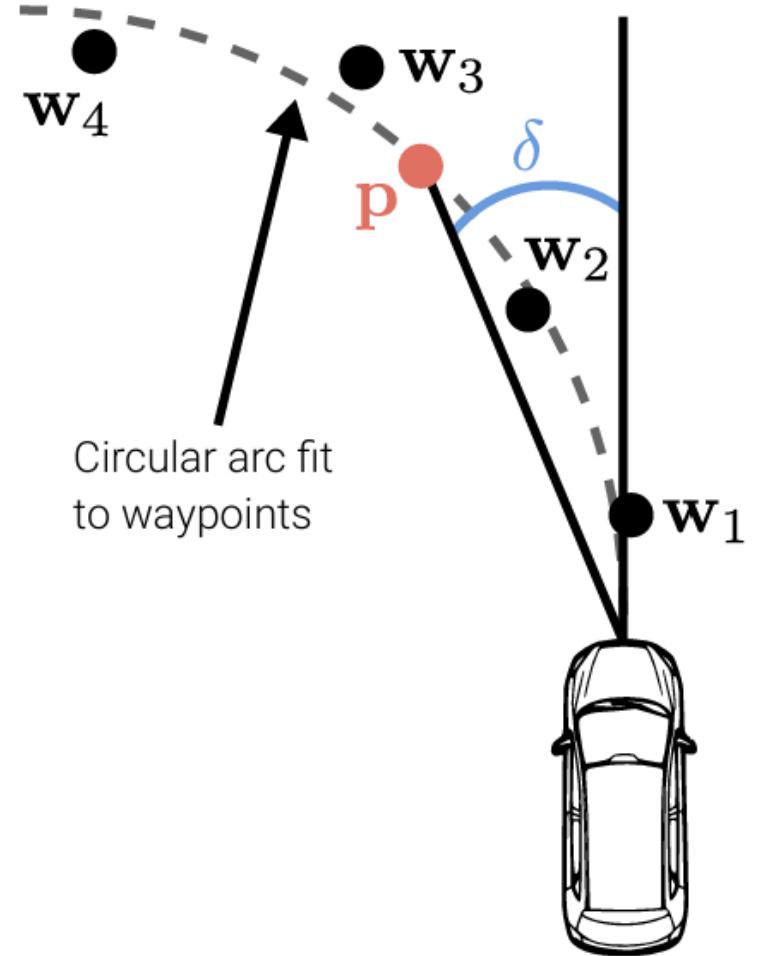
## Example: Waypoint-based Vehicle Control

- ▶ **Input:** Waypoints  $\mathbf{w} = \{\mathbf{w}_1, \dots, \mathbf{w}_K\}$
- ▶ **Velocity:** (Longitudinal PID control)

$$v = \frac{1}{K} \sum_{k=1}^K \frac{\|\mathbf{w}_k - \mathbf{w}_{k-1}\|_2}{\Delta t}$$

- ▶ **Steering angle:** (Lateral PID control)

$$\delta = \tan^{-1} \left( \frac{p_y}{p_x} \right)$$



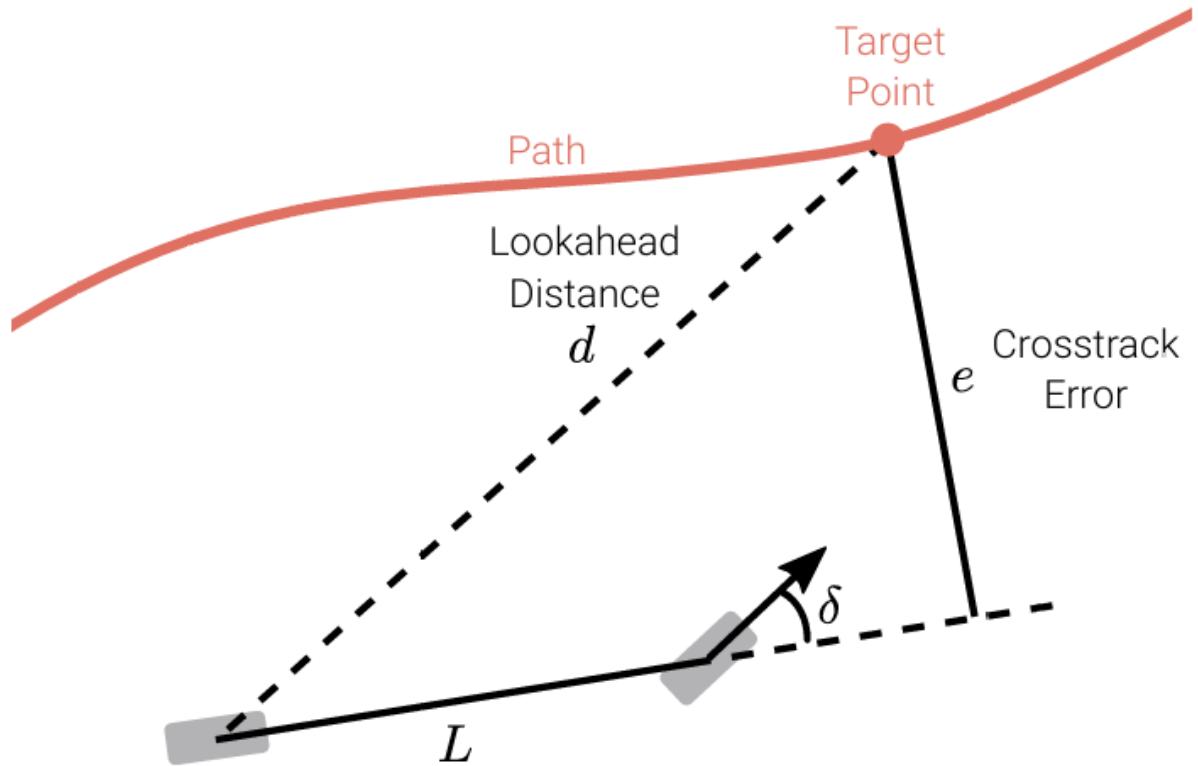
# 6.3

## Geometric Control

# Pure Pursuit Control

## Goal:

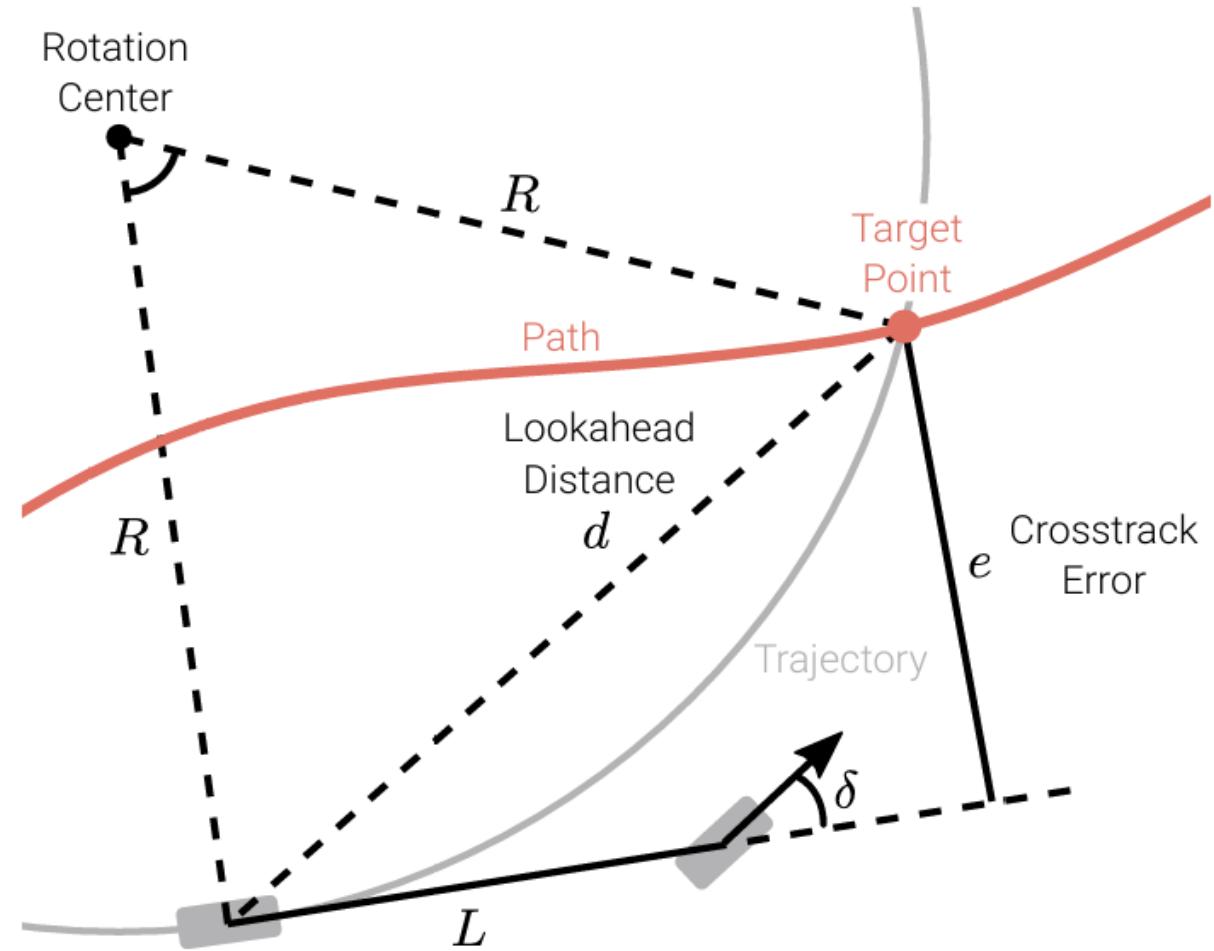
- ▶ Track **target point** at **lookahead distance**  $d$  to follow path
- ▶ Exploit geometric relationship between vehicle and path to follow



# Pure Pursuit Control

## Goal:

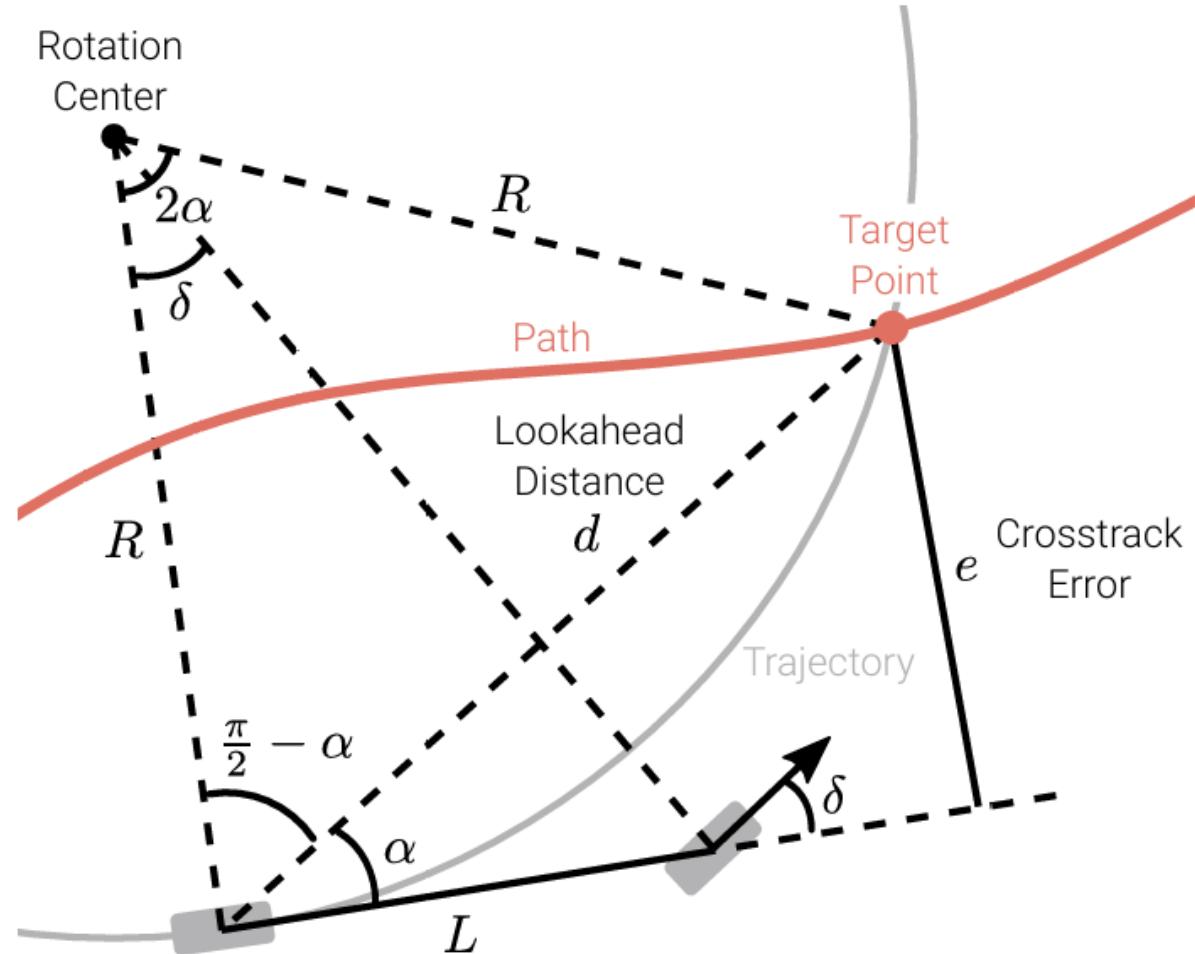
- ▶ Track **target point** at **lookahead distance**  $d$  to follow path
- ▶ Exploit geometric relationship between vehicle and path to follow
- ▶ Minimize **crosstrack error**  $e$  by following circular trajectory



# Pure Pursuit Control

## Goal:

- ▶ Track **target point** at **lookahead distance**  $d$  to follow path
- ▶ Exploit geometric relationship between vehicle and path to follow
- ▶ Minimize **crosstrack error**  $e$  by following circular trajectory
- ▶ Steering angle  $\delta$  determined by angle  $\alpha$  between vehicle heading direction and lookahead direction:  $\delta(\alpha) = ?$



# Pure Pursuit Control

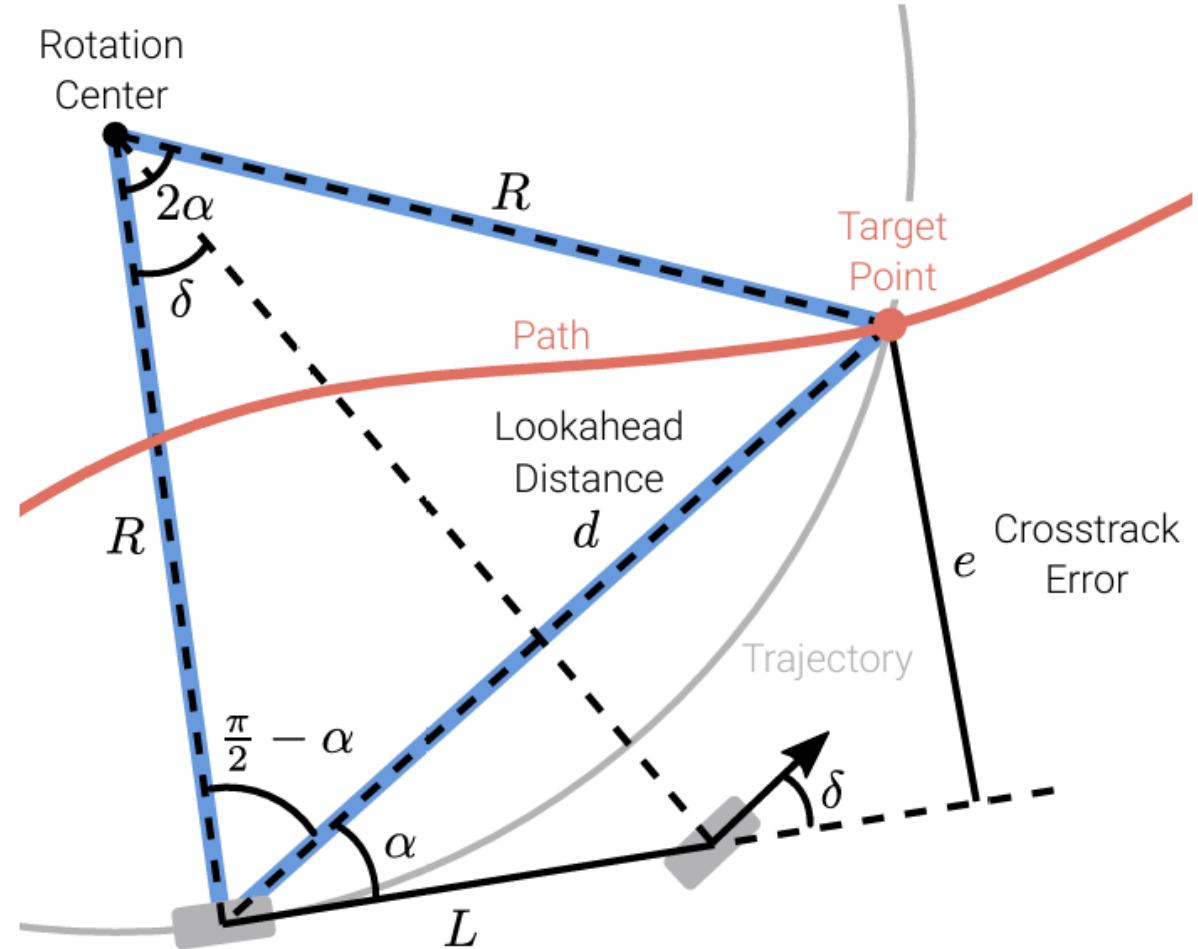
From the **law of sines**:

$$\frac{d}{\sin(2\alpha)} = \frac{R}{\sin(\frac{\pi}{2} - \alpha)}$$

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

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

with  $\kappa$  the curvature of trajectory.



# Pure Pursuit Control

The **steering angle** is calculated as:

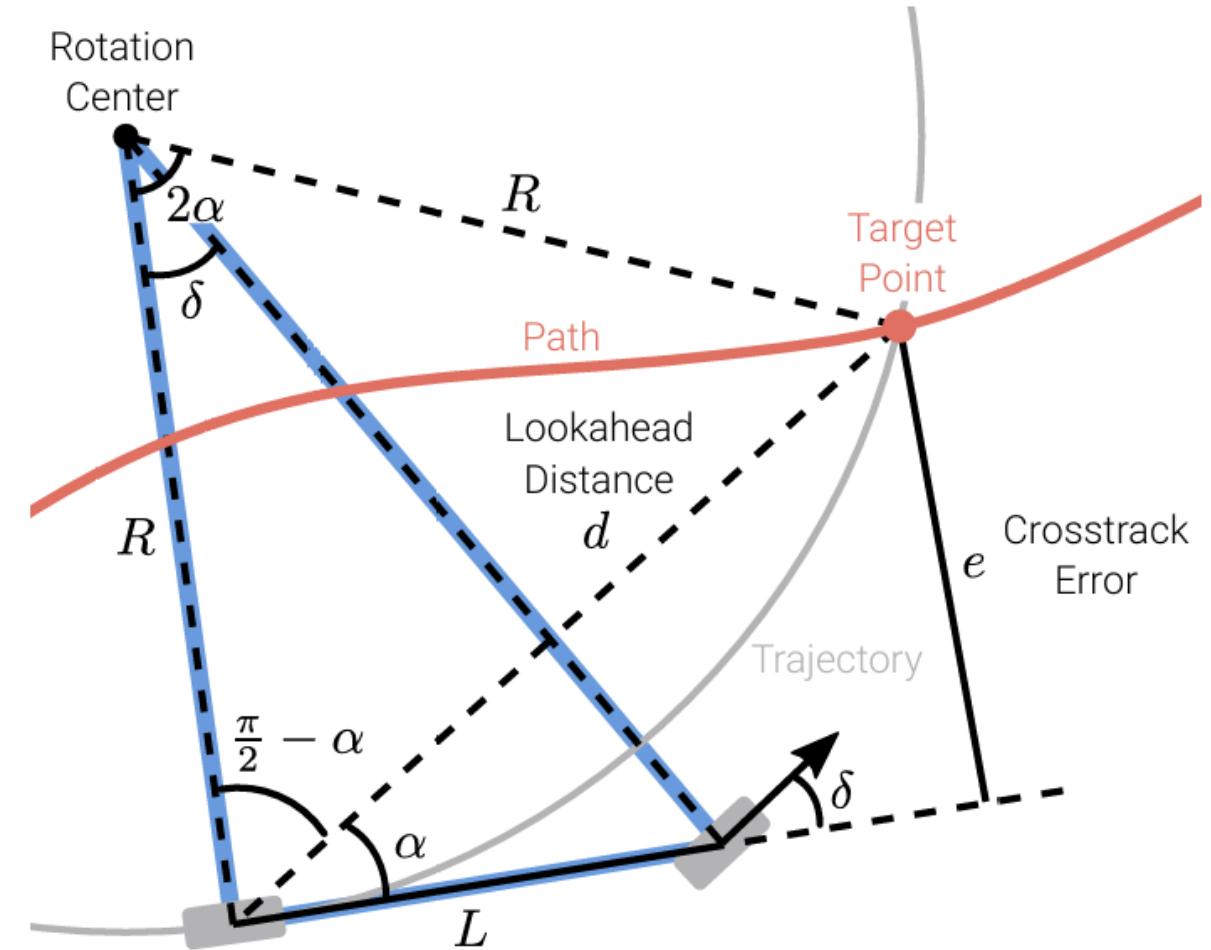
$$\tan(\delta) = \frac{L}{R} = \frac{2L \sin(\alpha)}{d}$$

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

$$\delta \approx \frac{2L \sin(\alpha)}{d}$$

- $d$  is often based on vehicle speed  $v$ :

$$d = Kv \quad \text{with constant } K$$



# Pure Pursuit Control

In terms of **cross track error** we obtain:

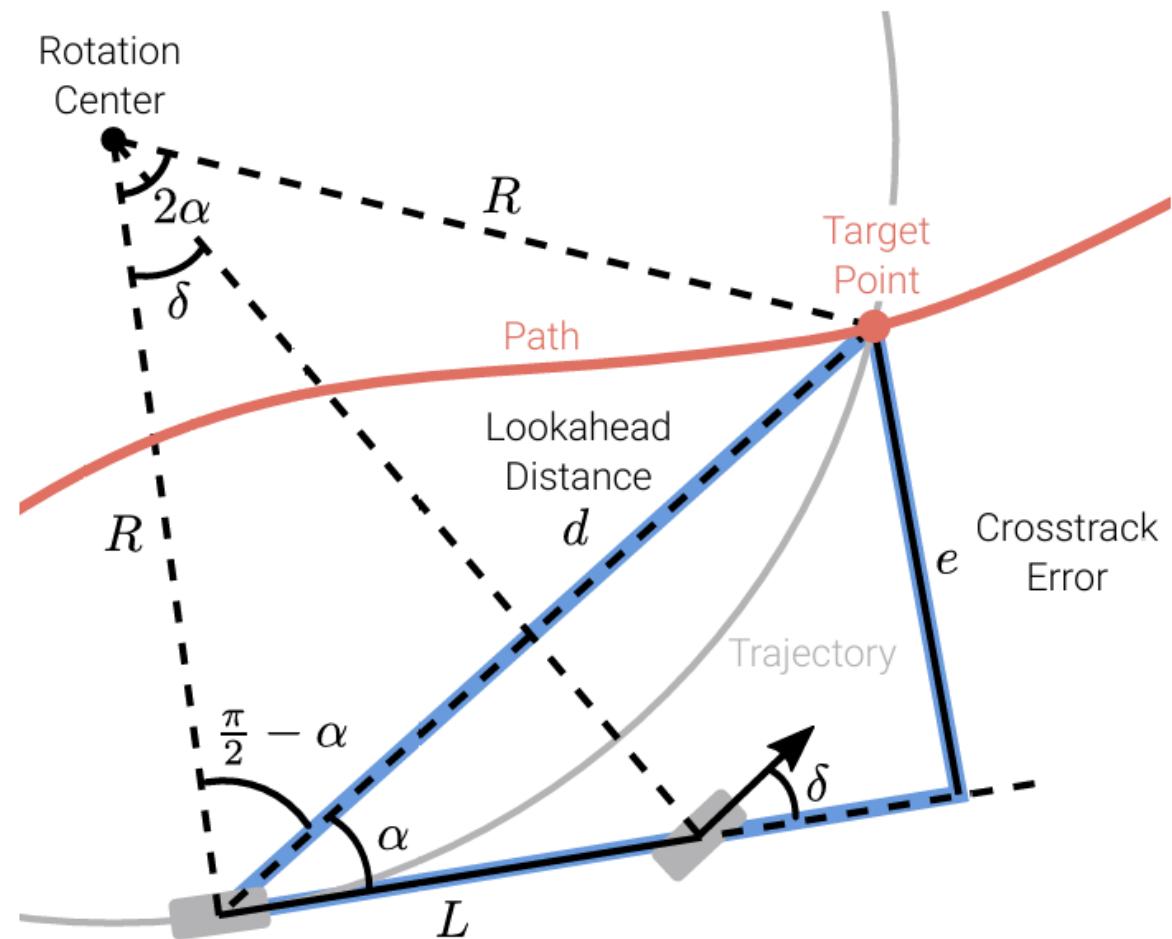
$$\sin \alpha = \frac{e}{d}$$

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

$$\delta = \tan^{-1} \left( \frac{2Le}{d^2} \right) \approx \frac{2L}{d^2} e$$

- Pure pursuit acts as a **proportional** controller wrt. the crosstrack error
- $d$  is often based on vehicle speed  $v$ :

$$d = Kv \quad \text{with constant } K$$

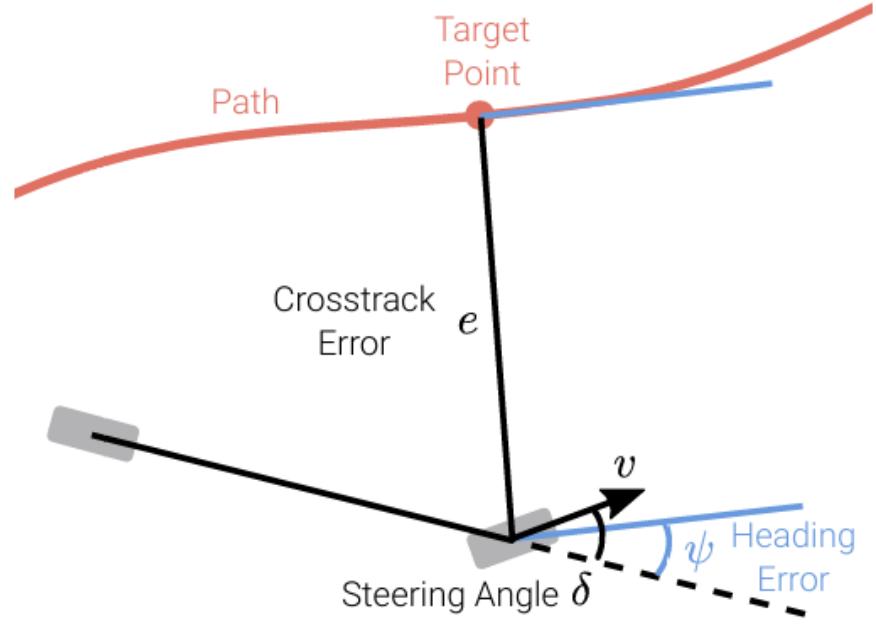


# Stanley Control

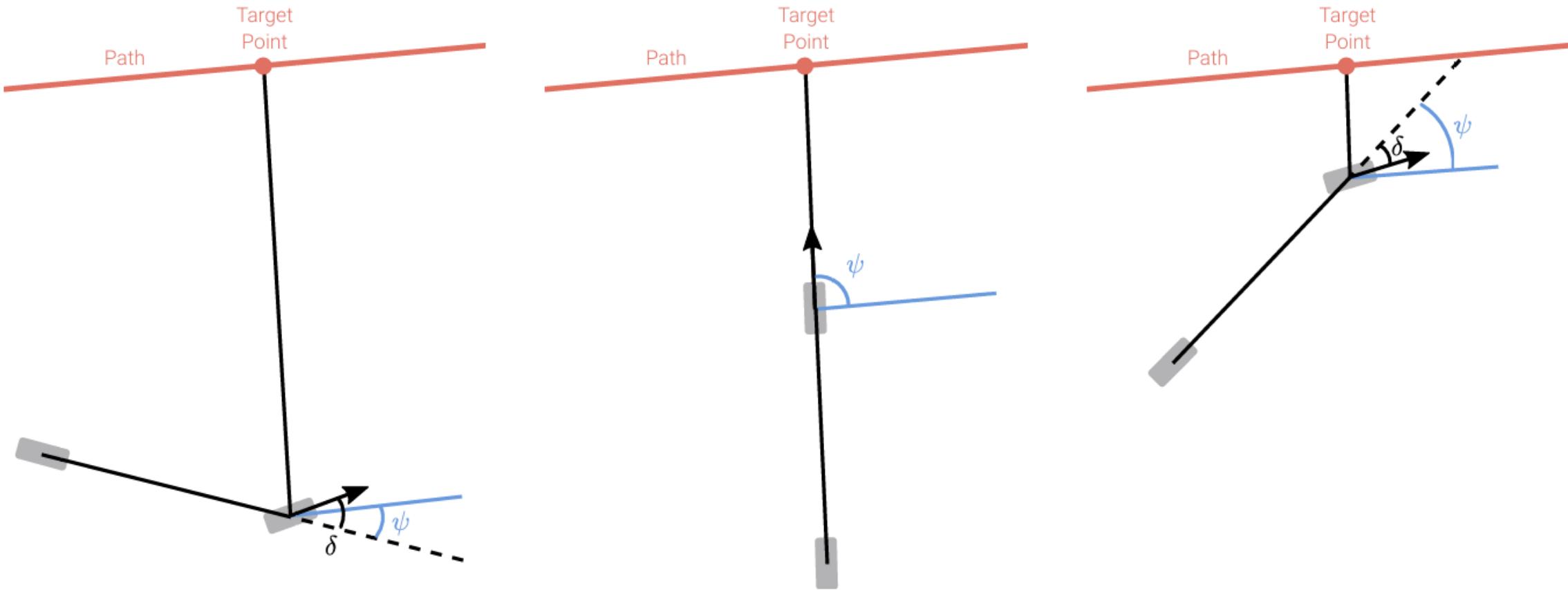
**Control Law** used by **Stanley** in **DARPA Challenge**:

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

- ▶  $v$  = speed,  $\psi$  = heading err.,  $e$  = crosstrack err.
- ▶ Reference at **front axle, no lookahead**
- ▶ Combines **heading** and **crosstrack error**
- ▶ It can be shown that the crosstrack error **converges** exponentially to 0 (indep. of  $v$ )
- ▶ Works for small velocities w/o disturbances

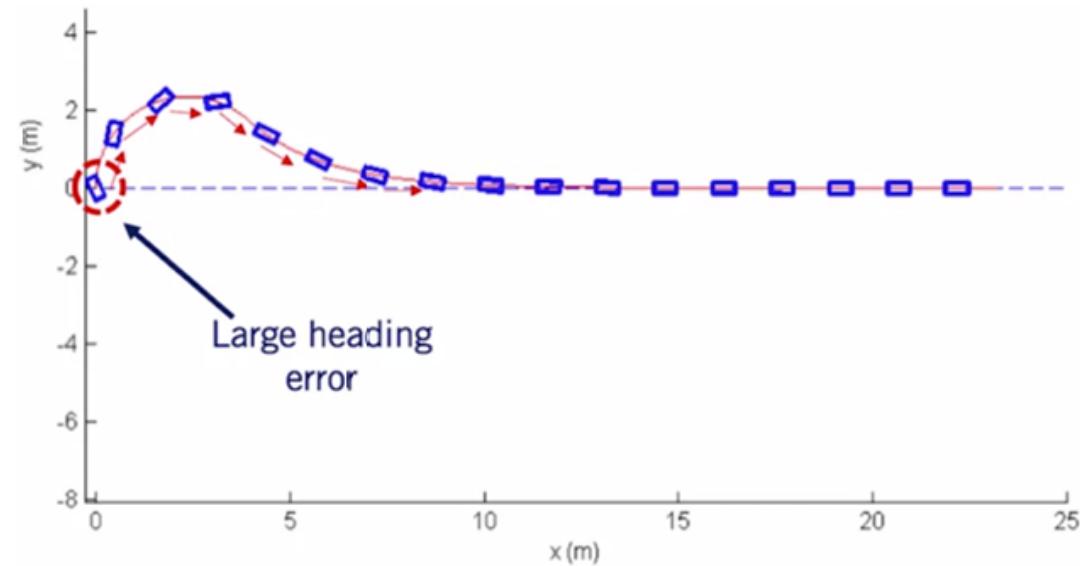
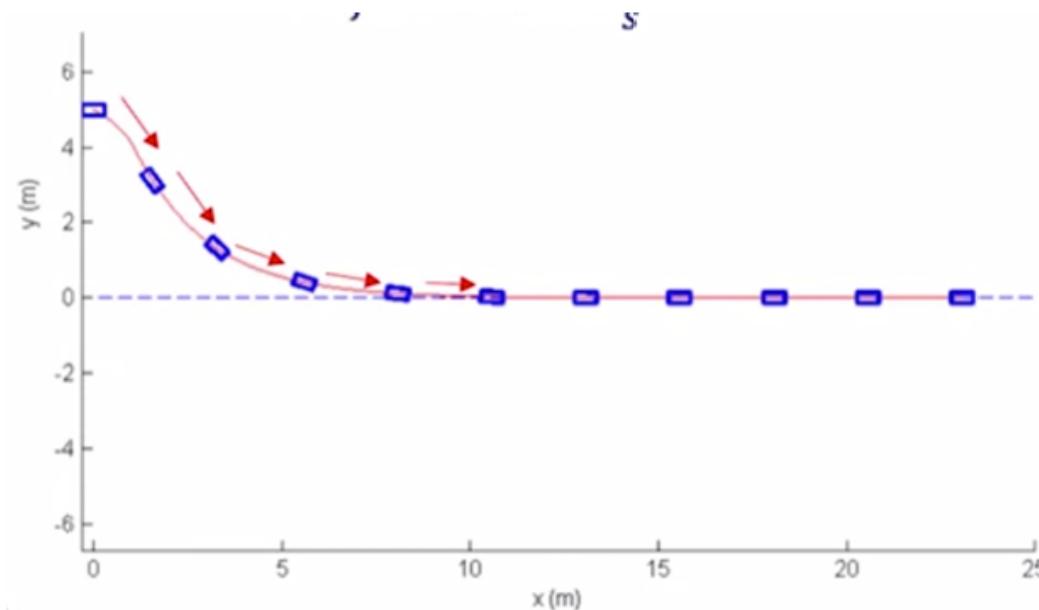


# Stanley Control



- As heading changes, heading correction **counteracts** crosstrack correction

# Stanley Control



- ▶ The Stanley controller can correct **large crosstrack** and **large heading errors**
- ▶ **Global stability:** Independent of initial conditions guides vehicle back (proven)
- ▶ But does not consider noisy observations, actuator dynamics, tire force effects
- ▶ Softening/dampening terms and curvature information can be added

# 6.4

## Optimal Control

# Optimal Control

**Recap:** Dynamic Bicycle Model

$$\begin{bmatrix} \dot{v}_y \\ \dot{\psi} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} -\frac{c_r + c_f}{mv_x} & 0 & \frac{c_r l_r - c_f l_f}{mv_x} - v_x \\ 0 & 0 & 1 \\ \frac{l_r c_r - l_f c_f}{I_z v_x} & 0 & -\frac{l_f^2 c_f + l_r^2 c_r}{I_z v_x} \end{bmatrix} \underbrace{\begin{bmatrix} v_y \\ \psi \\ \omega \end{bmatrix}}_{\text{State } \mathbf{x}} + \underbrace{\begin{bmatrix} \frac{c_f}{m} \\ 0 \\ \frac{c_f}{I_z} l_f \end{bmatrix}}_{\text{Input}} \delta$$

With state  $\mathbf{x}$  and front steering angle  $\delta$ .

We can rewrite this equation as the following linear system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}\delta \quad \text{with} \quad \mathbf{x} = (v_y, \psi, \omega)^T$$

# Optimal Control

**Linear Quadratic Regulator (LQR):** For the continuous-time linear system

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}\delta \quad \text{with} \quad \mathbf{x}(0) = \mathbf{x}_{\text{init}}$$

and quadratic cost functional defined as ( $\mathbf{Q}$  is a diagonal weight matrix)

$$J = \frac{1}{2} \int_0^{\infty} \Delta\mathbf{x}^T(t) \mathbf{Q} \Delta\mathbf{x}(t) + q\delta(t)^2 dt$$

with  $\Delta\mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}_{\text{target}}$ . The feedback control  $\delta(t)$  that minimizes  $J$  is given by

$$\delta(t) = -\mathbf{k}^T(t) \Delta\mathbf{x}(t)$$

with  $\mathbf{k}(t) = \frac{1}{q} \mathbf{b}^T \mathbf{P}(t)$  and  $\mathbf{P}(t)$  the solution to a Riccati equation (no details here).

# Model Predictive Control

## Generalizes LQR to:

- ▶ **Non-linear** cost function and dynamics (consider straight road leading into turn)
- ▶ **Flexible**: allows for receding window & incorporation of constraints
- ▶ **Expensive**: non-linear optimization required at every iteration

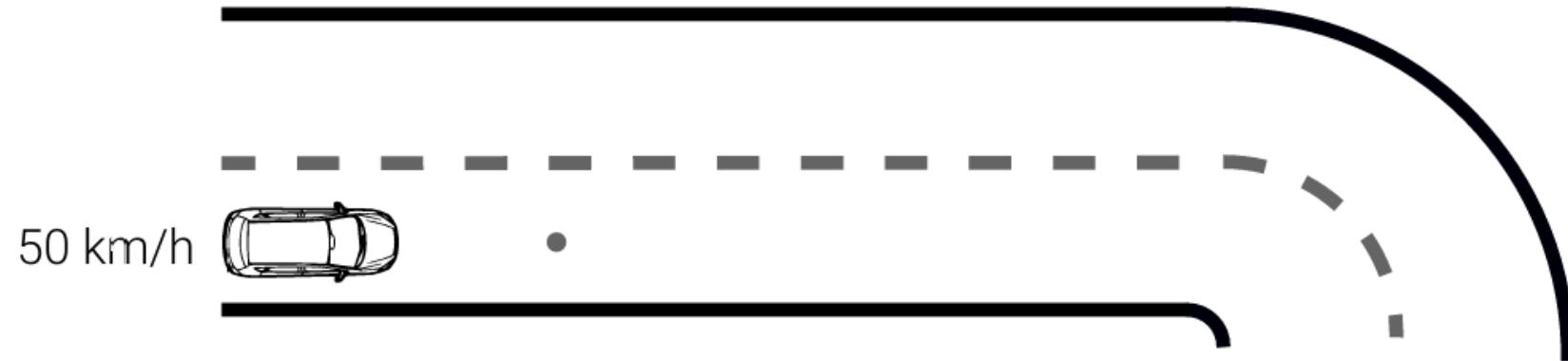
## Formally:

$$\begin{aligned} \operatorname{argmin}_{\delta_1, \dots, \delta_T} \quad & \sum_{t=1}^T C_t(\mathbf{x}_t, \delta_t) && \text{(sum of costs)} \\ \text{s.t.} \quad & \mathbf{x}_1 = \mathbf{x}_{\text{init}} && \text{(initialization)} \\ & \mathbf{x}_{t+1} = f(\mathbf{x}_t, \delta_t) && \text{(dynamics model)} \\ & \underline{\delta} \leq \delta_t \leq \bar{\delta} && \text{(constraints)} \end{aligned}$$

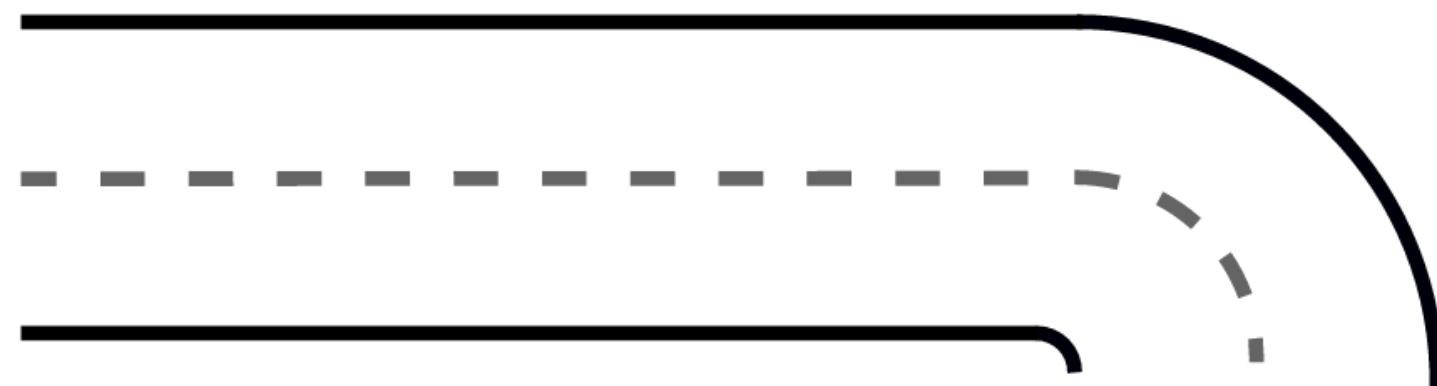
- ▶ Unroll dynamic model  $T$  times  $\Rightarrow$  apply non-linear optimization to find  $\delta_1, \dots, \delta_T$

# Model Predictive Control

PID Control / Path Tracking

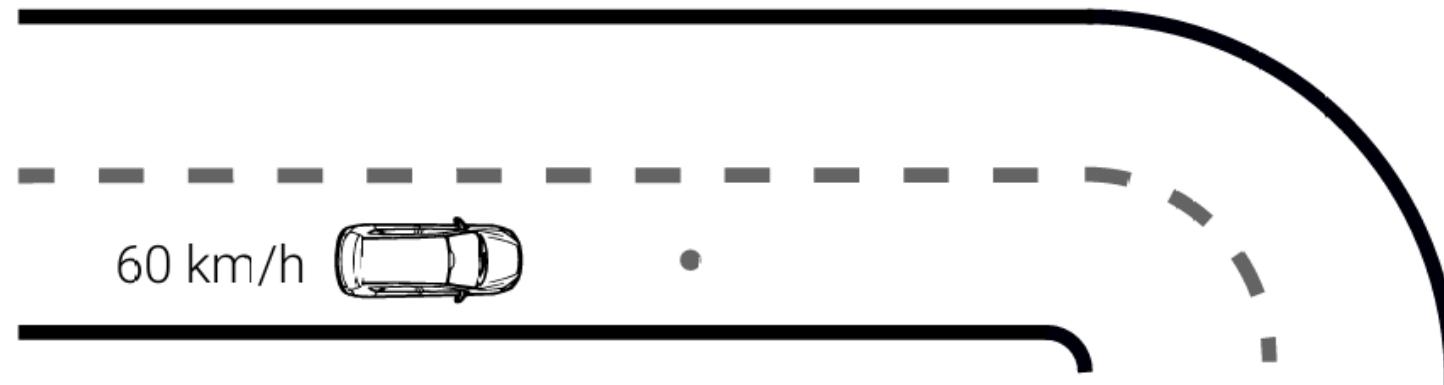


Model Predictive Control

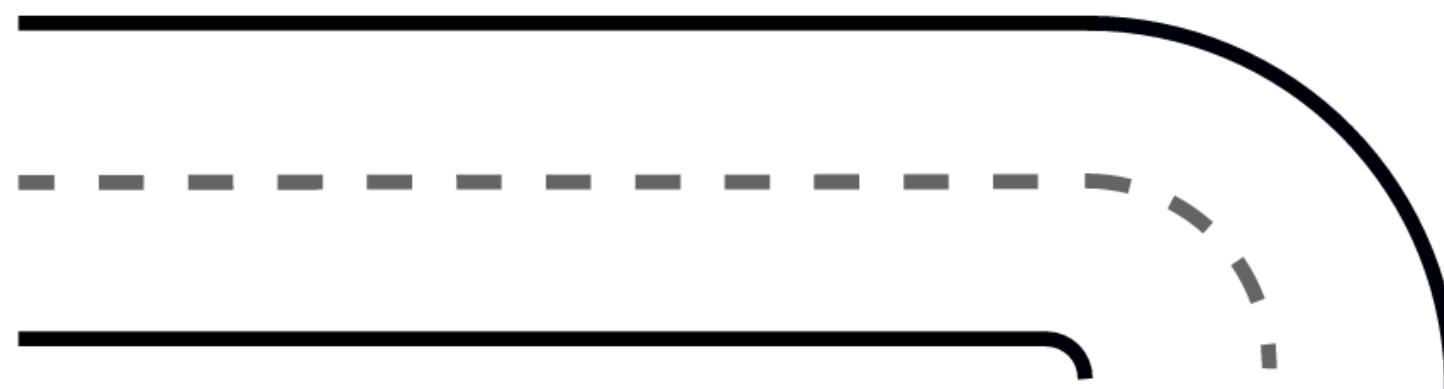


# Model Predictive Control

PID Control / Path Tracking

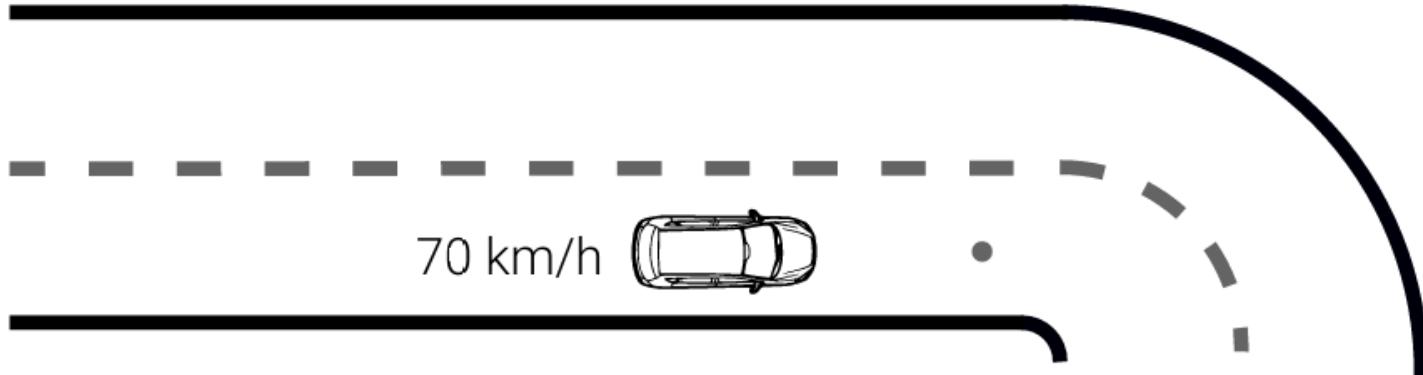


Model Predictive Control

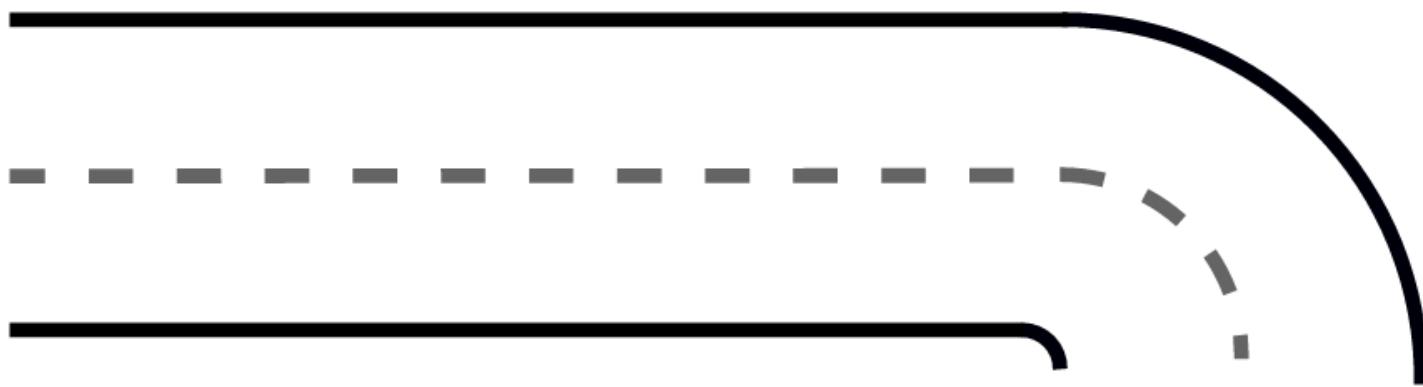


# Model Predictive Control

PID Control / Path Tracking

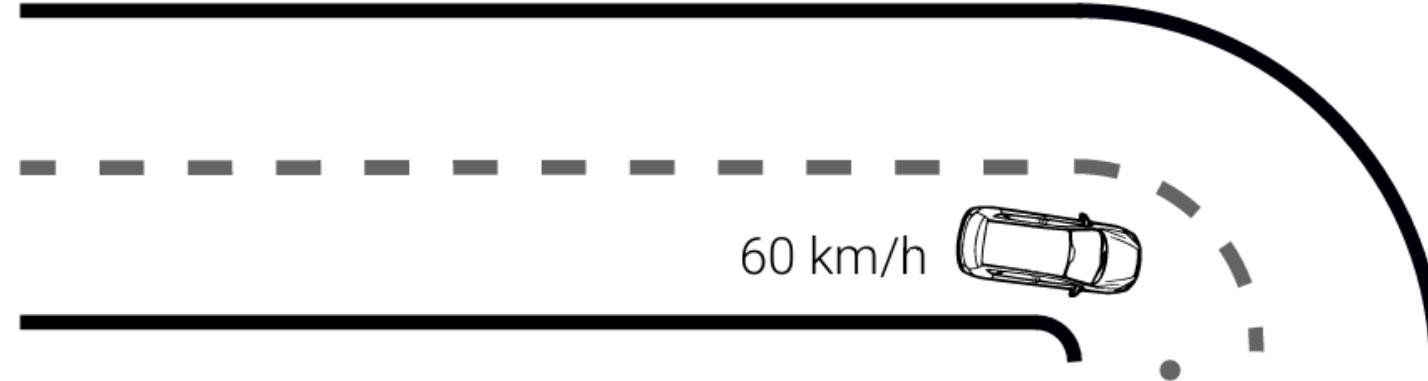


Model Predictive Control

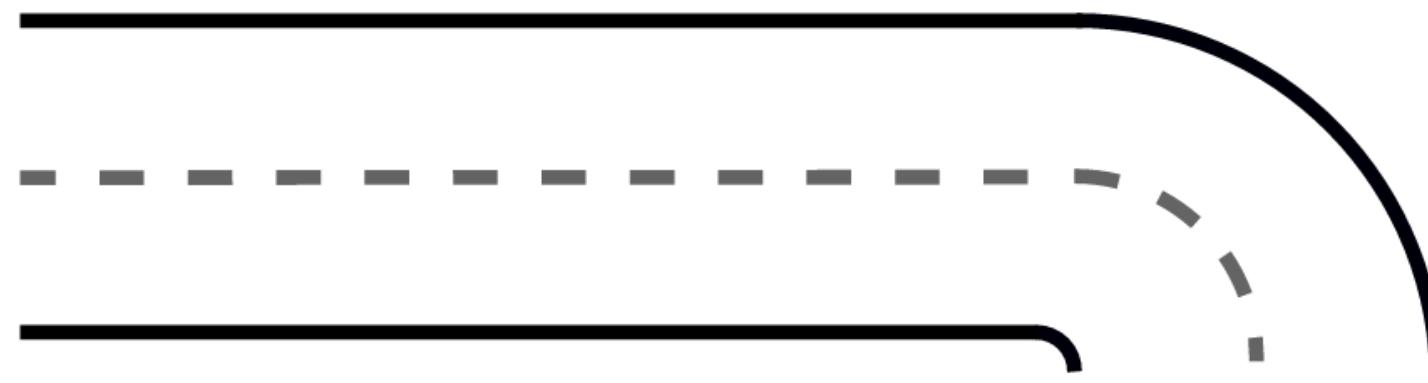


# Model Predictive Control

PID Control / Path Tracking

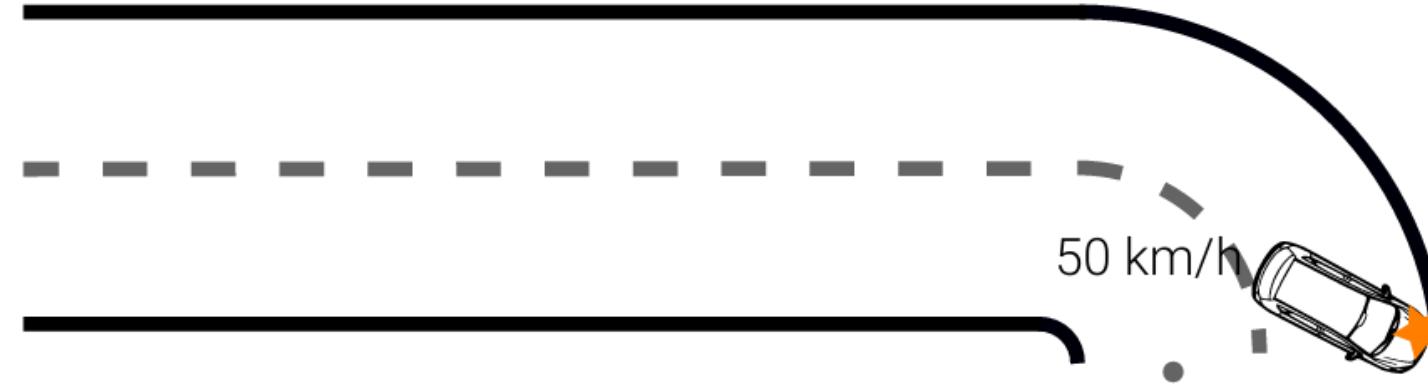


Model Predictive Control

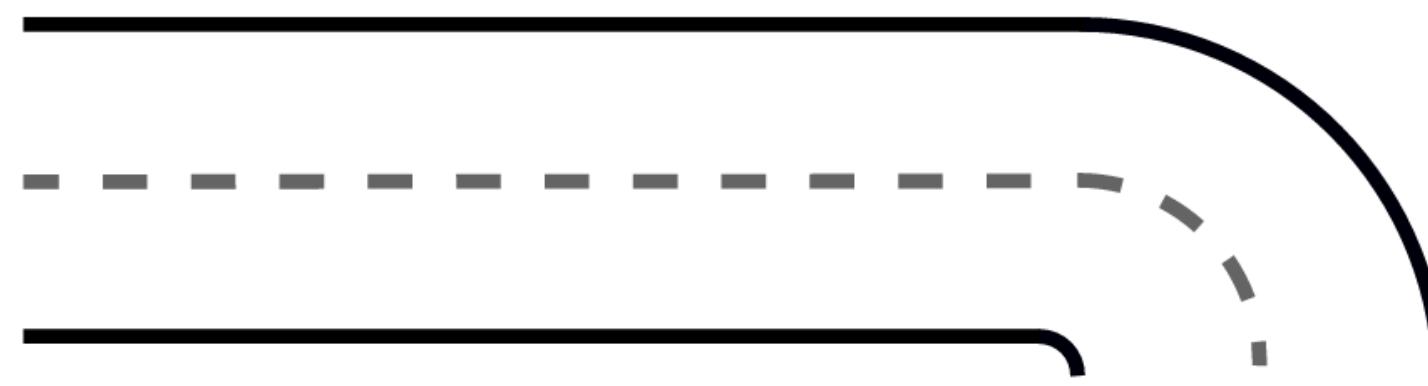


# Model Predictive Control

PID Control / Path Tracking

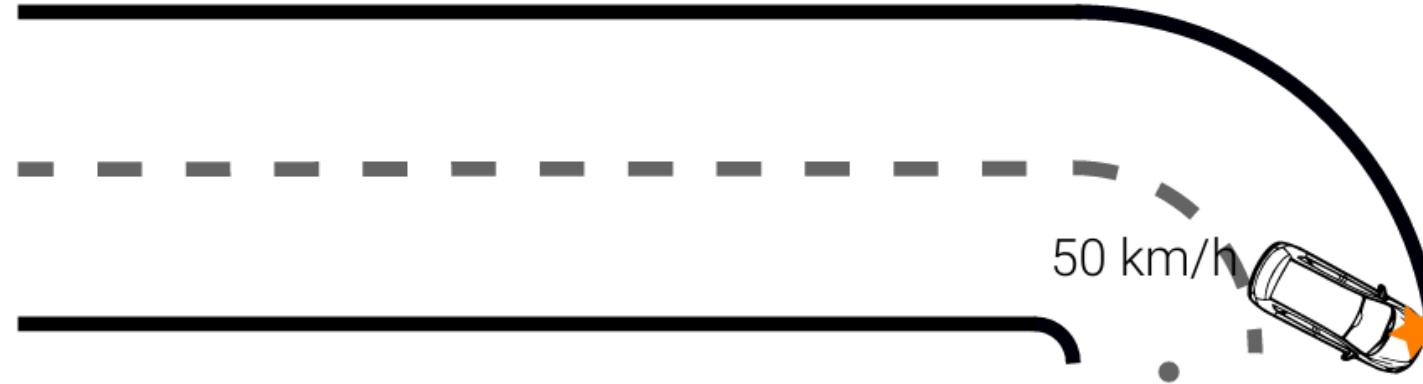


Model Predictive Control

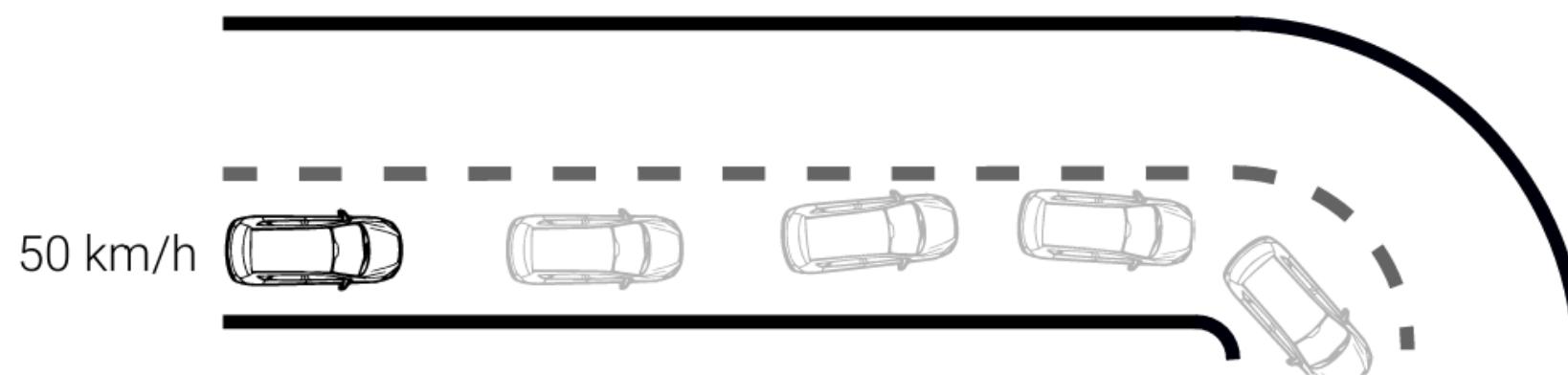


# Model Predictive Control

PID Control / Path Tracking

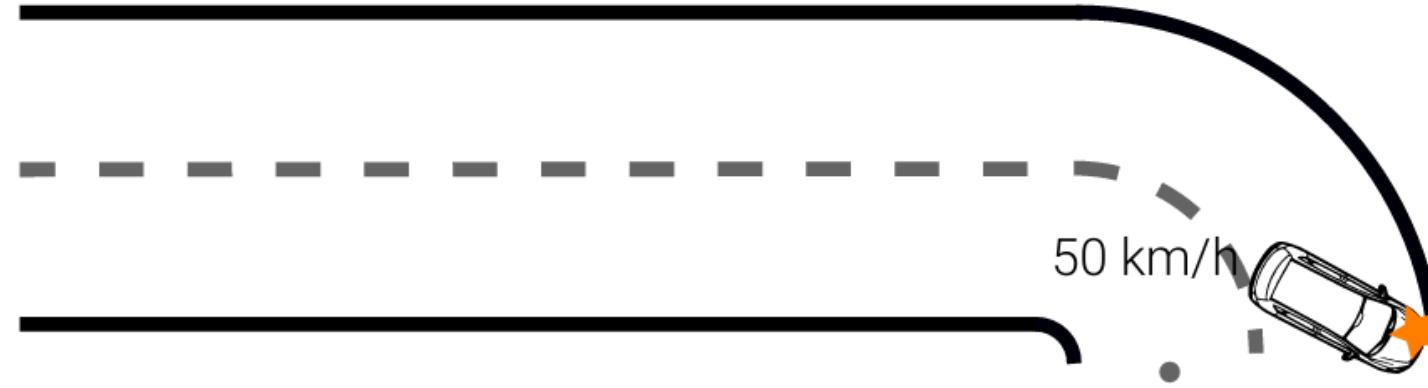


Model Predictive Control

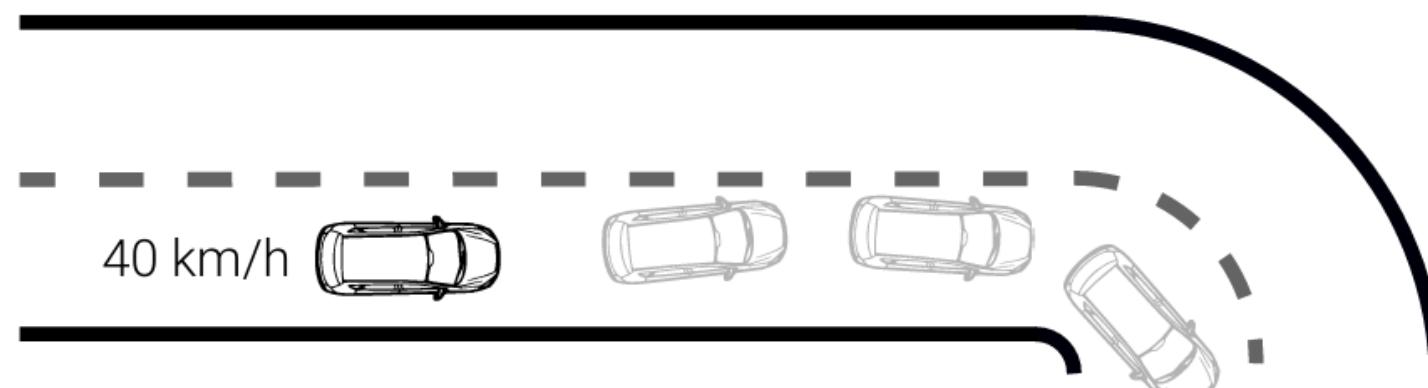


# Model Predictive Control

## PID Control / Path Tracking

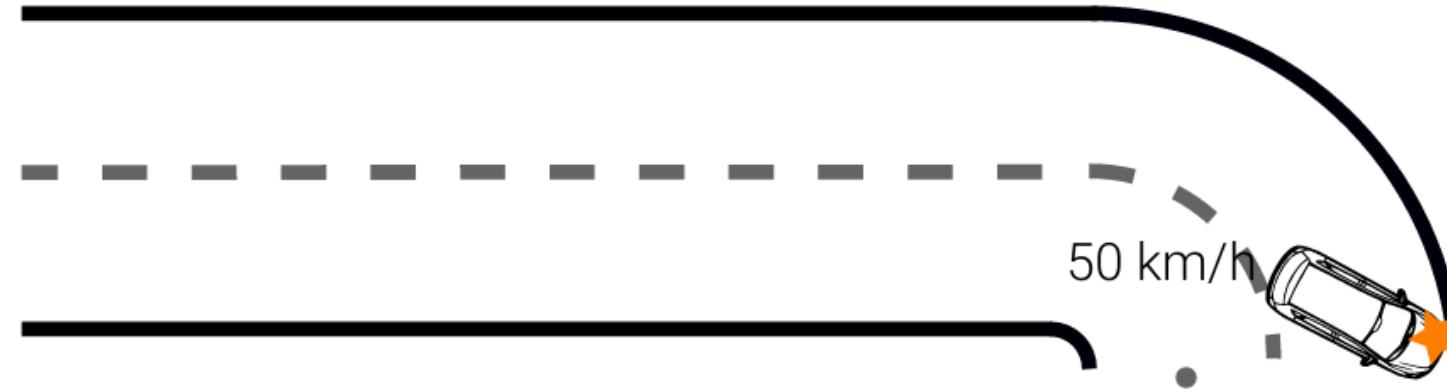


## Model Predictive Control

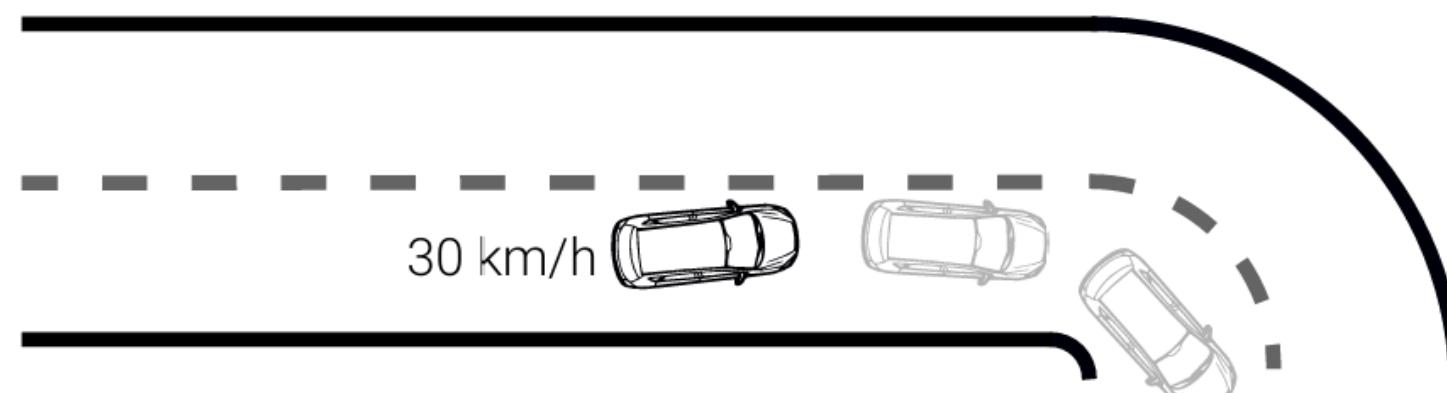


# Model Predictive Control

## PID Control / Path Tracking

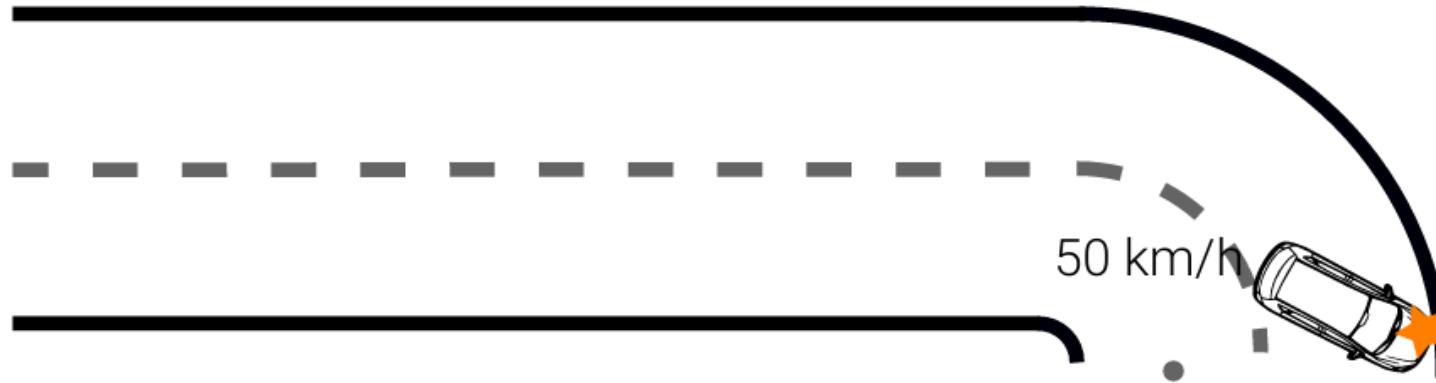


## Model Predictive Control

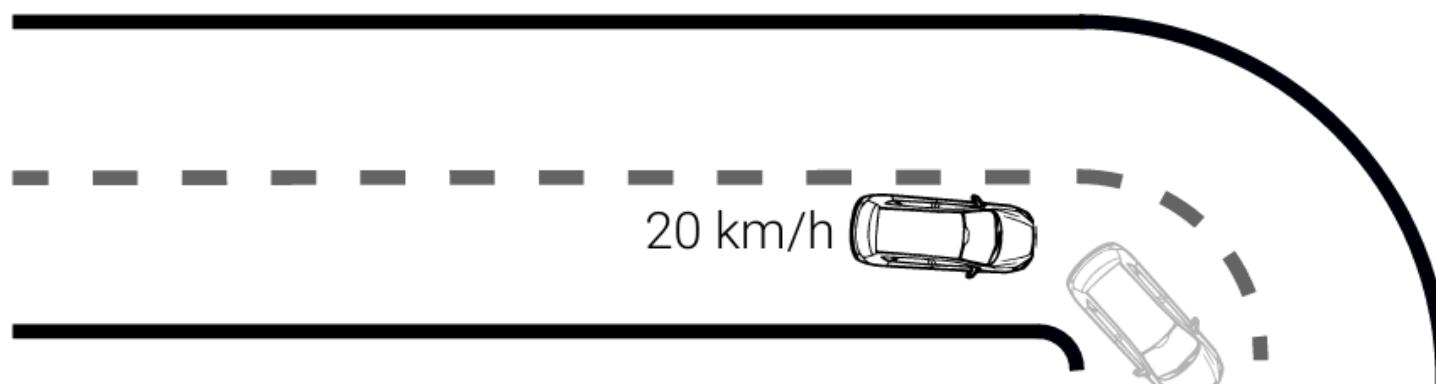


# Model Predictive Control

PID Control / Path Tracking

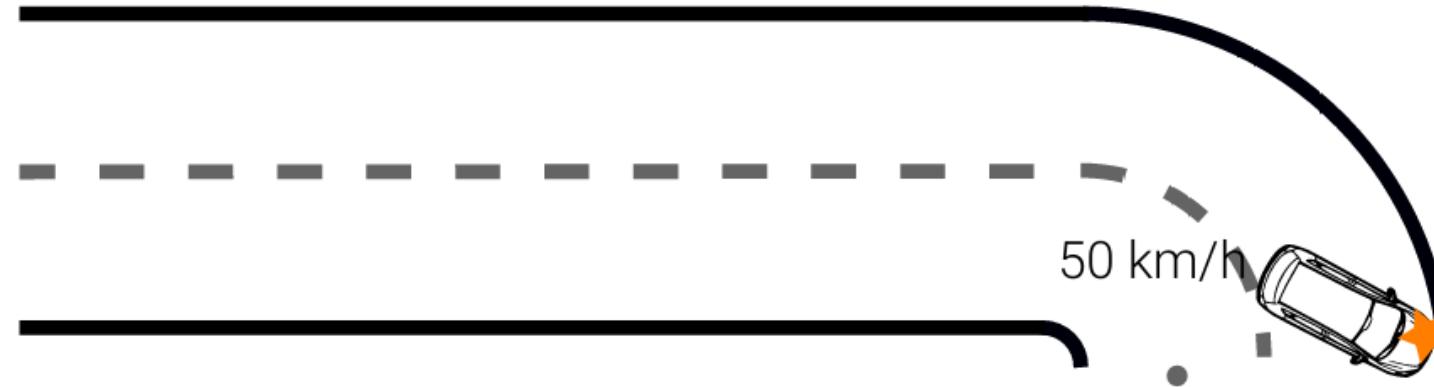


Model Predictive Control

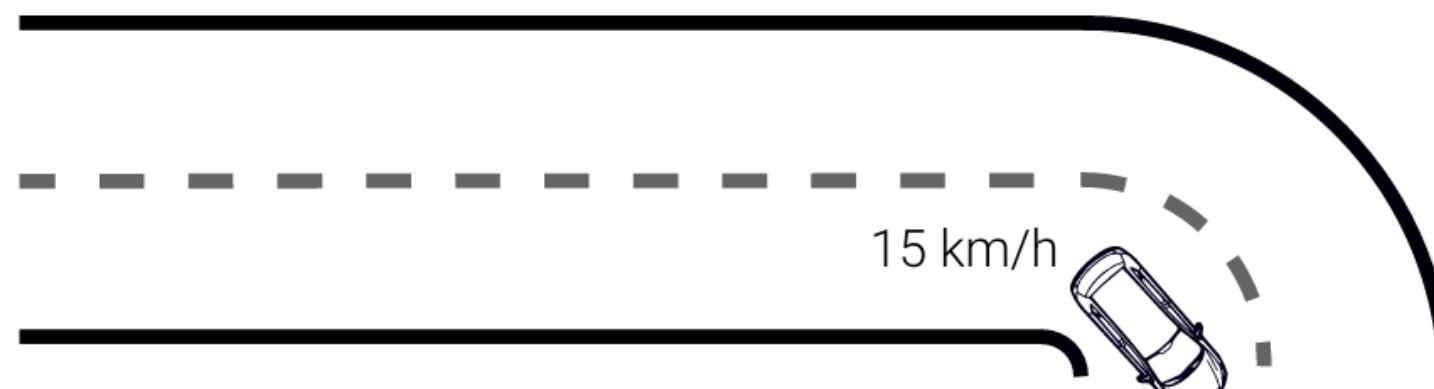


# Model Predictive Control

PID Control / Path Tracking



Model Predictive Control



# Summary

- ▶ Open-loop controllers cannot handle unknown disturbances
- ▶ In practice, we thus require closed-loop control with sensor feedback
- ▶ Black box controllers don't require knowledge about the process
- ▶ Most popular black box controller: PID controller
- ▶ Geometric controllers exploit geometric relationships for path tracking
- ▶ Optimal controllers use a vehicle model and optimize a cost function
- ▶ MPC is the most flexible and powerful approach
- ▶ However, MPC requires solving an optimization problem at every time step