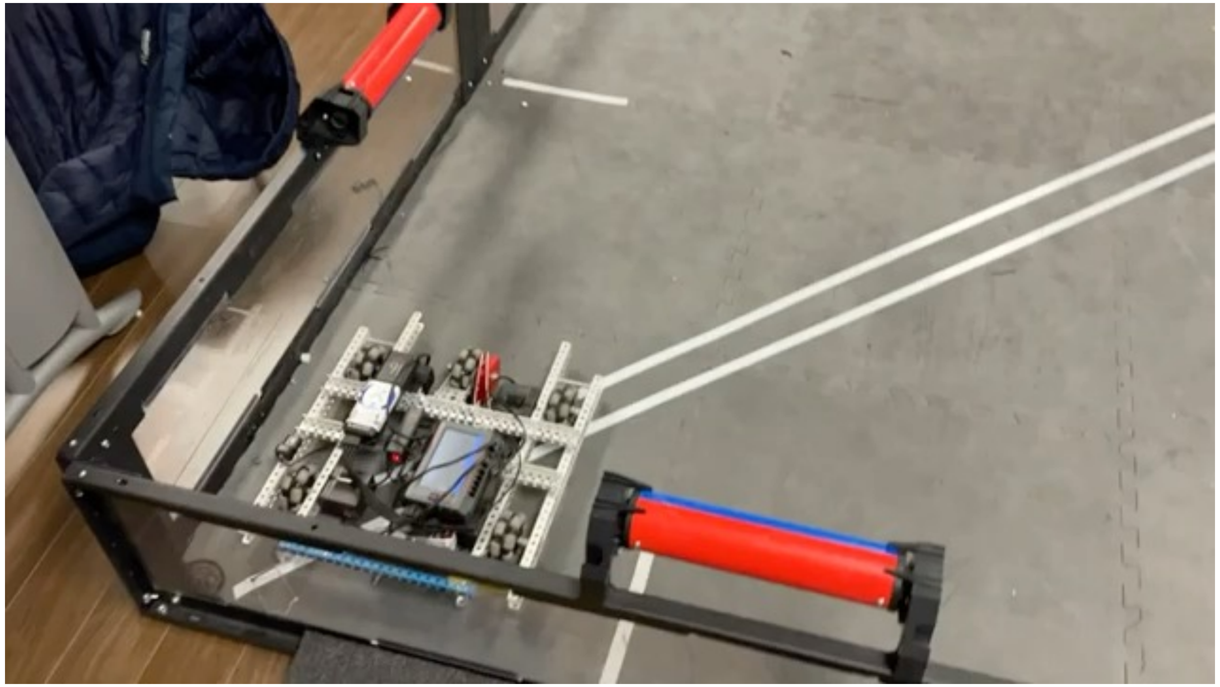
Autonomous Robotics
Winter 2025

Abhishek Gupta

TAs: Carolina Higuera, Entong Su, Bernie Zhu



Class Outline

State Estimation

- Robotic System Design
- Filtering
- Localization
- SLAM

Control

- Feedback Control
- PID Control
- MPC
- LQR

Planning

- Search
- Heuristic Search
- Motion Planning
- Lazy Search

Learning

- Imitation Learning
- Policy Gradient
- Actor-Critic
- Model-Based RL

Logistics

- HW3 out now
- Reading discussions due Wed Feb 12

- Post questions, discuss any issues you are having on Ed.
- Students with **no** access to 002, e-mail us with your student ID.
- Students that have not been added to the class, email abhgupta@cs.washington.edu with the subject-line "Waitlisted for CSE478"

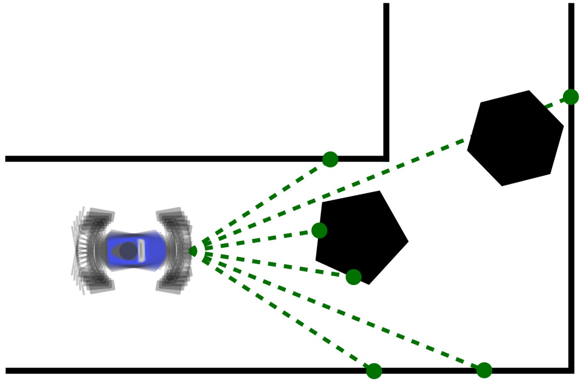
What is bang-bang control?

0

- choose control proportional to error — 0%
- choose control based on the sign of error — 0%
- choose control based on squared error — 0%
- None of the above — 0%

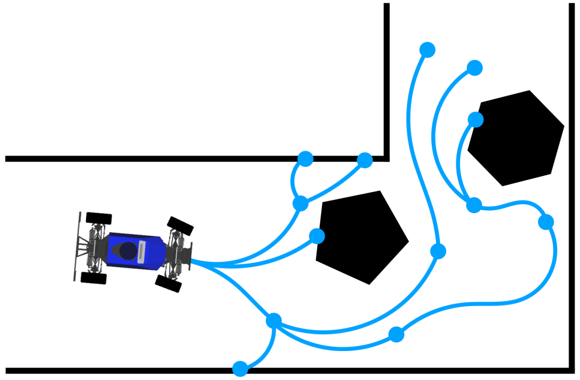
Recap

The Sense-Plan-Act Paradigm



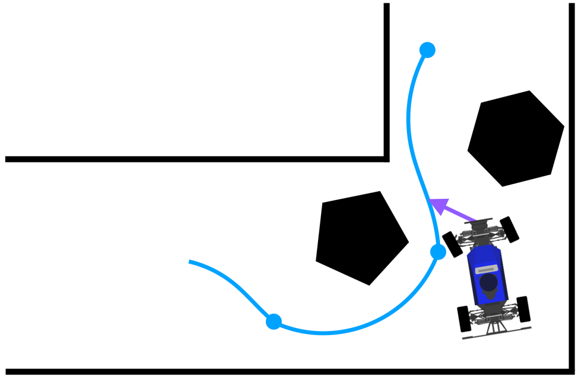
Estimate
robot state

Solved over last 3 weeks



Plan sequence of
motions

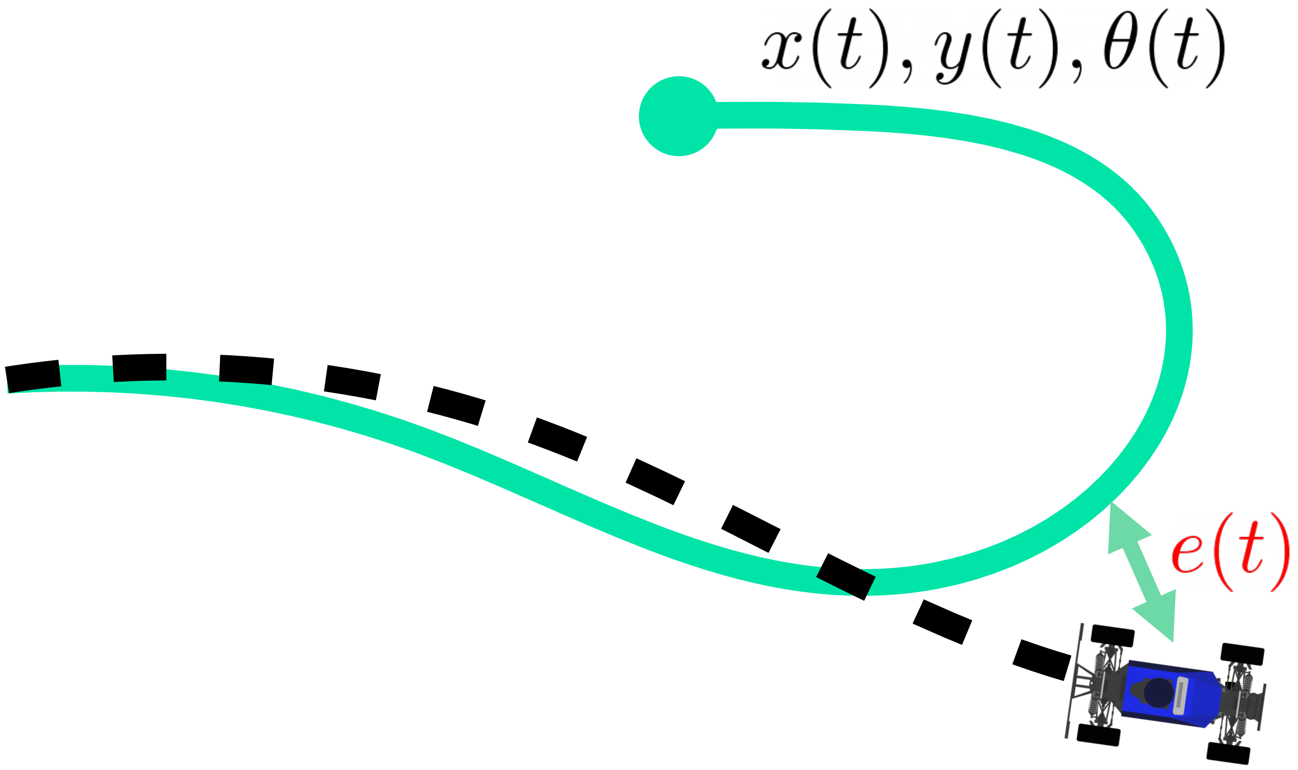
Assume to be solved for now



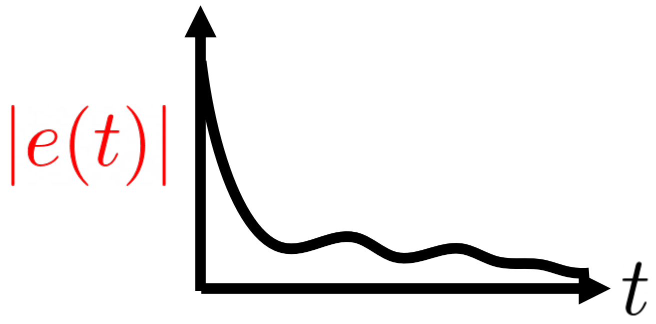
Control robot to
follow plan

??

Feedback Control



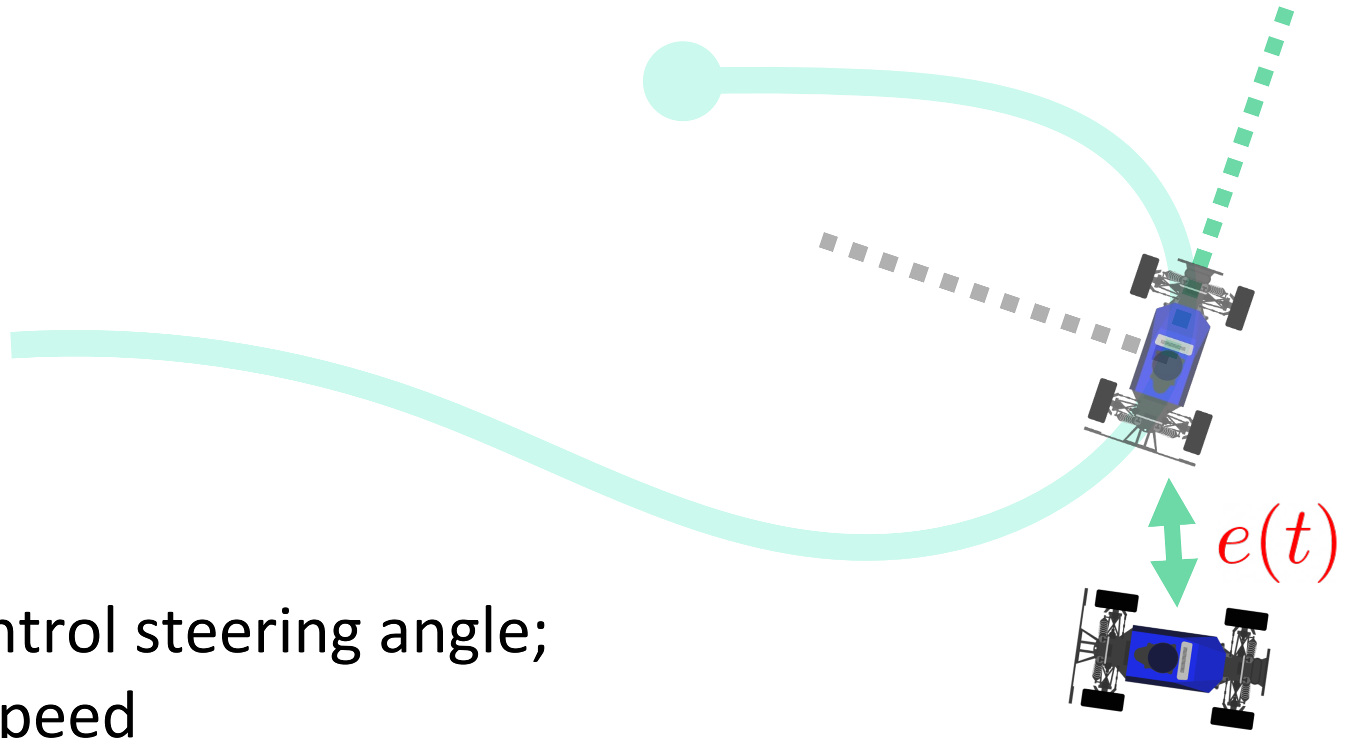
1. Measure error between reference and current state.
2. Take actions to minimize this error.



Controller Design Decisions

1. Get a reference path/trajectory to track
2. Pick a reference state from the reference path/trajectory
3. Compute error to reference state
4. Compute control law to minimize error

Step 4: Compute control law



We will only control steering angle;
fixed constant speed
As a result, no real control for along-track error
Some control laws will only minimize cross-track error, others will also minimize heading

$$u = K(e)$$

Step 4: Compute control law

Compute control action based on instantaneous error

$$u = K(\mathbf{x}, e)$$

control state error

(steering angle, speed)

Apply control action, robot moves a bit, compute new error, repeat

Different laws have different trade-offs

Different Control Laws

Proportional-integral-derivative (PID) control

Pure-pursuit control

Model-predictive control (MPC)

Linear-quadratic regulator (LQR)

And many many more!

Lecture Outline

Recap

↓

Bang-Bang Control

↓

PID Control

↓

Pure Pursuit

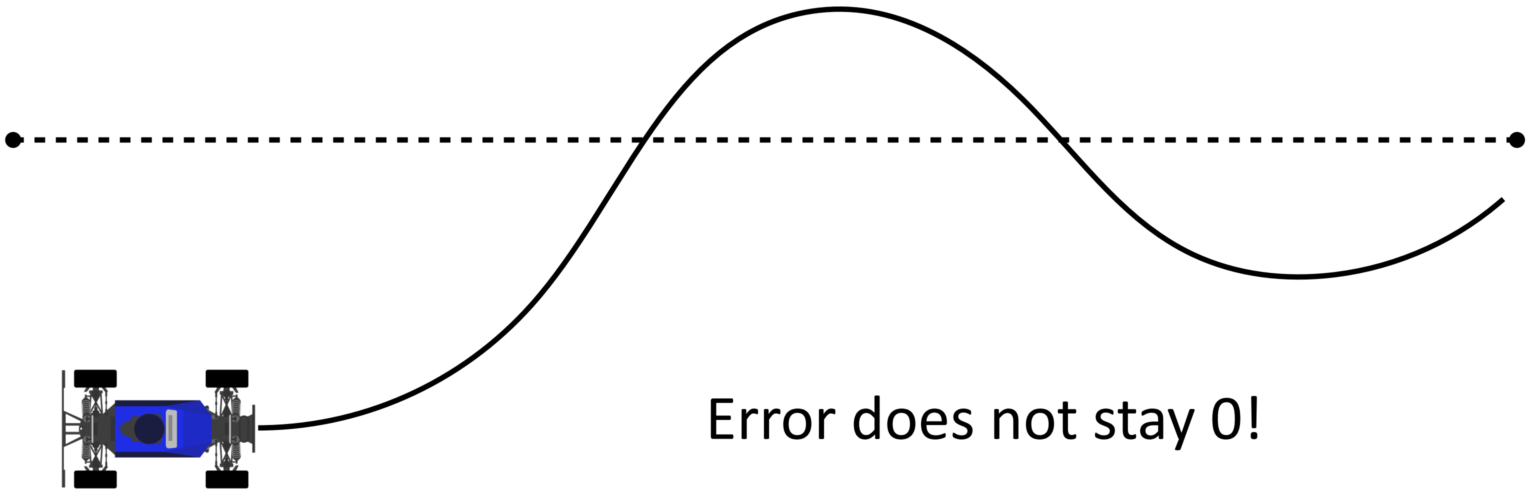
Bang-bang control

Simple control law - choose between hard left and hard right

$$u = \begin{cases} u_{max} & \text{if } e_{ct} < 0 \\ -u_{max} & \text{otherwise} \end{cases}$$

Bang-bang control

What happens when we run this control?



Error does not stay 0!

Need to adapt the magnitude of control proportional to the error …

This clearly sucks! How can we do better?

Lecture Outline

Recap

↓

Bang-Bang Control

↓

PID Control

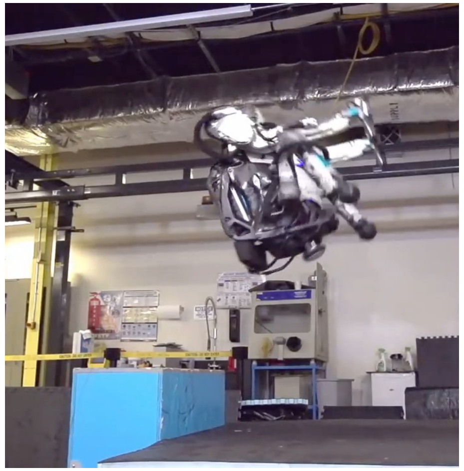
↓

Pure Pursuit

PID controllers

Used widely in industrial control from 1900s

Regulate temp, press, speed etc



Do not try this with PID!!!

PID control overview

Select a control law that tries to drive error to zero (and keep it there)

$$u = -\left(K_p e_{ct} + K_i \int e_{ct} \, dt + K_d \dot{e}_{ct} \right)$$

PROPORTIONAL (PRESENT) — **INTEGRAL** (PAST) — **DERIVATIVE** (FUTURE)

PID Intuition

$$u = -\left(K_p e_{\mathrm{ct}} + K_i \int e_{\mathrm{ct}} \, dt + K_d \dot{e}_{\mathrm{ct}} \right)$$

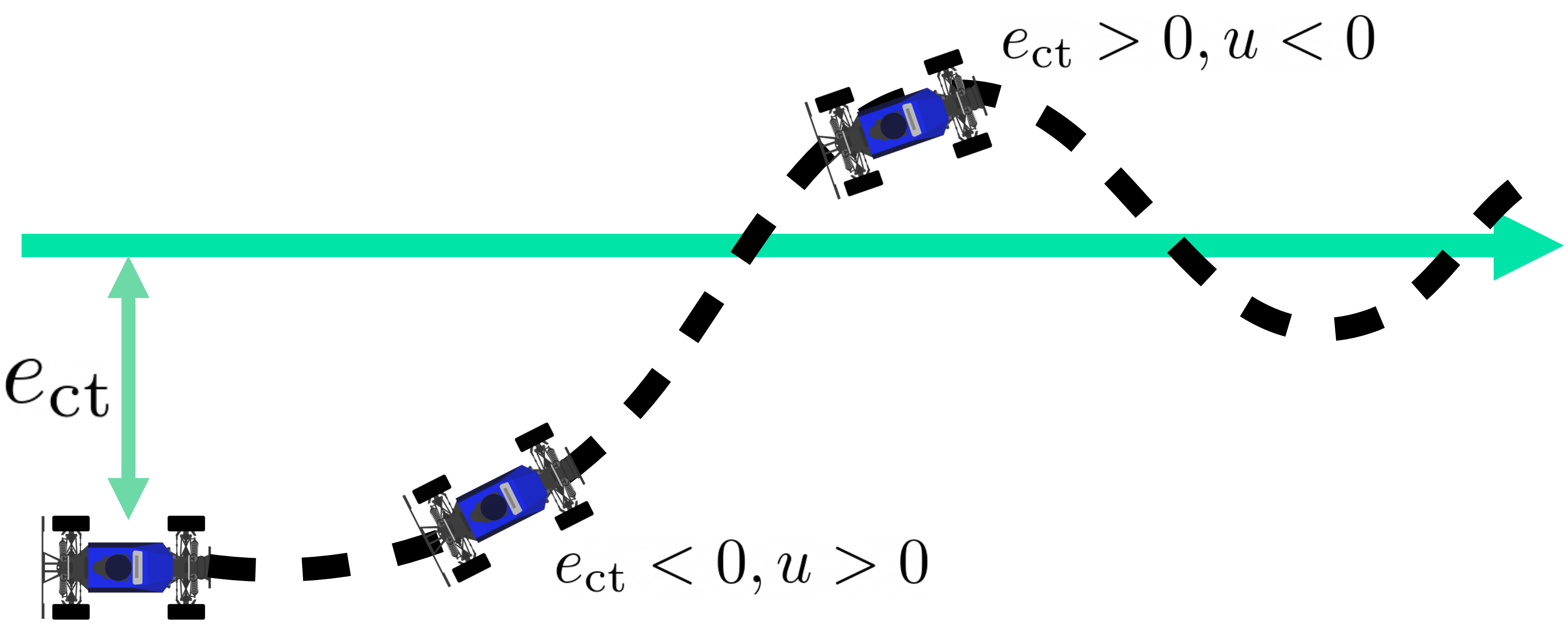
PROPORTIONAL	**INTEGRAL**	**DERIVATIVE**
(PRESENT)	(PAST)	(FUTURE)

Proportional: minimize the current error!

Integral: if I’m accumulating error, try harder!

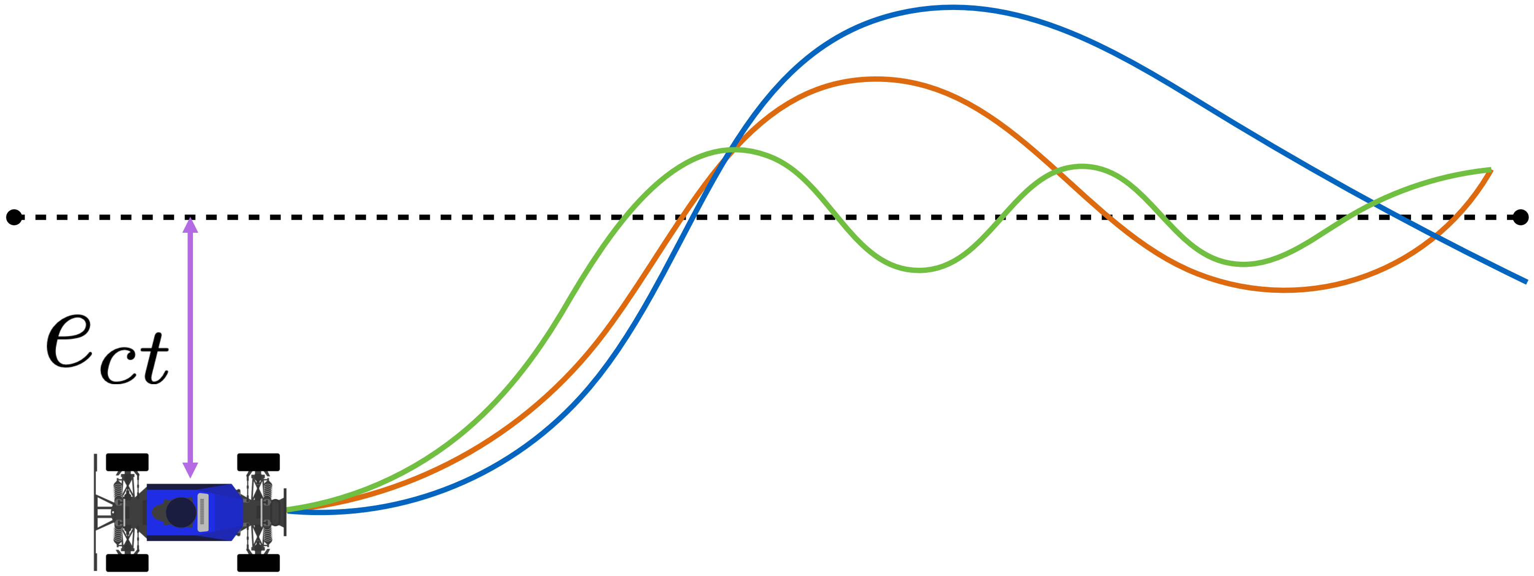
Derivative: if I’m going to overshoot, slow down!

Proportional Control



$$u = -K_p e_{\text{ct}}$$

The proportional gain matters!

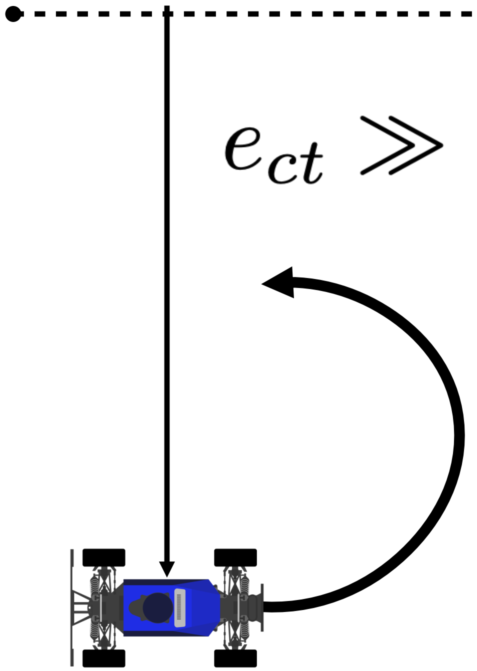


What happens when gain is low?

What happens when gain is high?

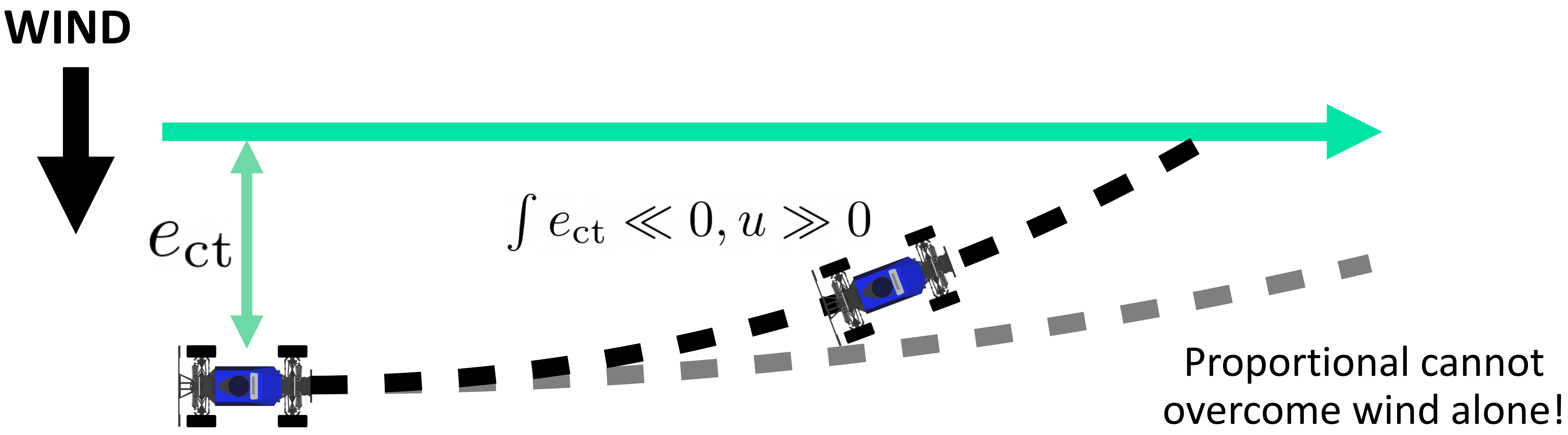
Proportional term

What happens when gain is too high?



$$e_{ct} \gg \frac{u_{max}}{K_p}$$

Proportional Integral (PI) Control

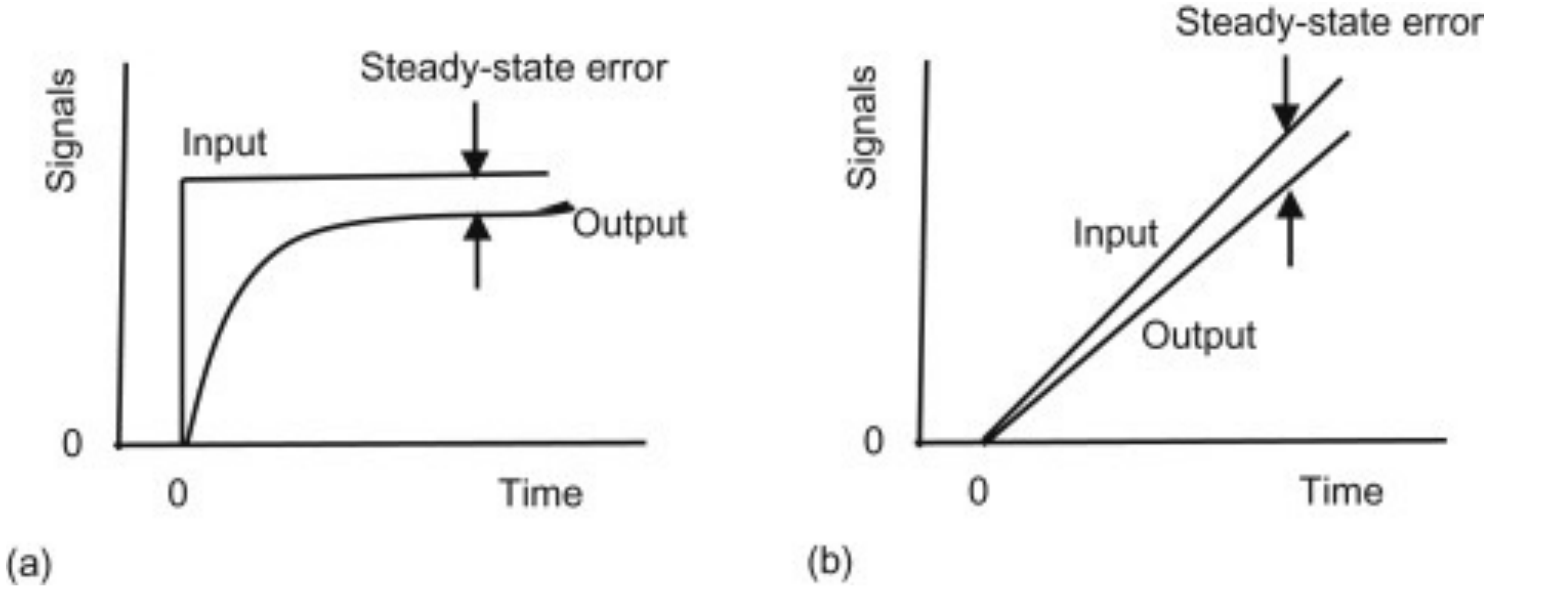


$$u = -\left(K_p e_{\text{ct}} + K_i \int e_{\text{ct}} dt\right)$$

Proportional Integral (PI) Control

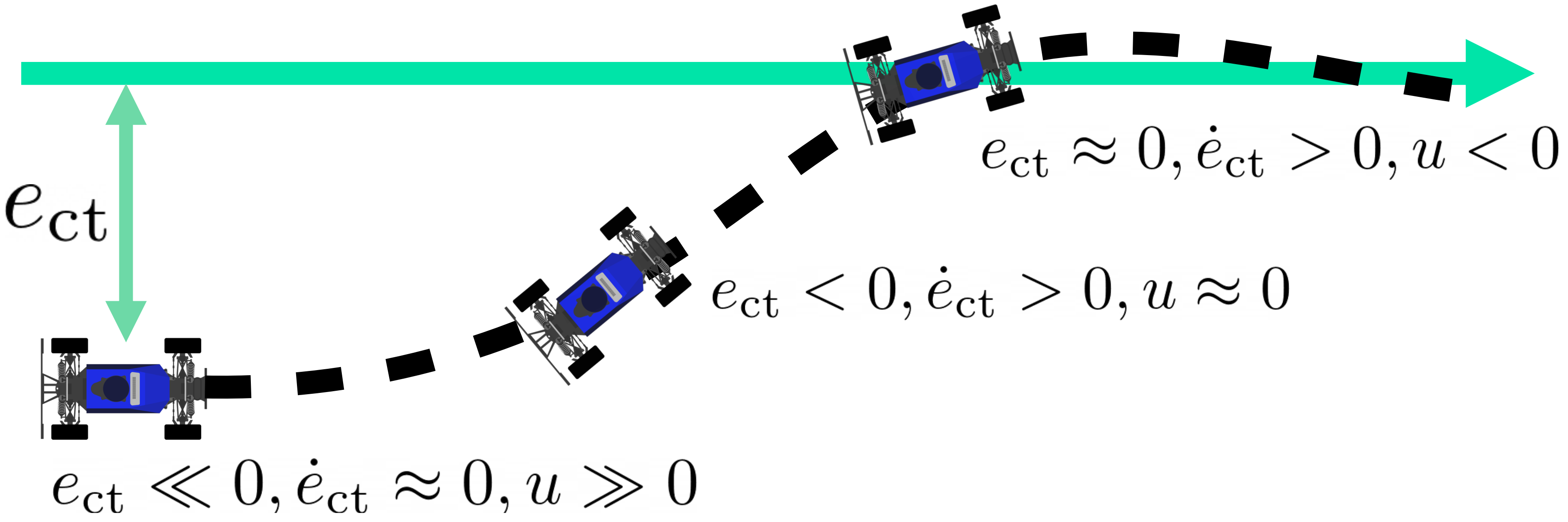
$$u = -\left(K_p e_{\mathrm{ct}} + K_i \int e_{\mathrm{ct}}\, dt\right)$$

Integral control gets rid of this term since the integral keeps growing



Proportional Derivative (PD) Control

Apply the brakes when moving too fast! → converge to the steady state



$$u = -\left(K_p e_{\text{ct}} + K_d \dot{e}_{\text{ct}}\right)$$

How do you evaluate the derivative term?

Terrible way: Numerically differentiate error. Why is this a bad idea?

Smart way: Analytically compute the derivative of the cross track error

$$e_{ct} = -\sin(\theta_{ref})(x - x_{ref}) + \cos(\theta_{ref})(y - y_{ref})$$

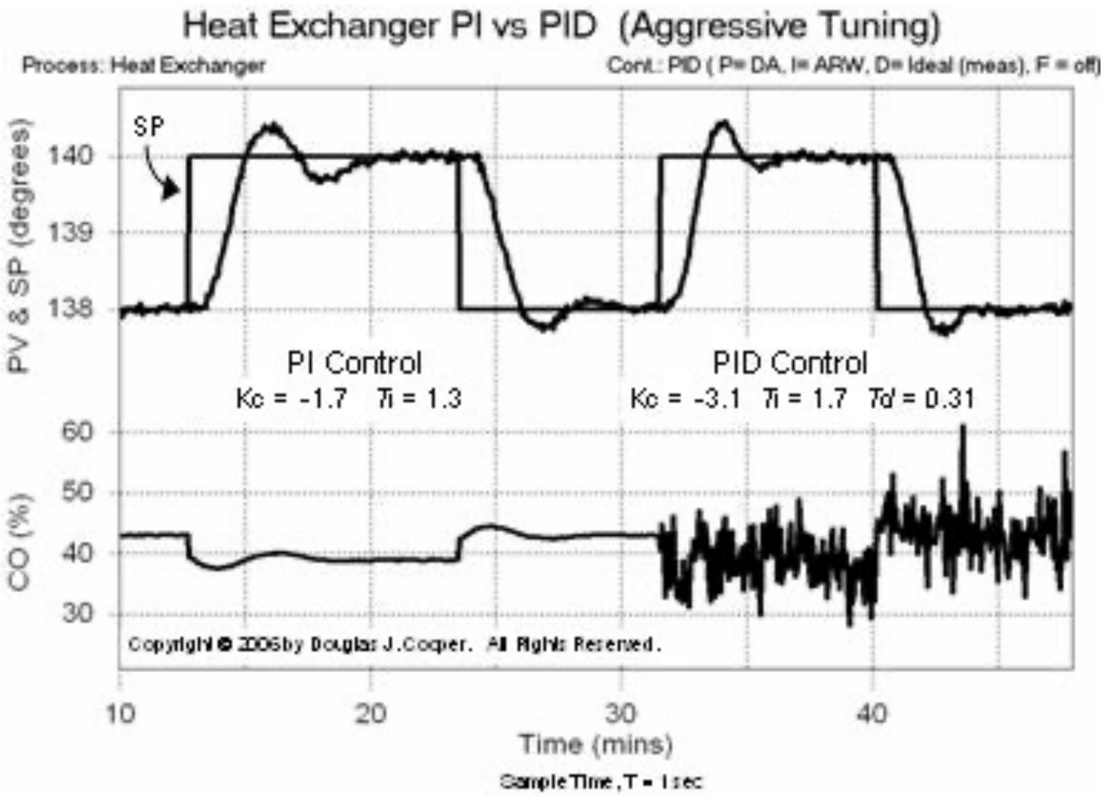
$$\begin{aligned}\dot{e}_{ct} &= -\sin(\theta_{ref})\dot{x} + \cos(\theta_{ref})\dot{y} \\ &= -\sin(\theta_{ref})V\cos(\theta) + \cos(\theta_{ref})V\sin(\theta) \\ &= V\sin(\theta - \theta_{ref}) = V\sin(\theta_e)\end{aligned}$$

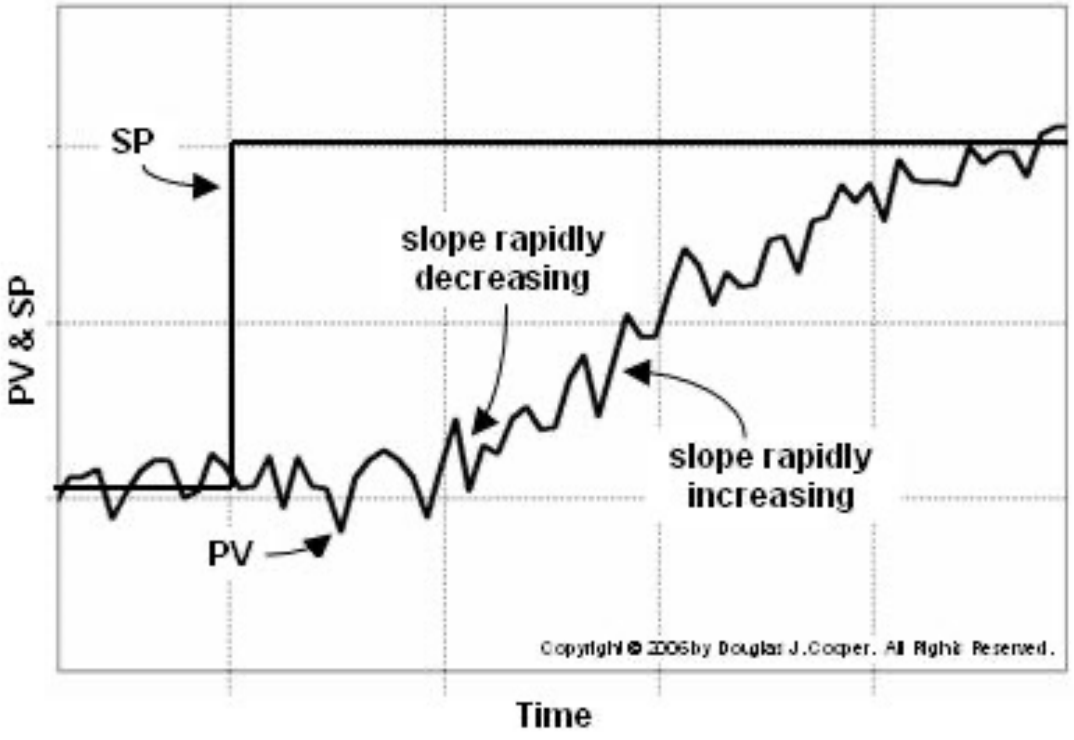
New control law! Penalize error in cross track and in heading

$$u = -\left(K_p e_{ct} + K_d V \sin\theta_e\right)$$

Challenges with using the derivative term

Noise can lead to wildly changing derivatives – leading to huge control variations





PID Intuition

$$u = -\left(K_p e_{\text{ct}} + K_i \int e_{\text{ct}} dt + K_d \dot{e}_{\text{ct}}\right)$$

PROPORTIONAL	INTEGRAL	DERIVATIVE
(PRESENT)	(PAST)	(FUTURE)

Proportional: minimize the current error!

Integral: if I’m accumulating error, try harder!

Derivative: if I’m going to overshoot, slow down!

Tuning PID controllers

$$u = -\left(K_p e_{\mathrm{ct}} + K_i \int e_{\mathrm{ct}} dt + K_d \dot{e}_{\mathrm{ct}} \right)$$

PROPORTIONAL
(PRESENT)

INTEGRAL
(PAST)

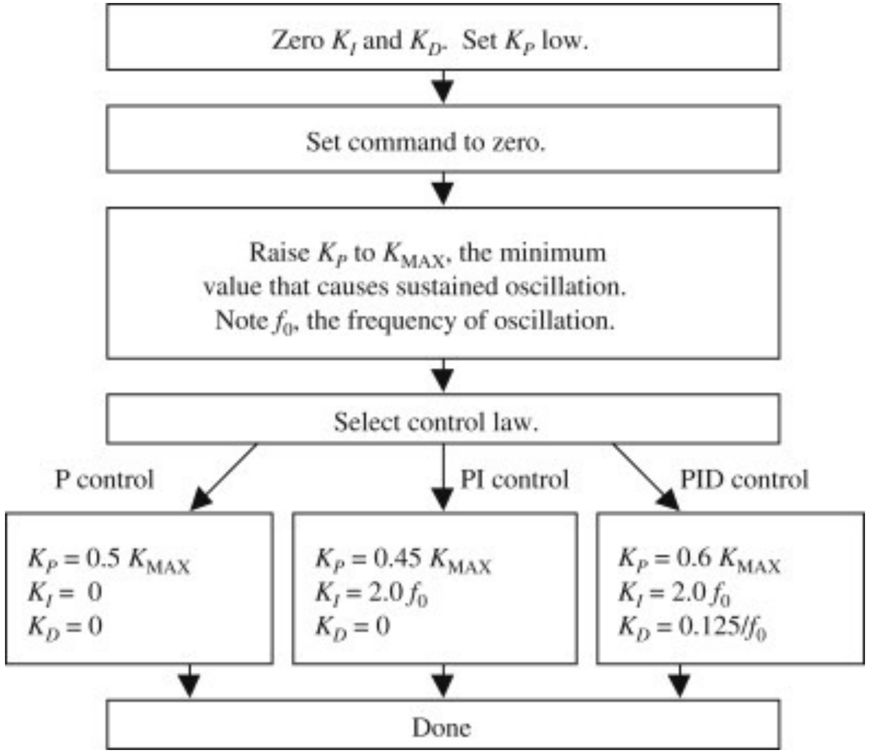
DERIVATIVE
(FUTURE)

How do you set the K_p, K_i, K_d constants for a particular system?

Tuning PID controllers: Ziegler-Nichols

Heuristic/empirical method for computing K_p, K_i, K_d

$$u = -\left(K_p e_{\mathrm{ct}} + K_i \int e_{\mathrm{ct}} dt + K_d \dot{e}_{\mathrm{ct}}\right)$$



See how the system responds to proportional gain

Adjust integral and proportional accordingly

Lecture Outline

Recap

↓

Bang-Bang Control

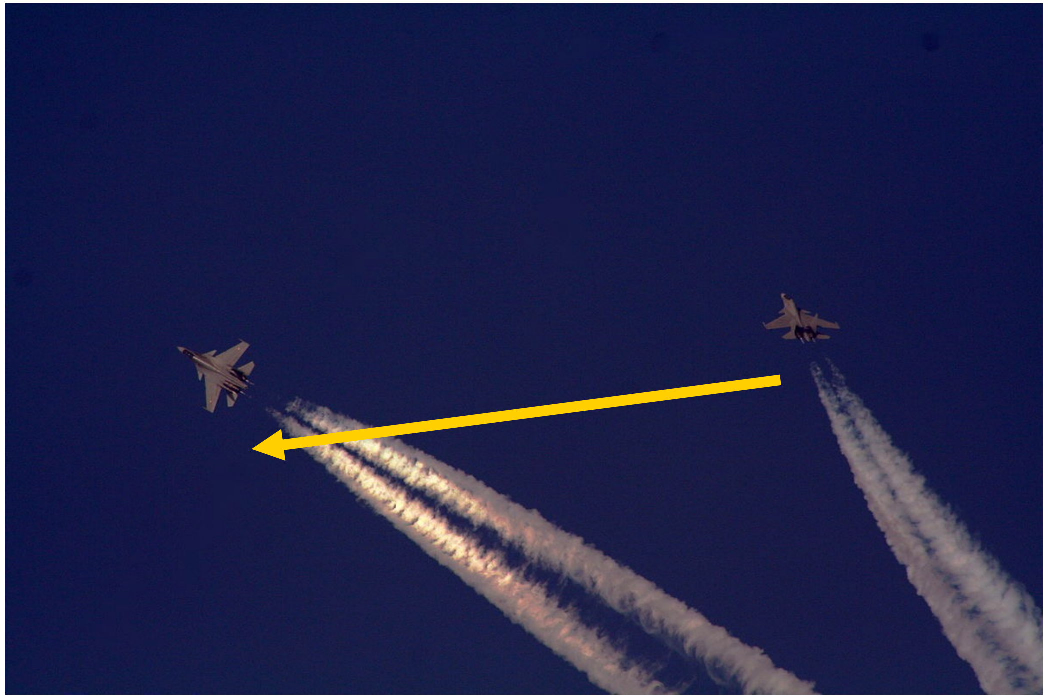
↓

PID Control

↓

Pure Pursuit

Pure Pursuit Control



Aerial combat in which aircraft pursues another aircraft by pointing its nose directly towards it

Similar to carrot on a stick!

Rationale: Controller should leverage model!

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

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

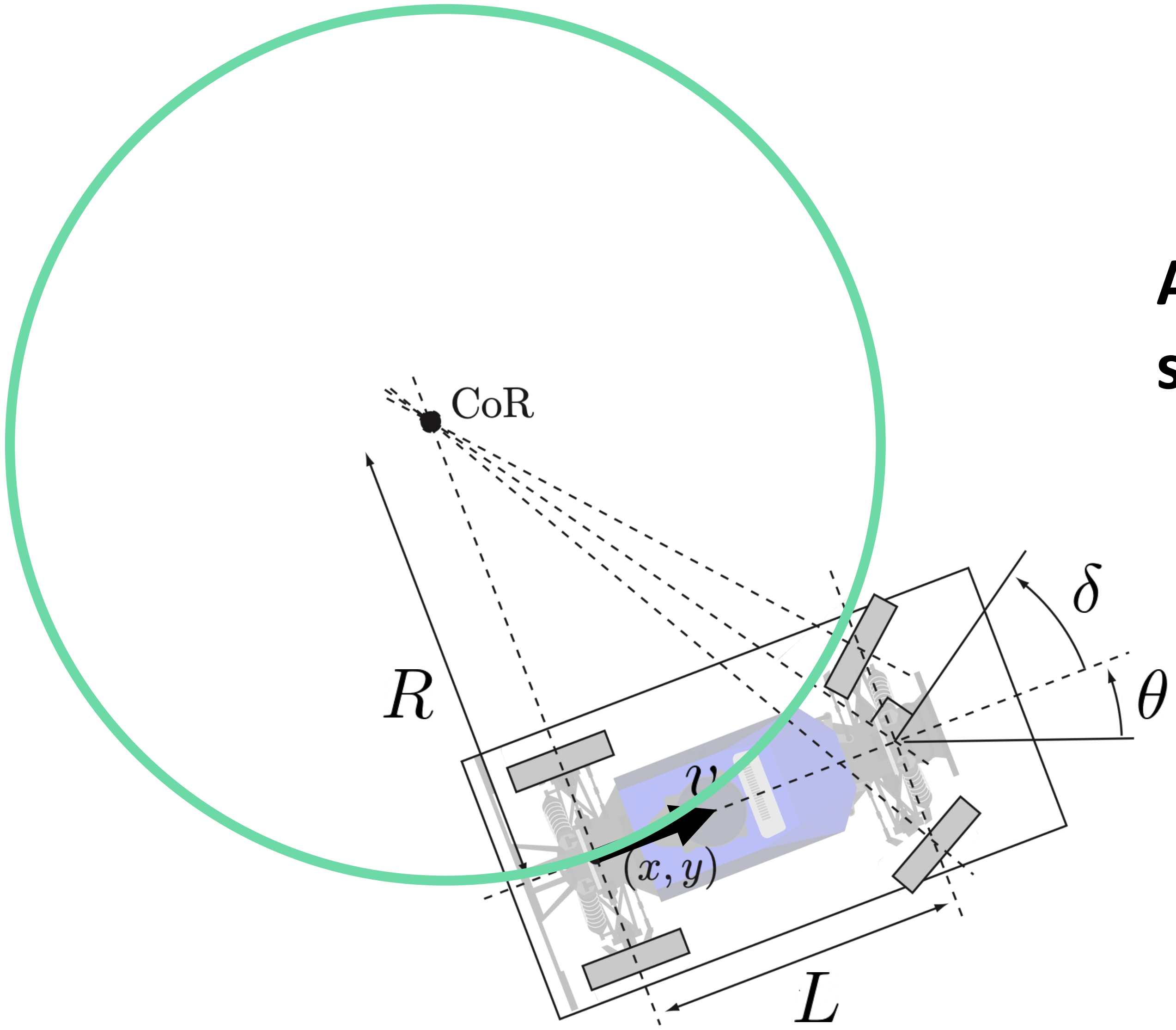
$$\dot{\theta} = \omega = \frac{v}{R} = \frac{v\tan\delta}{L}$$

PID control doesn't directly utilize the fact that we know the kinematic car model

Key Idea:

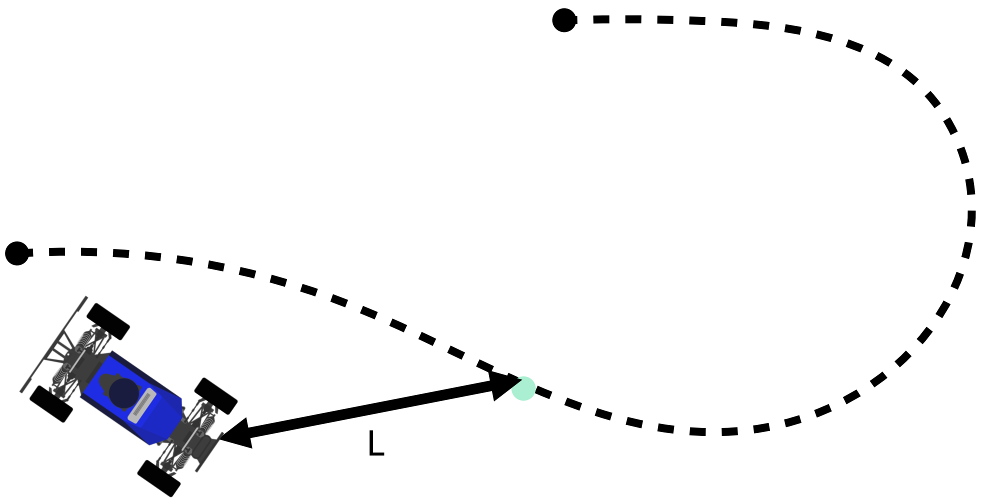
The car is always moving in a circular arc

Pure Pursuit Controller



Assume the car is moving with fixed steering angle

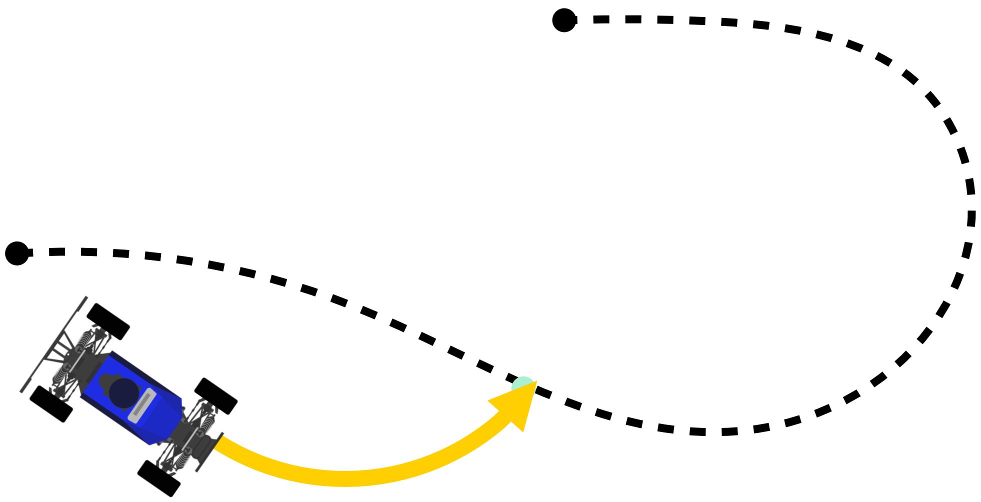
Consider a reference at a lookahead distance



$$\left\| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{ref} \\ y_{ref} \end{bmatrix} \right\| = L$$

Problem: Can we solve for a steering angle that guarantees that the car will pass through the reference?

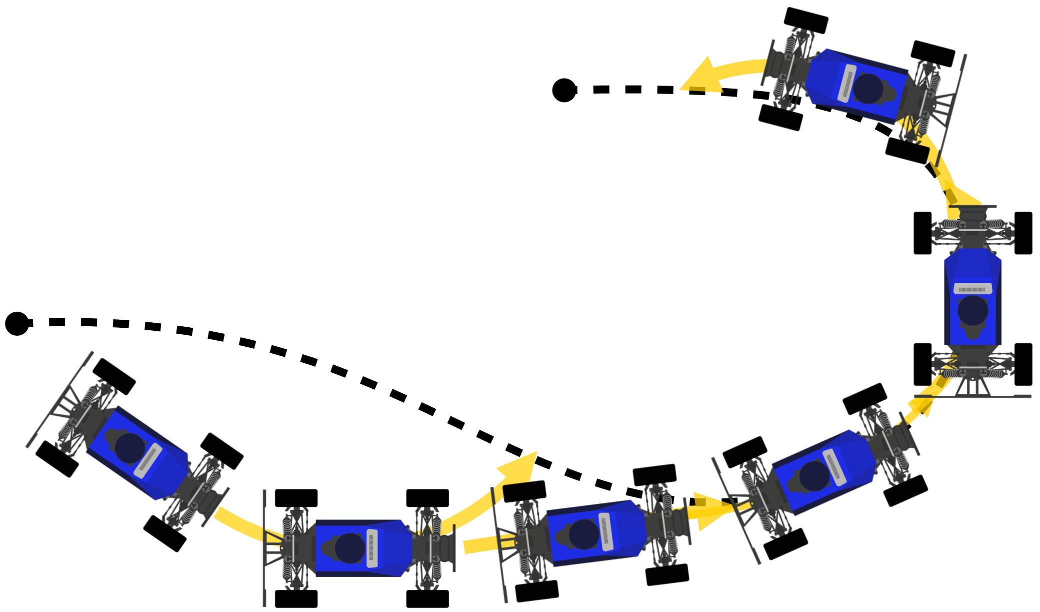
Solution: Compute a circular arc



We can always solve for a arc that passes through a lookahead point

Note: As the car moves forward, the point keeps moving

Pure pursuit: Keep chasing looakahead



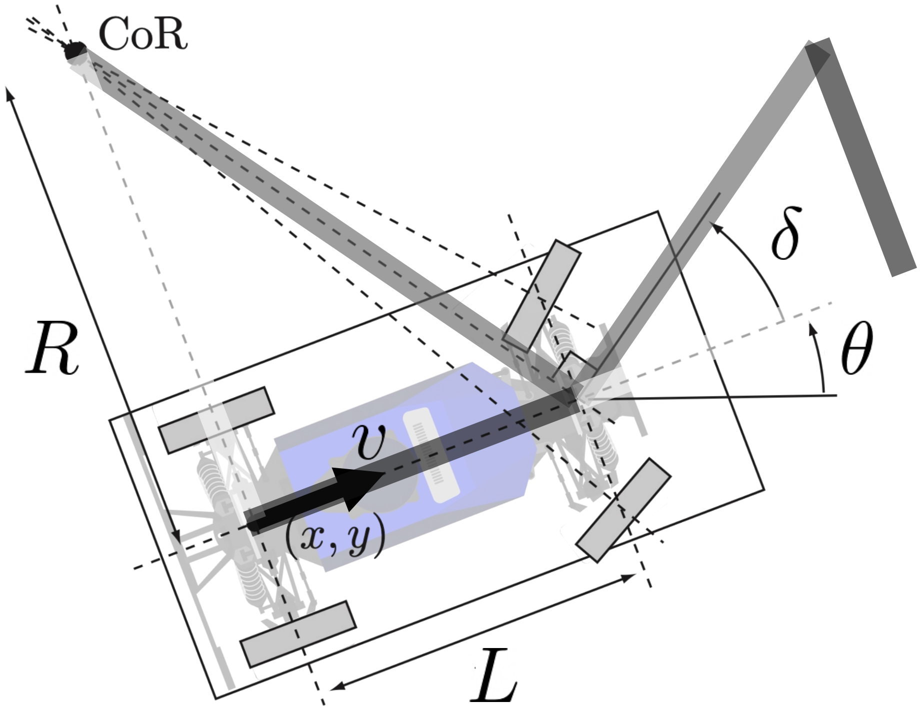
1. Find a lookahead and compute arc

2. Move along the arc

3. Go to step 1

Equations of Motion

RECALL



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

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

$$\dot{\theta} = \omega = \frac{v}{R} = \frac{v\tan\delta}{L}$$

$$\tan\delta = \frac{L}{R} \rightarrow R = \frac{L}{\tan\delta}$$

Kinematic Car Model

RECALL

$$\dot{x} = f(x, u)$$

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

X-COORDINATE

Y-COORDINATE

HEADING

$$\begin{bmatrix} v \\ \delta \end{bmatrix}$$

SPEED

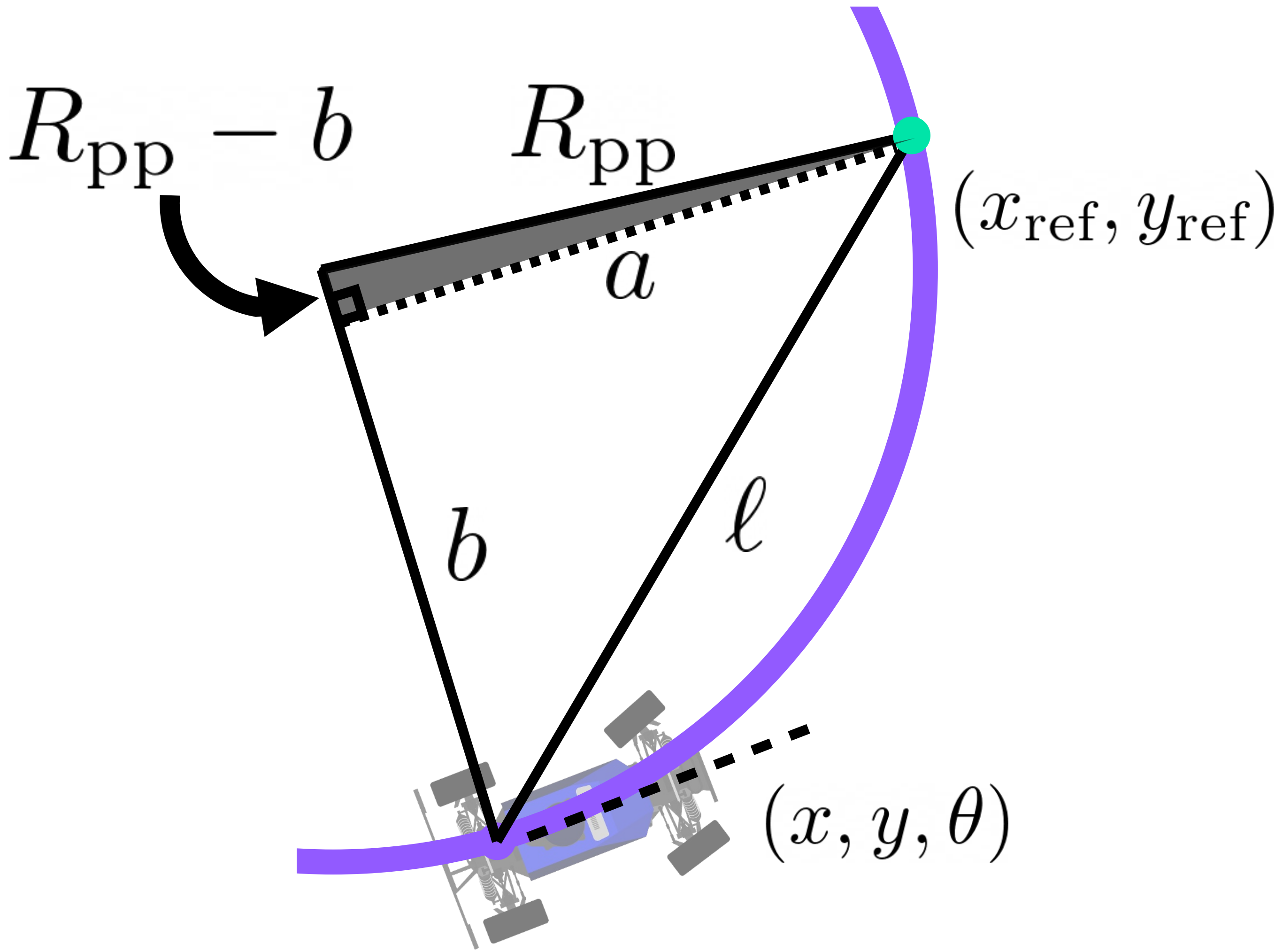
STEERING ANGLE

$$\begin{bmatrix} 1 \\ u \end{bmatrix}$$

Computing the Arc Radius

$$(R_{\text{pp}} - b)^2 + a^2 = R_{\text{pp}}^2$$

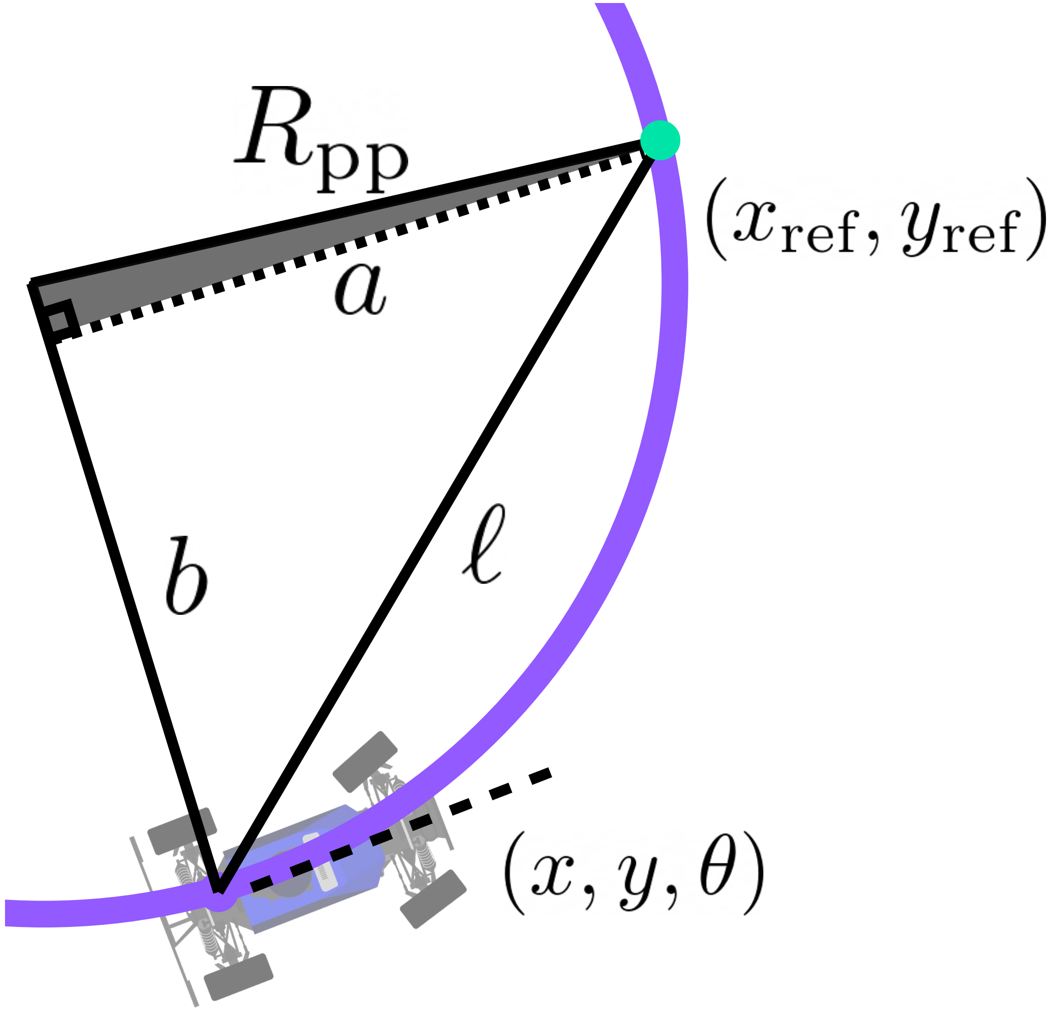
$$R_{\text{pp}} = \frac{a^2 + b^2}{2b}$$



Computing the Arc Radius

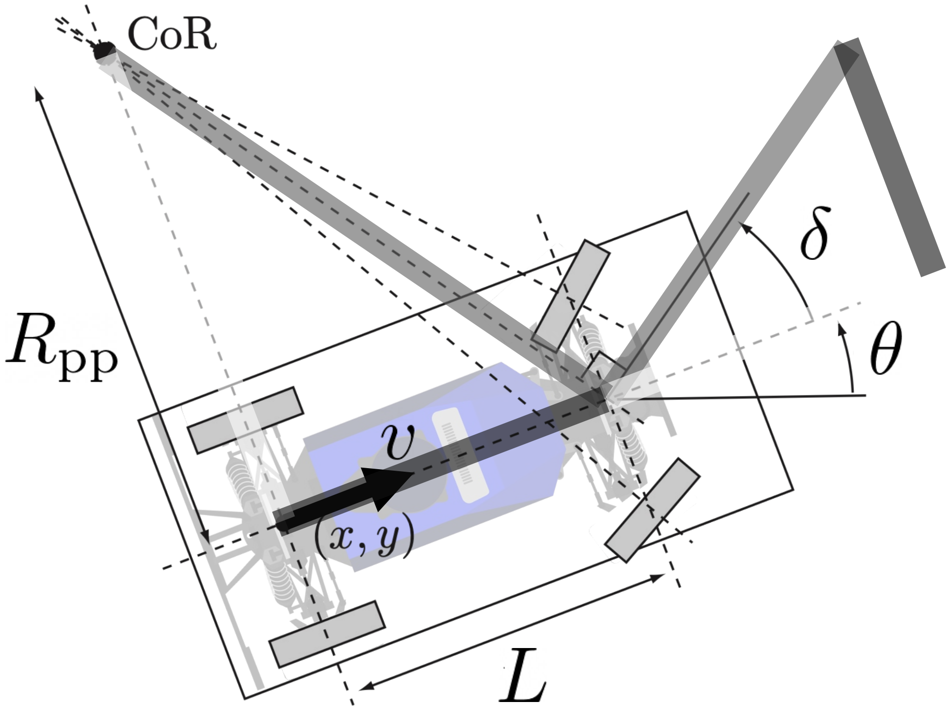
$$R_{\text{pp}} = \frac{a^2 + b^2}{2b}$$

$$\begin{bmatrix} a \\ b \end{bmatrix} = R(-\theta)\left(\begin{bmatrix} x_{\text{ref}} \\ y_{\text{ref}} \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix}\right)$$



Different than cross-track error (this is ref. position in robot frame; vice versa for cross-track error)

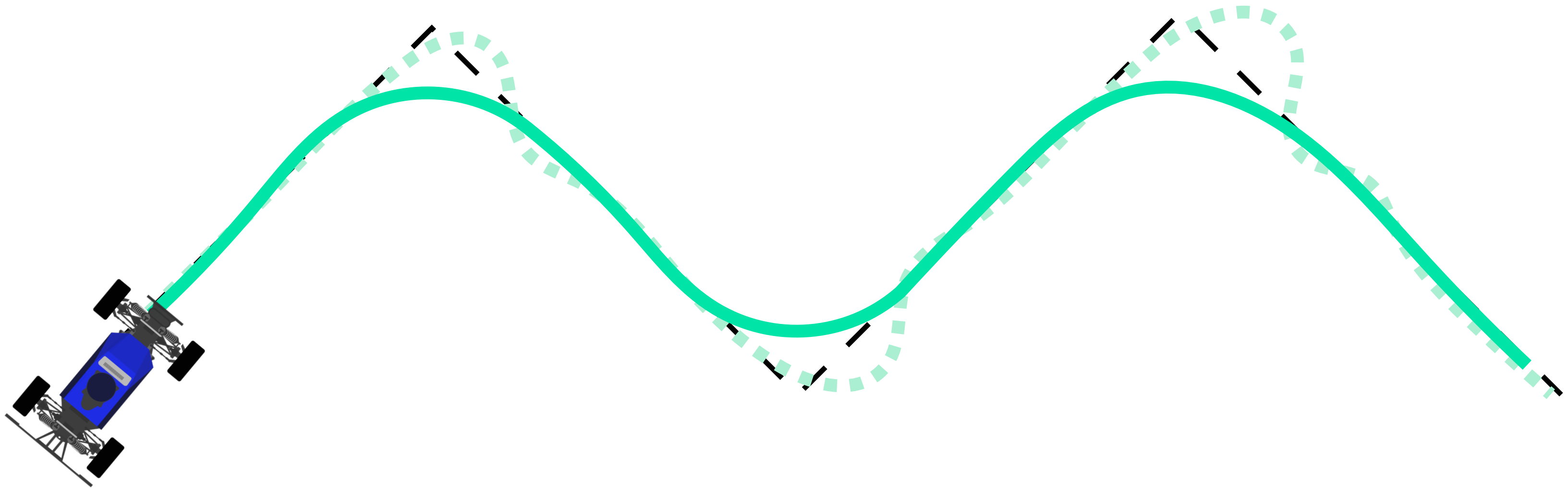
Computing the Steering Angle



$$R_{\mathrm{pp}} = \frac{a^2 + b^2}{2b}$$

$$\tan \delta = \frac{L}{R_{\mathrm{pp}}}$$

Question: How do I choose L?



Controller Design Decisions

1. Get a reference path/trajectory to track
2. Pick a reference state from the reference path/trajectory
3. Compute error to reference state
4. Compute control law to minimize error

Option 1:
Bang-bang control

Option 2:
PID control

Option 3:
Pure-pursuit control

Are we done?

Class Outline

State Estimation
- Robotic System Design
- Filtering
- Localization
- SLAM

Control
- Feedback Control
- PID Control
- MPC
- LQR

Planning
- Search
- Heuristic Search
- Motion Planning
- Lazy Search

Learning
- Imitation Learning
- Policy Gradient
- Actor-Critic
- Model-Based RL