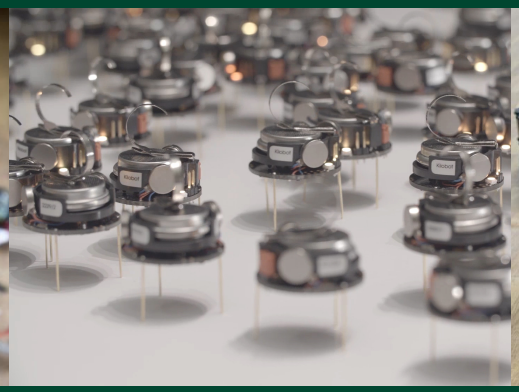
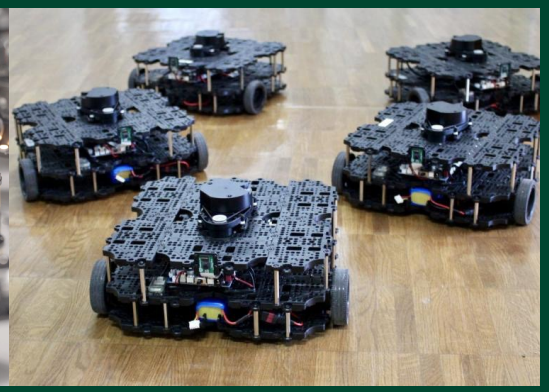
ECE693H, Spring 2025: Multi-robot System Design

“Action 2”

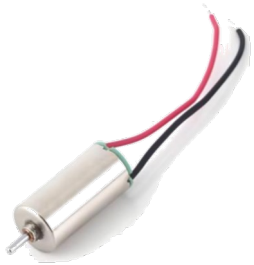



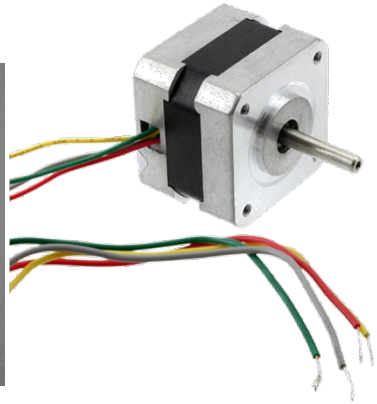
Dr. Daniel Drew

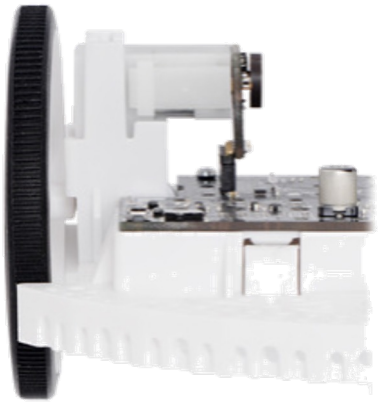
College of Engineering
University of Hawai'i at Mānoa

Action 1 Recap

1. Electromagnetic motors
2. Motor gearing and mechanics
3. Motor driver electronics
4. Encoders







Questions?

Class Overview

1. Lecture (~30 minutes)
2. Astound the Class! (~40 minutes)
3. Subsystem check-in (~5 minutes)

Action: Lecture 2 of 2

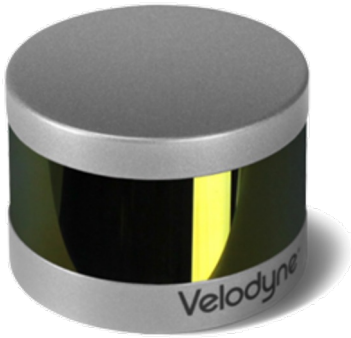
Lecture 1: motors → encoders
Lecture 2: motor control → motion planning

Lecture Overview: Intersection with Intelligence

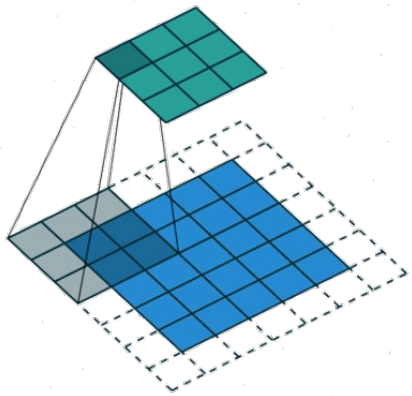
1. Motor control
2. Odometry
3. Motion planning

Closing sensorimotor loops;
Algorithmic decision making

Action

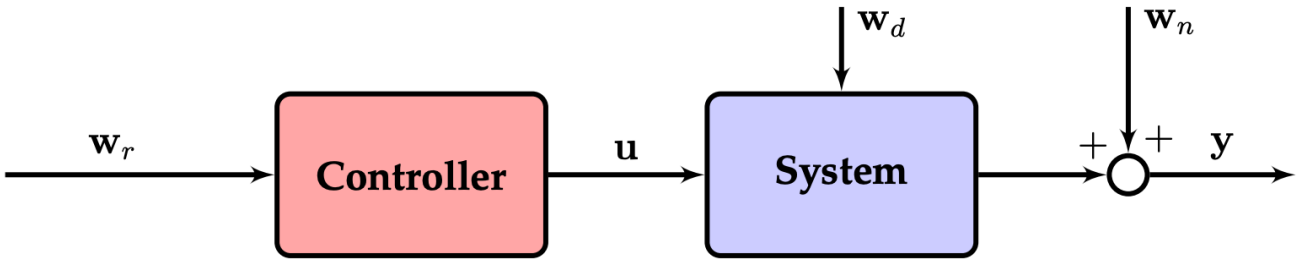


Perception



Intelligence

Motor Control: Feedback Principles



Open-loop control diagram

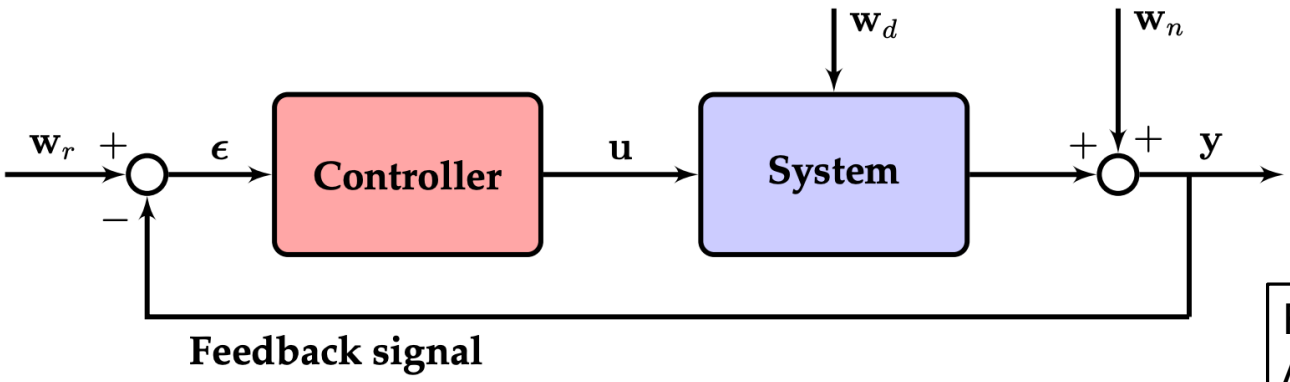
e.g.,

40RPM desired (w_r)

40 * 0.8 = PWM%

"controller"

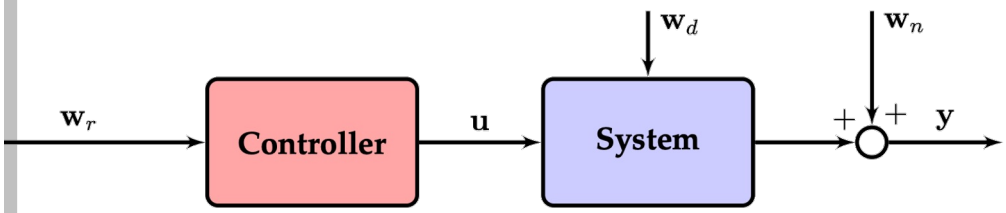
Might only work on flat ground, with full battery, for a perfectly assembled robot, with no payload, etc.

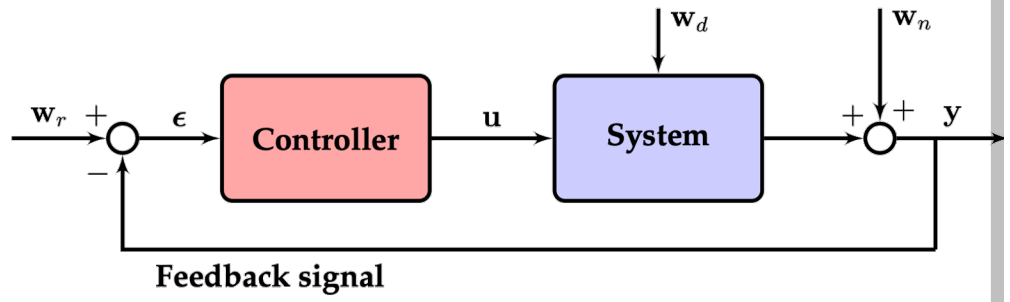


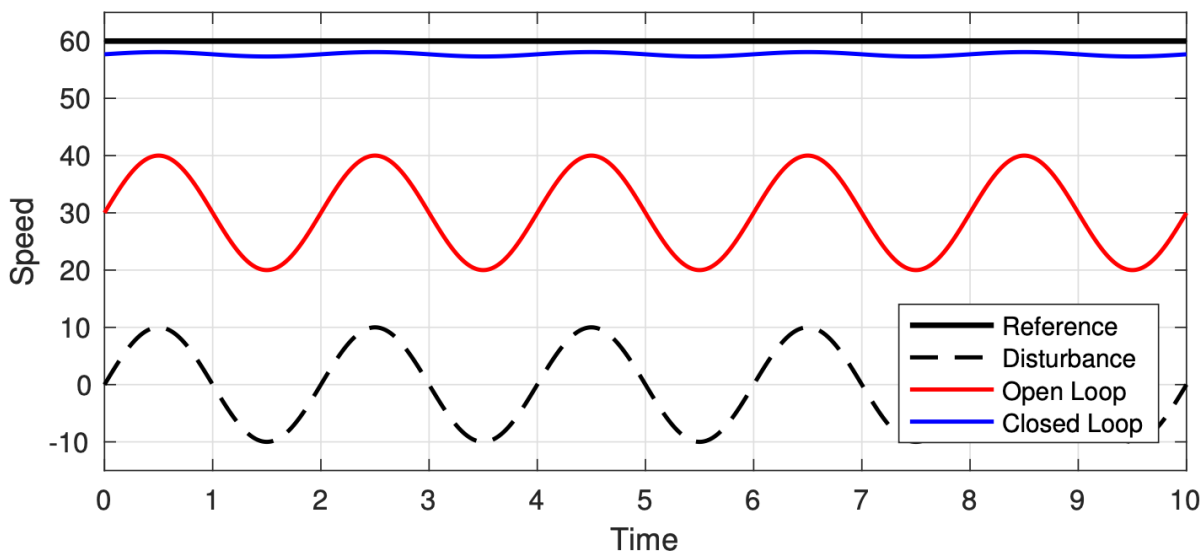
Closed-loop control diagram

Feeds measurements of the ACTUAL RPM back to the controller

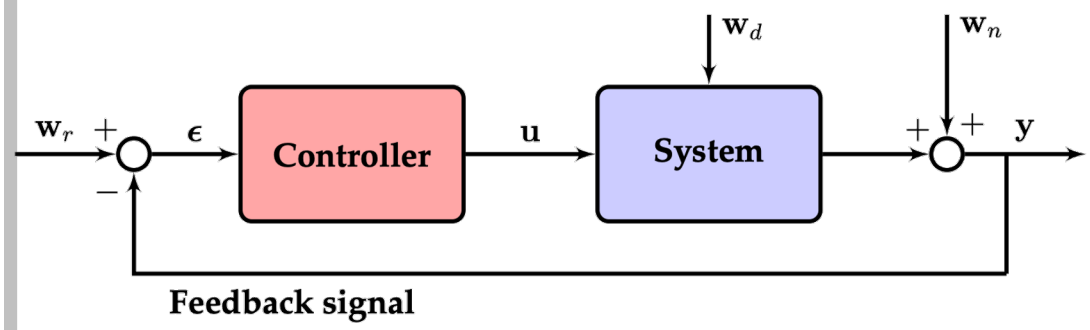
Motor Control: Feedback Principles







Motor Control: PID at a High Level



Laplace transform

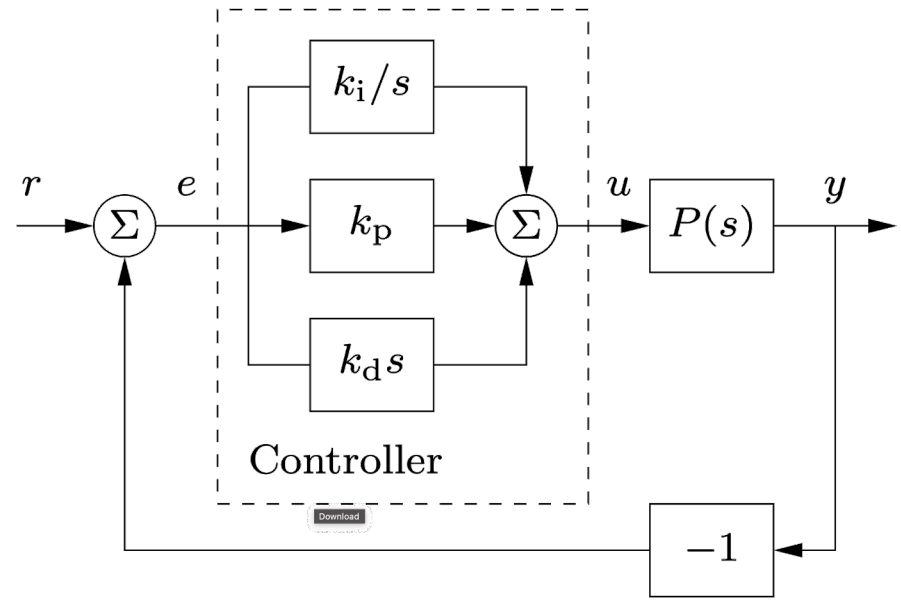
$$\frac{de}{dt} = se$$

$$\int e\,dt = \frac{1}{s}e$$

Proportional
Integral
Derivative

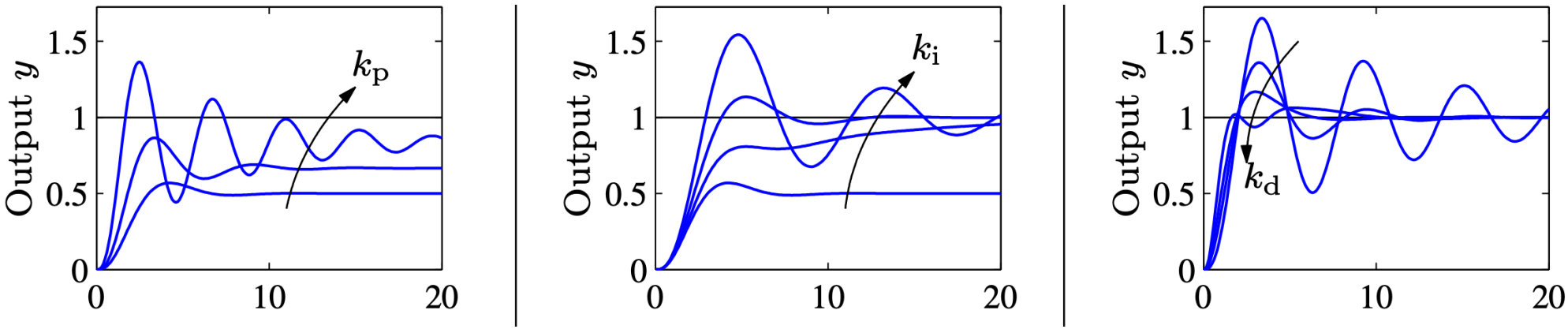
terms which work on the <u>error</u>

$$u = k_\mathrm{p} e + k_\mathrm{i} \int_0^t e(\tau)\,d\tau + k_\mathrm{d}\,\frac{de}{dt}$$



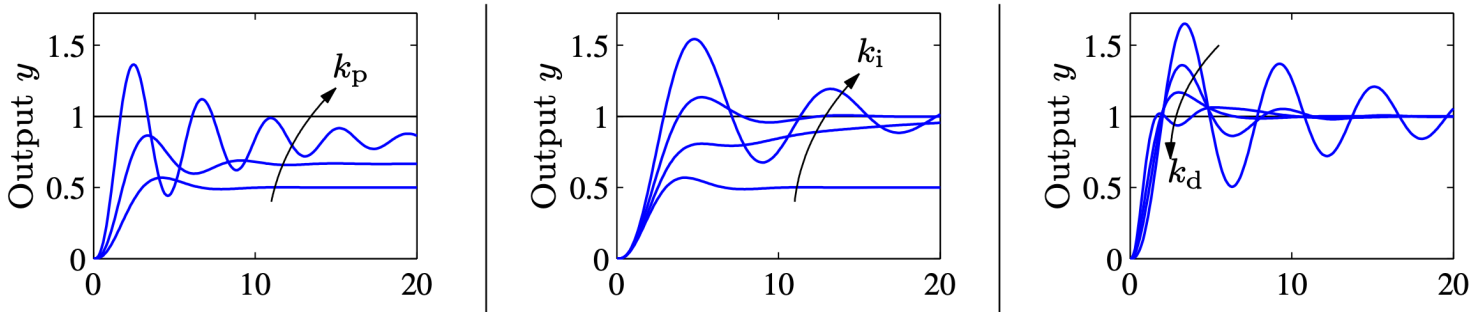
Motor Control: PID at a High Level

$$u = k_\mathrm{p} e + k_\mathrm{i} \int_0^t e(\tau)\, d\tau + k_\mathrm{d} \frac{de}{dt}$$

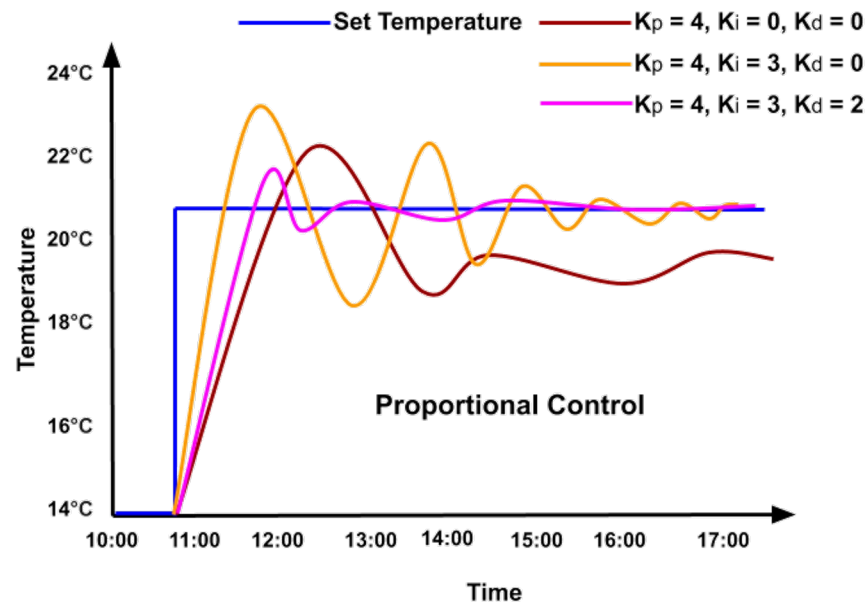


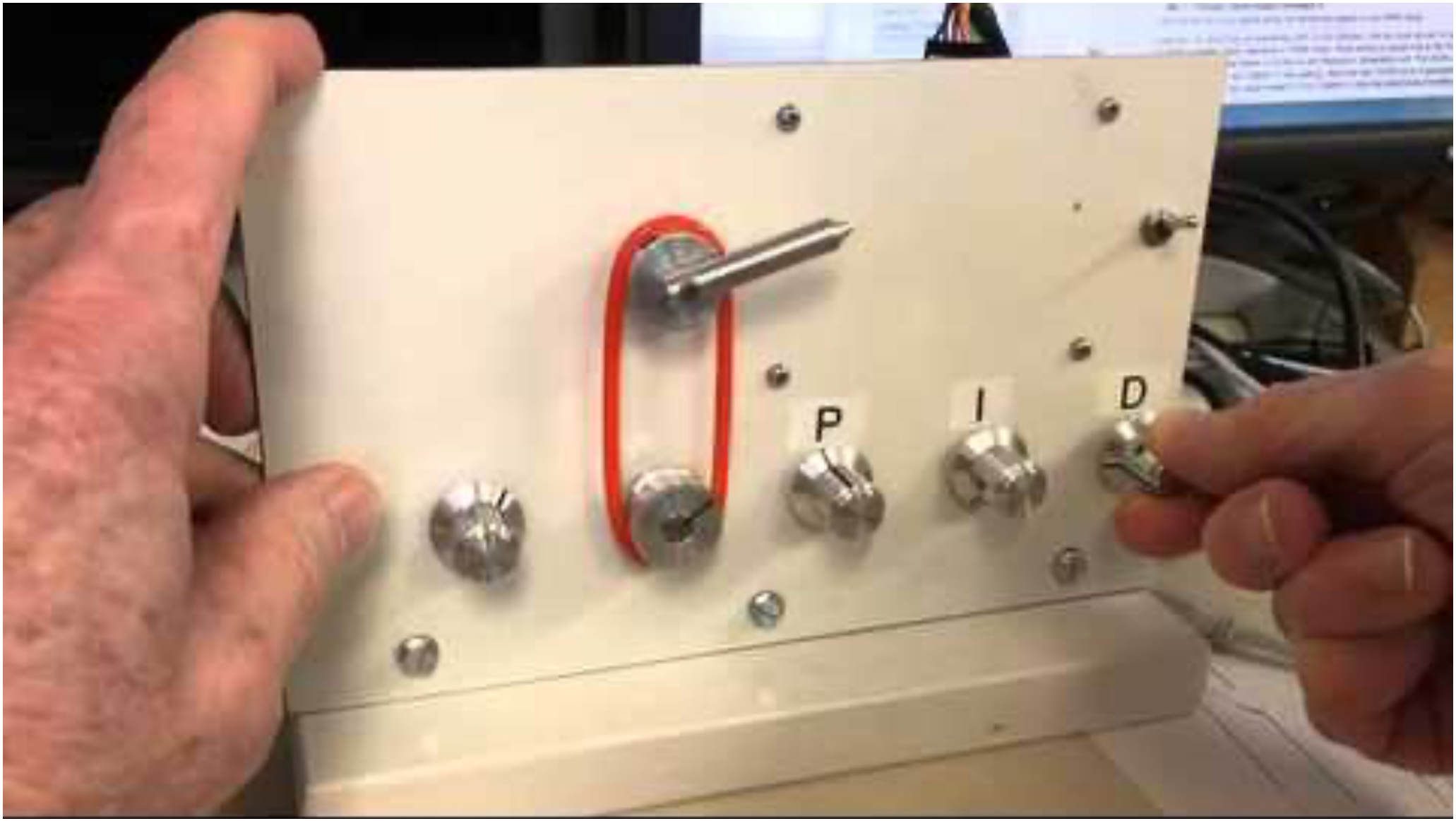
Integral = eliminate steady state error
Derivative = damping (resist fast changes)

Motor Control: PID Tuning

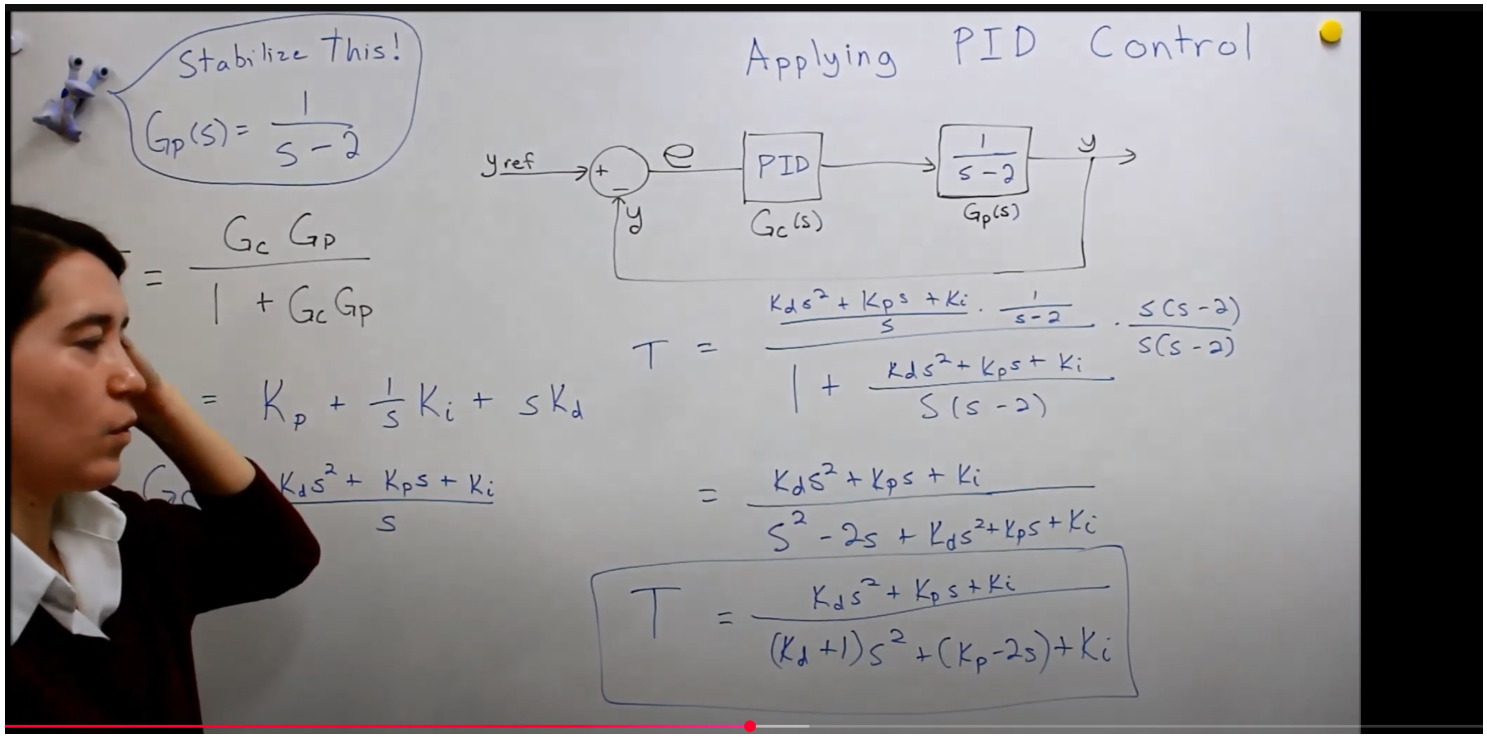


1. Find order-of-magnitude for k_p based on paper math
2. Tune k_p until slight overshoot and ringdown (stable)
3. Tune k_i to eliminate steady state error
4. Tune k_d to minimize overshoot and ringing




P
I
D

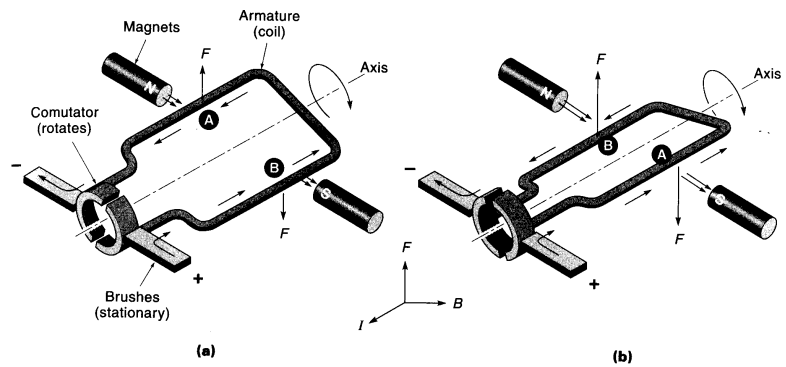
Motor Control: PID at a Low Level



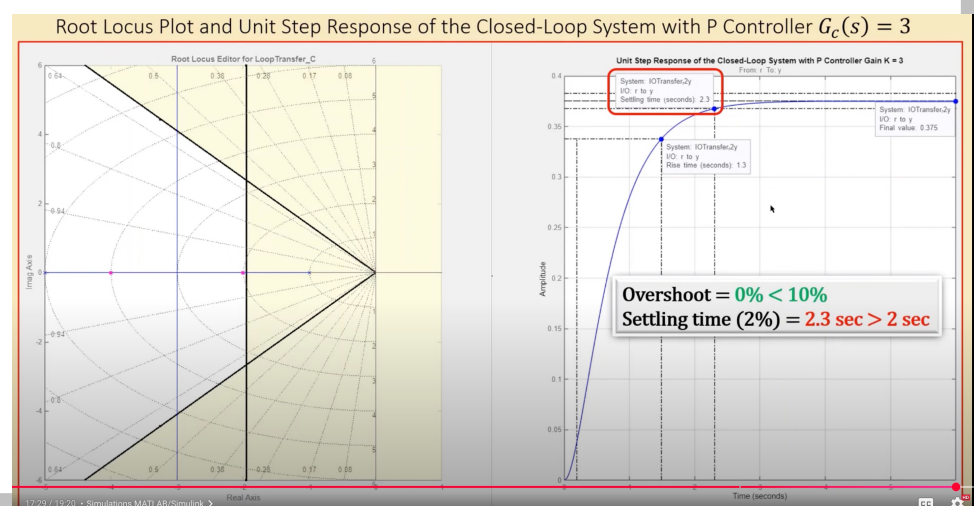
Transfer functions, root locus analysis, …

Motor Control: PID at a Low Level

$$\omega = \frac{\frac{k_t}{R_a}V - T_{load}}{\frac{k_t k_e}{R_a}}$$ **steady state velocity**



$$G\omega(s) = \frac{\omega(s)}{V(s)} = \frac{K_t}{(LS+R)(JS+b)+K_t K_b}$$



ECE 351 Linear Feedback-Control Systems (3)

College of Engineering, Electrical and Computer Engineering (ECE)

Analysis/design of feedback systems. Compensator design via root locus and Bode analysis. Routh/Nyquist stability. State space representation and introduction to MIMO formulation. Controllability/observability. Application to physical dynamic systems such as industrial robots. Pre: 315 or ME 375 or consent.

Odometry: Basic Principles

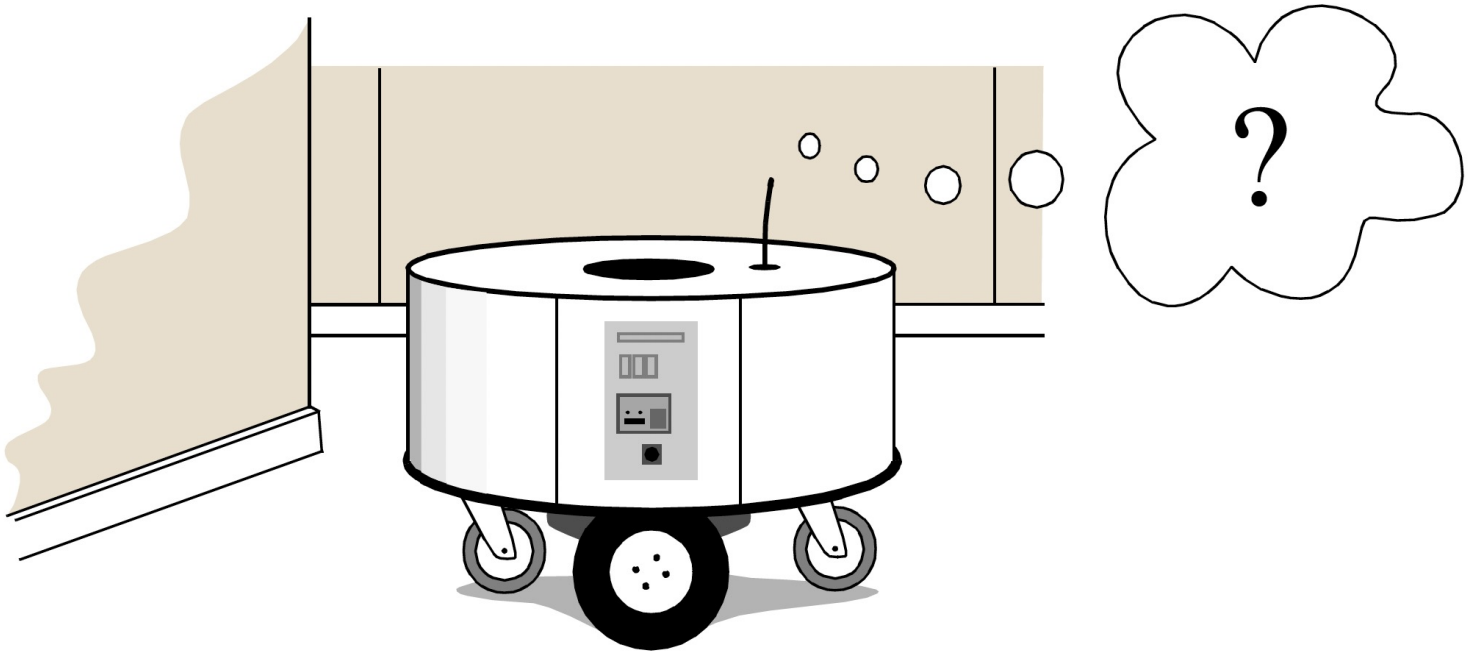
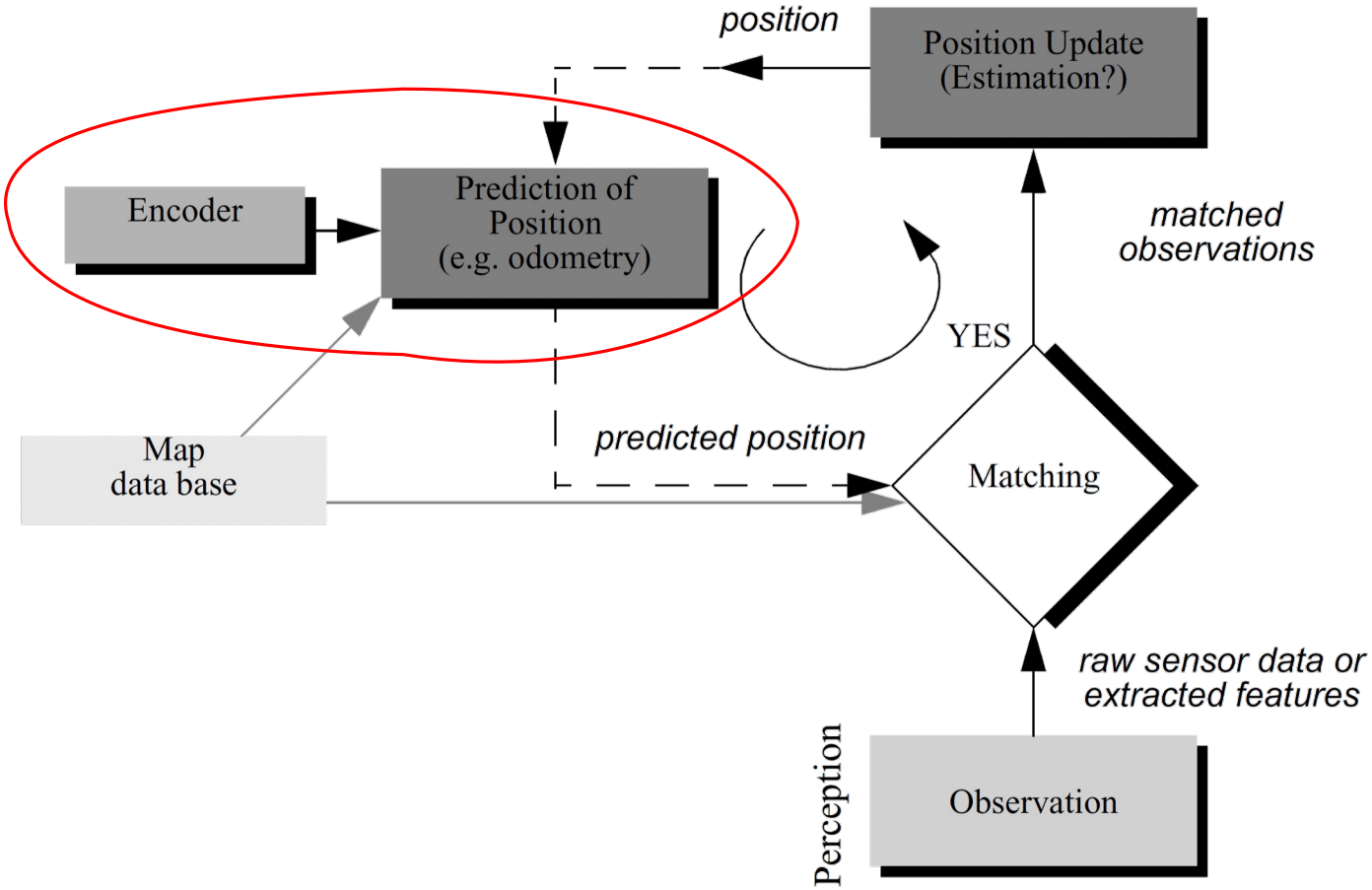
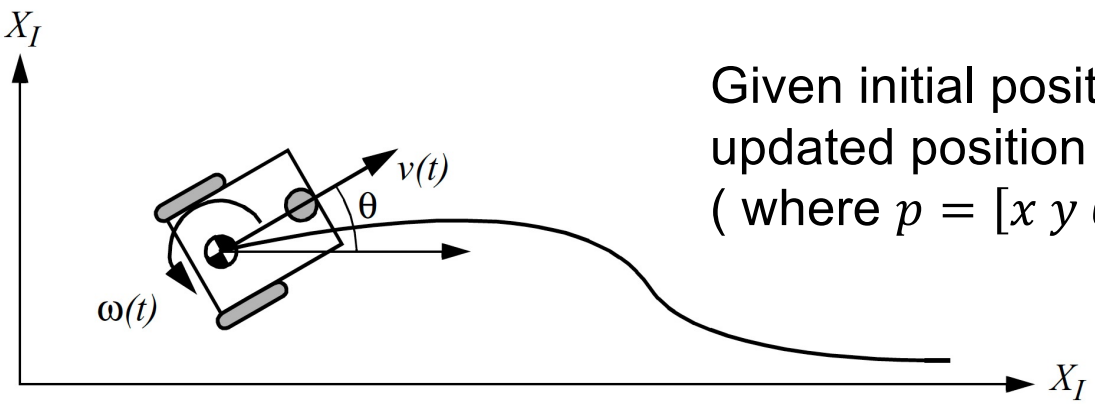


Figure 5.1
Where am I?

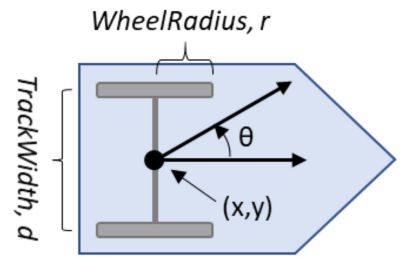
Odometry: Localization - Basic Principle



Odometry: Basic Principles



Given initial position p, how do I compute the updated position p' based on wheel velocities?
(where $p = [x\ y\ \theta]$)

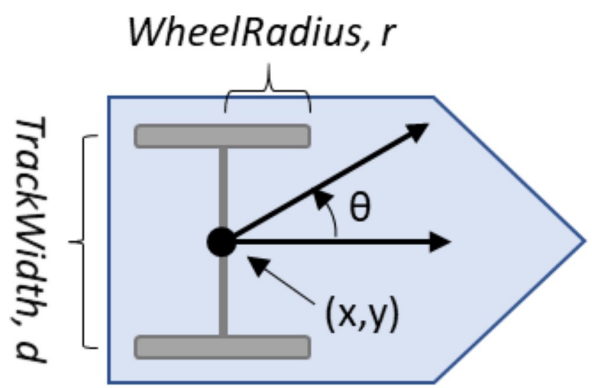


$$\Delta x = \Delta s \cos(\theta + \Delta\theta/2)$$
$$\Delta y = \Delta s \sin(\theta + \Delta\theta/2)$$
$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b}$$
$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2}$$
$$s = \frac{\Delta enc}{CPR * gr} * c_{wheel}$$

$$p' = f(x, y, \theta, \Delta s_r, \Delta s_l) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} \cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2} \sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix}$$

Odometry: Sources of Error

Major sources:
Wheel misalignment θ
Wheel size difference $c_{wheel} \rightarrow \Delta s$
Wheel slippage $\Delta enc \rightarrow \Delta s$



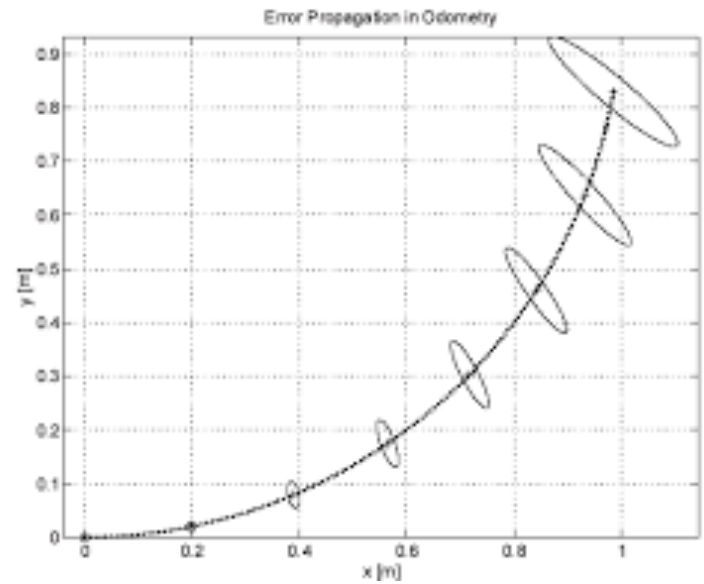
$$\Delta x = \Delta s \cos(\theta + \Delta\theta/2)$$
$$\Delta y = \Delta s \sin(\theta + \Delta\theta/2)$$
$$\Delta\theta = \frac{\Delta s_r - \Delta s_l}{b}$$
$$\Delta s = \frac{\Delta s_r + \Delta s_l}{2}$$
$$s = \frac{\Delta enc}{CPR * gr} * c_{wheel}$$

Odometry: Error (mathsplosion)

$$\Sigma_\Delta = covar(\Delta s_r, \Delta s_l) = \begin{bmatrix} k_r|\Delta s_r| & 0 \\ 0 & k_l|\Delta s_l| \end{bmatrix}$$

$$F_p = \nabla_p f = \nabla_p(f^T) = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta s \sin(\theta + \Delta\theta/2) \\ 0 & 1 & \Delta s \cos(\theta + \Delta\theta/2) \\ 0 & 0 & 1 \end{bmatrix}$$

$$F_{\Delta_{rl}} = \begin{bmatrix} \frac{1}{2}\cos\left(\theta + \frac{\Delta\theta}{2}\right) - \frac{\Delta s}{2b}\sin\left(\theta + \frac{\Delta\theta}{2}\right) & \frac{1}{2}\cos\left(\theta + \frac{\Delta\theta}{2}\right) + \frac{\Delta s}{2b}\sin\left(\theta + \frac{\Delta\theta}{2}\right) \\ \frac{1}{2}\sin\left(\theta + \frac{\Delta\theta}{2}\right) + \frac{\Delta s}{2b}\cos\left(\theta + \frac{\Delta\theta}{2}\right) & \frac{1}{2}\sin\left(\theta + \frac{\Delta\theta}{2}\right) - \frac{\Delta s}{2b}\cos\left(\theta + \frac{\Delta\theta}{2}\right) \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix}$$



Odometry: Sensor Fusion

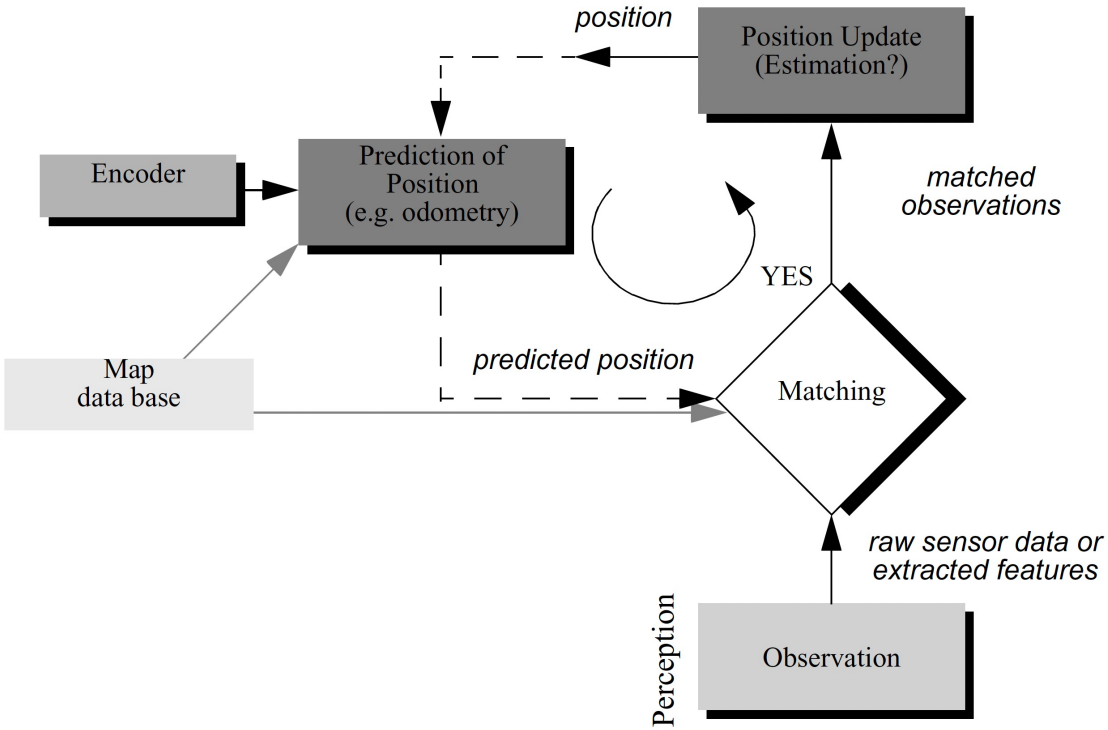
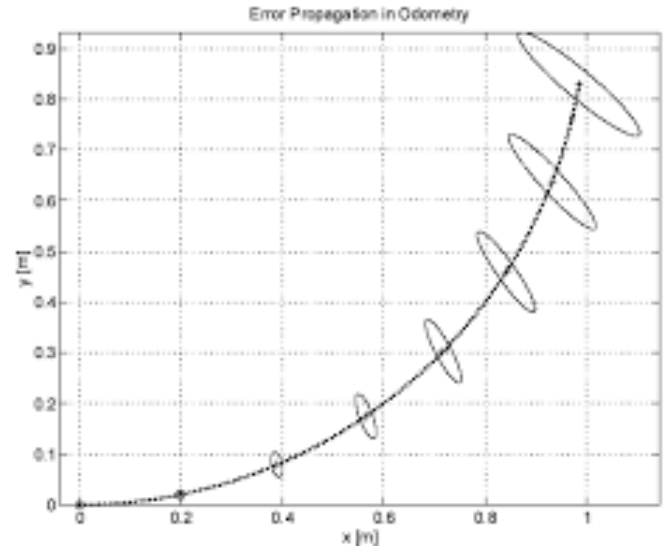


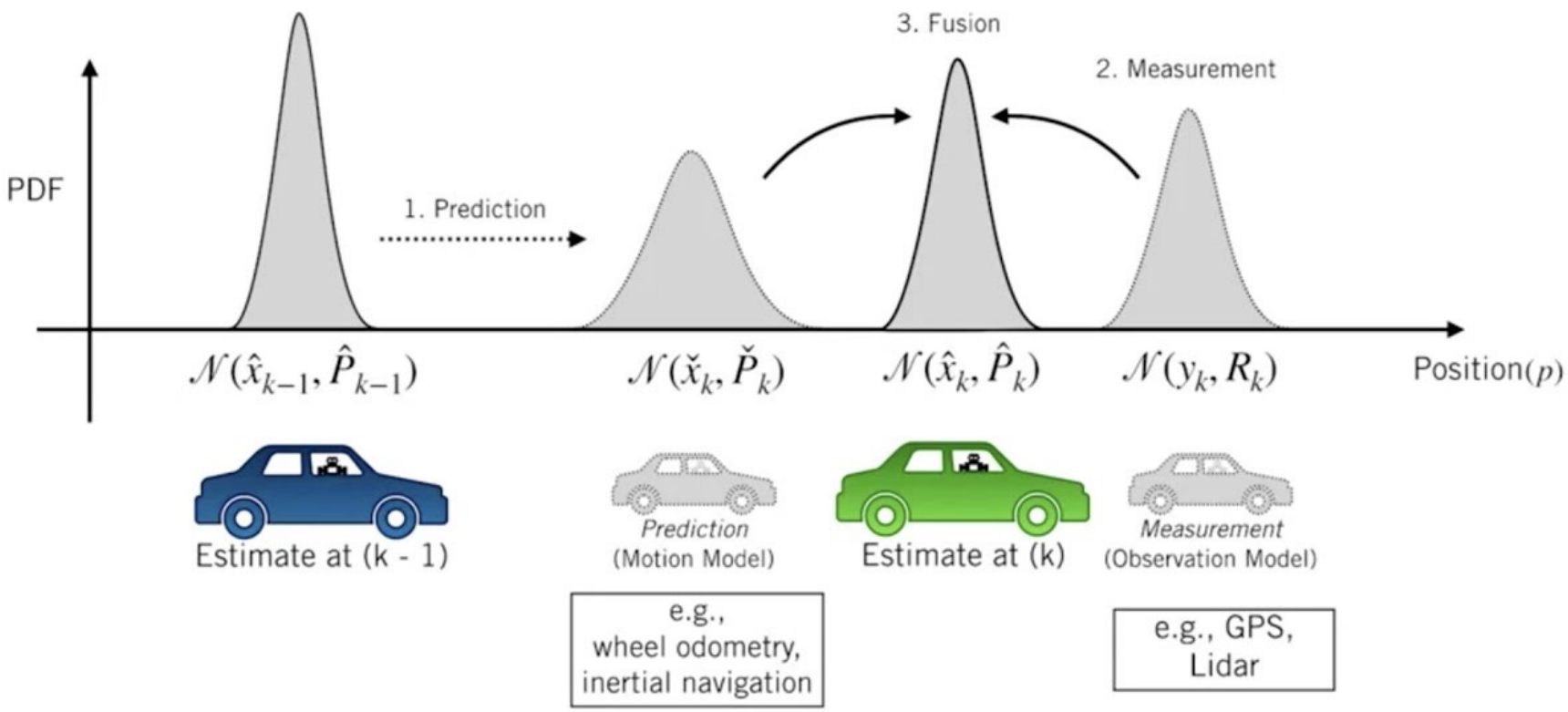
Figure 5.2
General schematic for mobile robot localization.



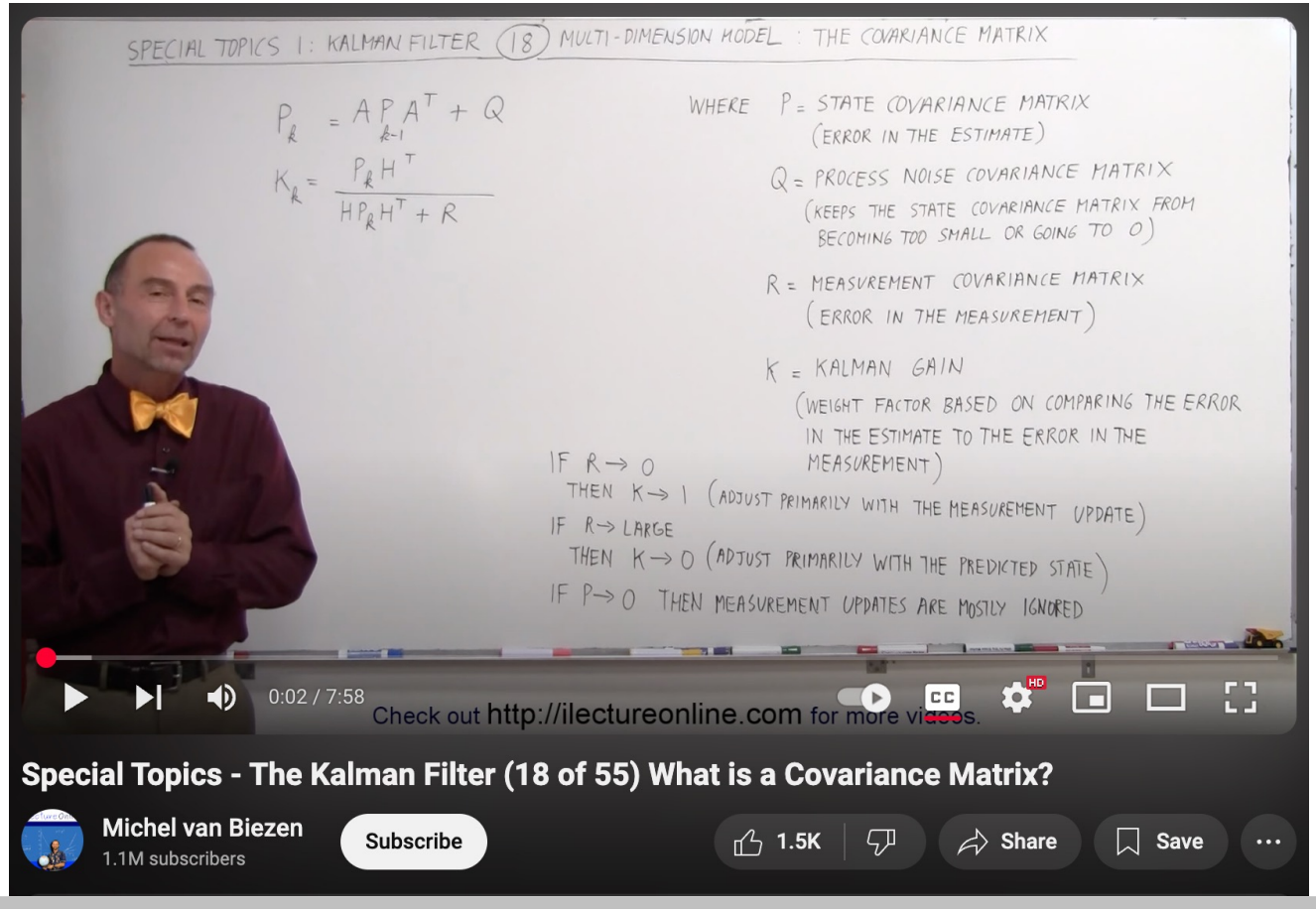
Error based on encoders grows with every estimate…
how to re-zero it?

→ Use a more trusted reference (gps, motion capture, …)

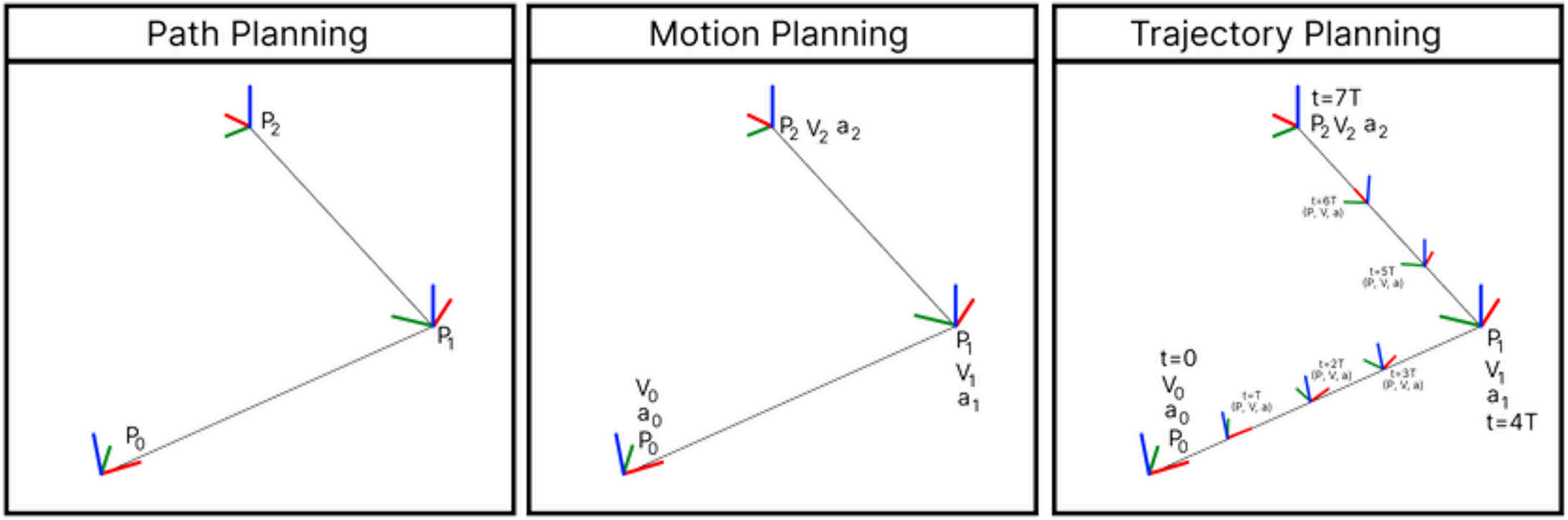
Odometry: Sensor Fusion



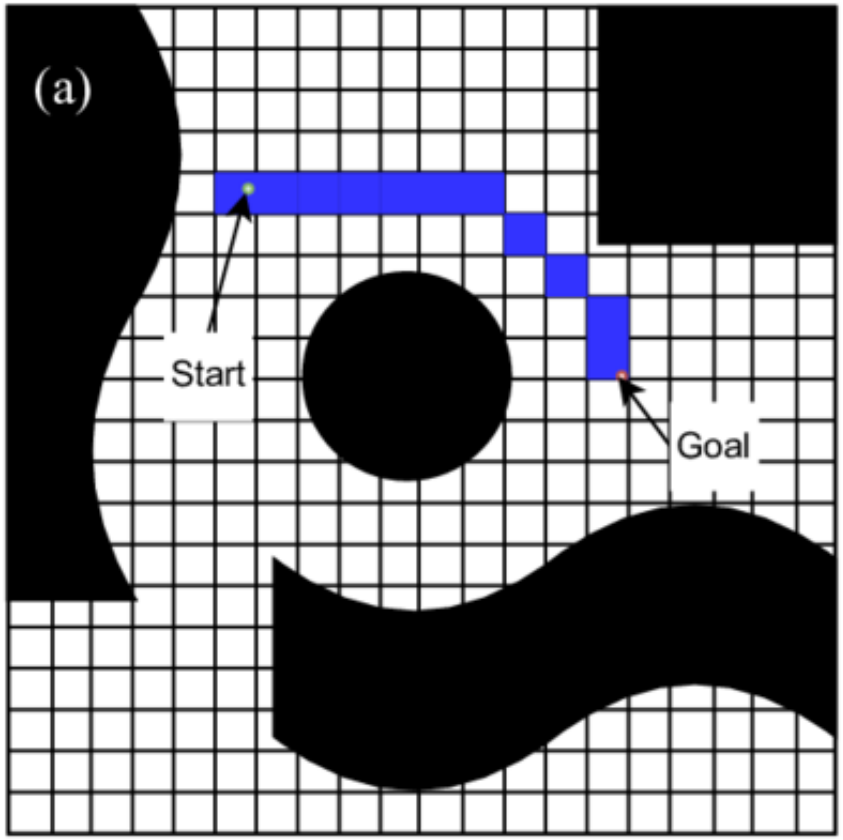
Odometry: Error



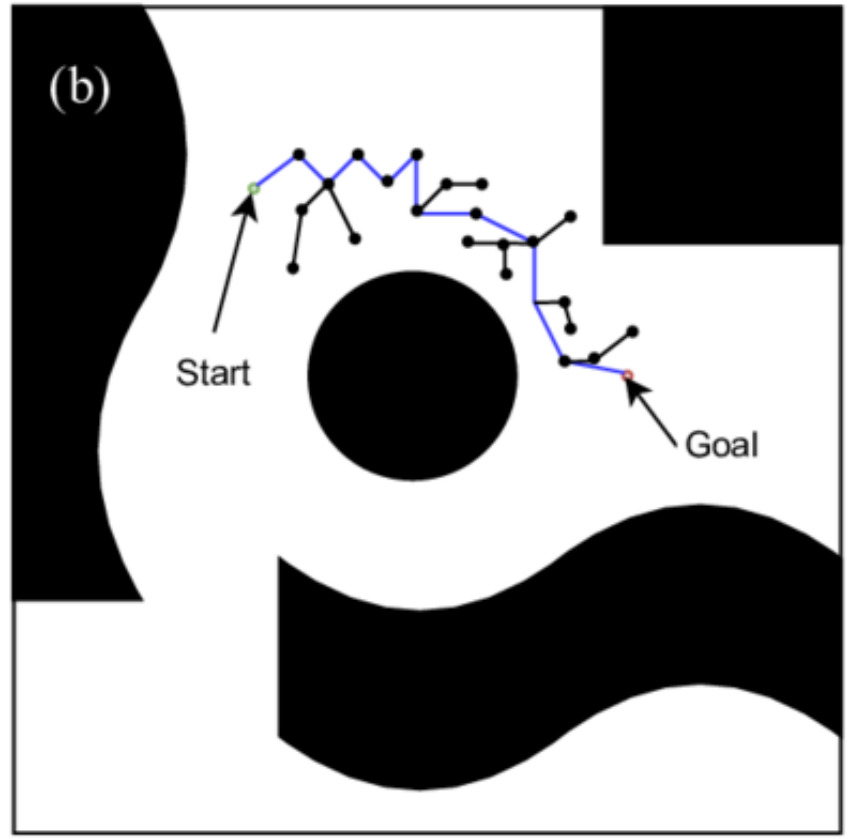
Motion Planning: High Level



Path Planning: Search vs Sampling

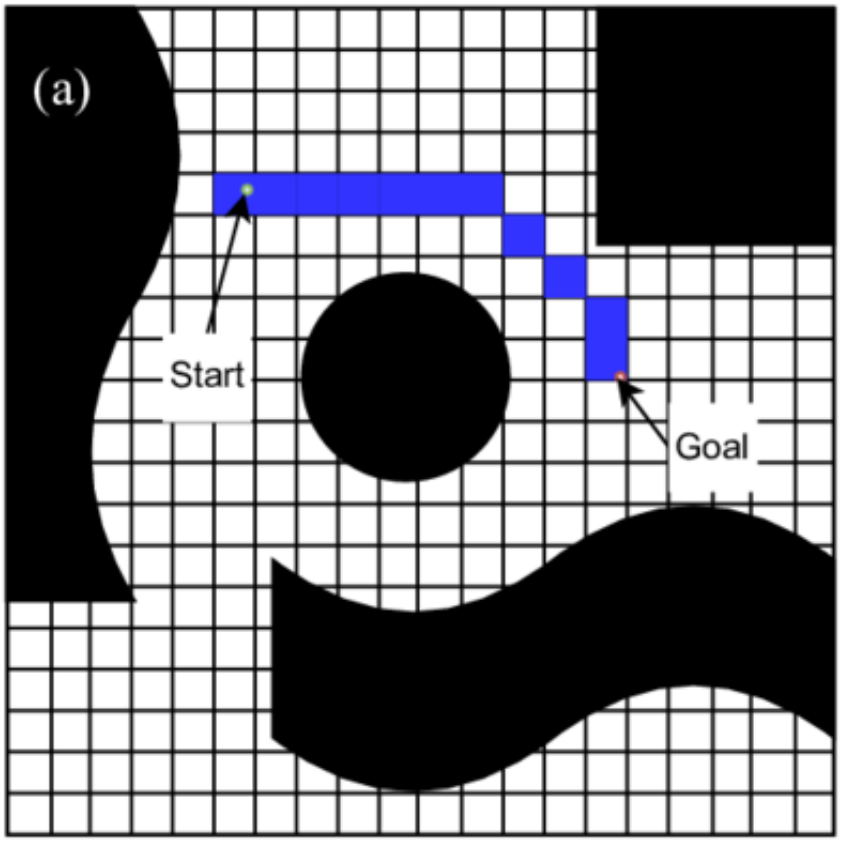


Grid-based Search



Sampling

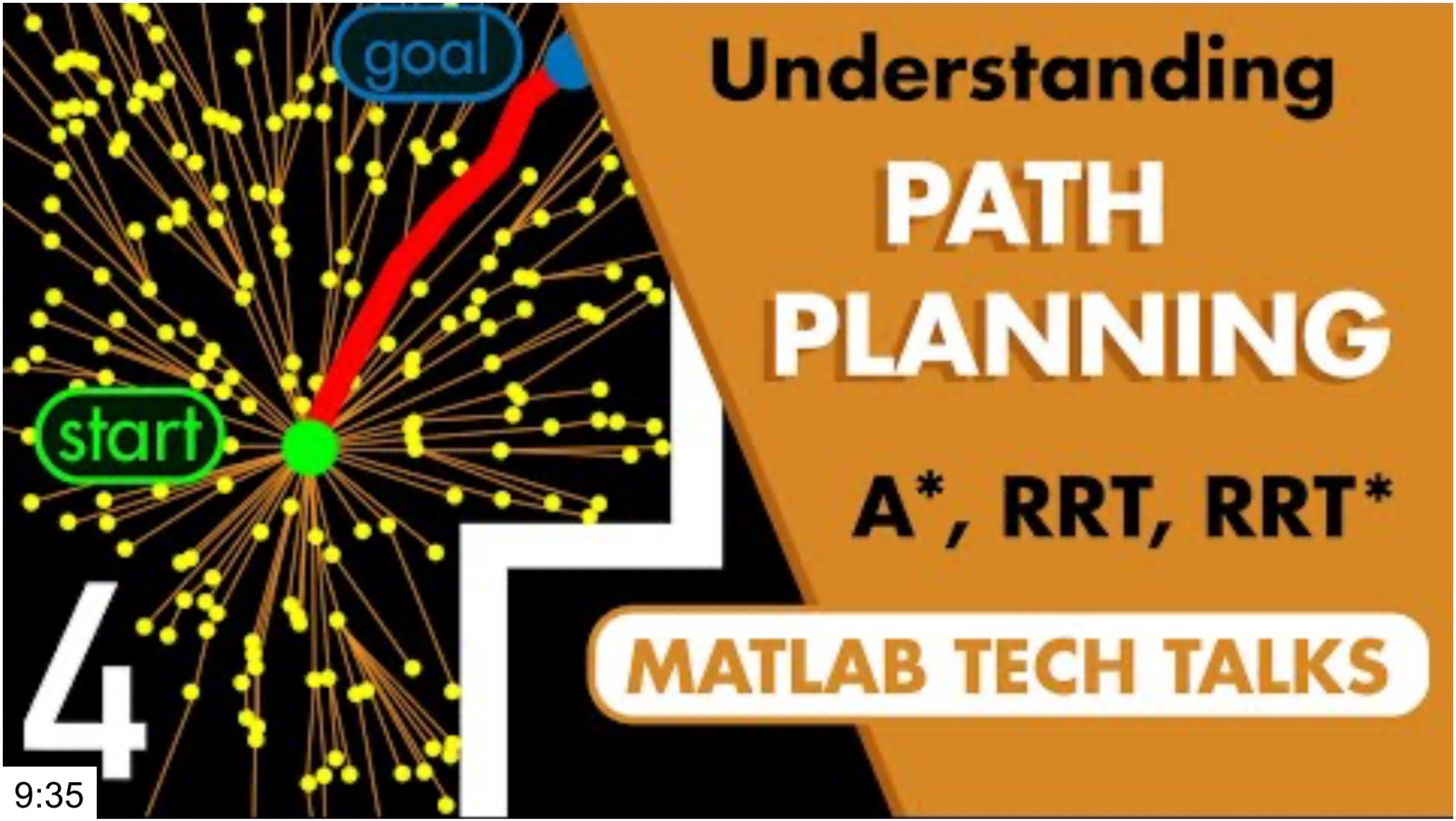
Path Planning: Search



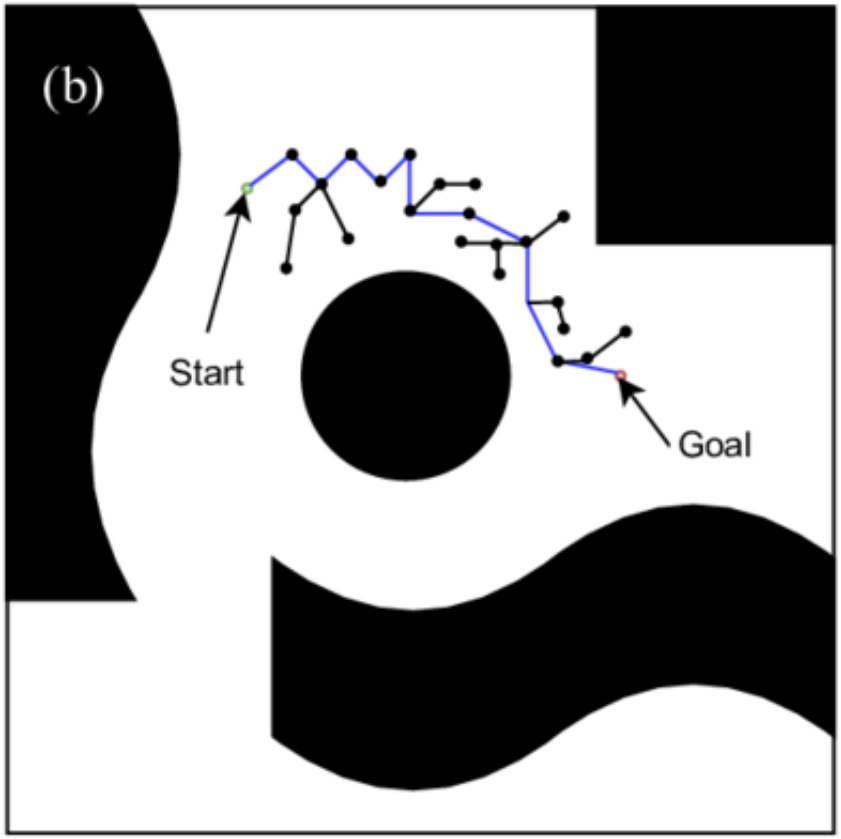
Grid-based Search

Given a weighted graph, find the shortest (lowest cost) path from A to B

e.g., A*, Dijkstra's


goal
start
4
Understanding
PATH
PLANNING
A*, RRT, RRT*
MATLAB TECH TALKS
9:35

Path Planning: Sampling

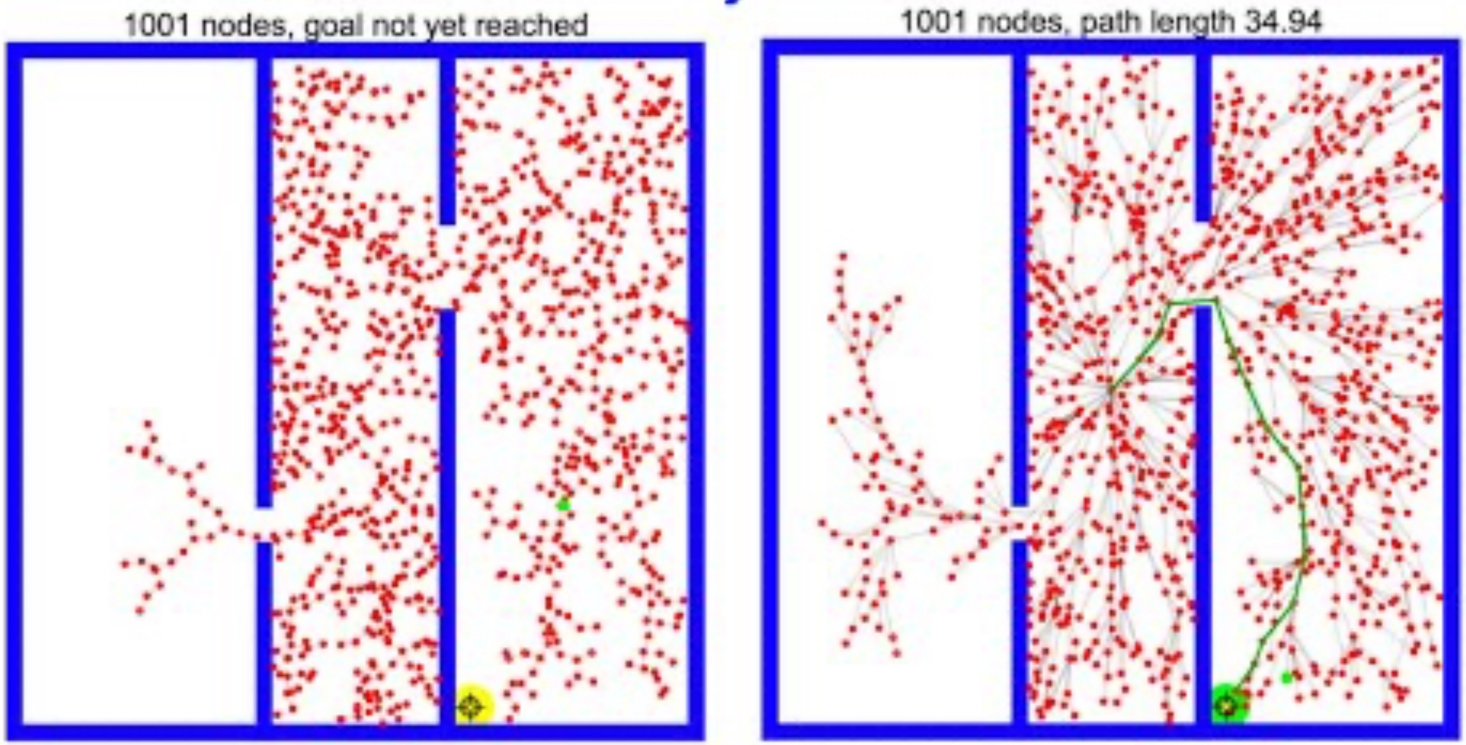


Sampling

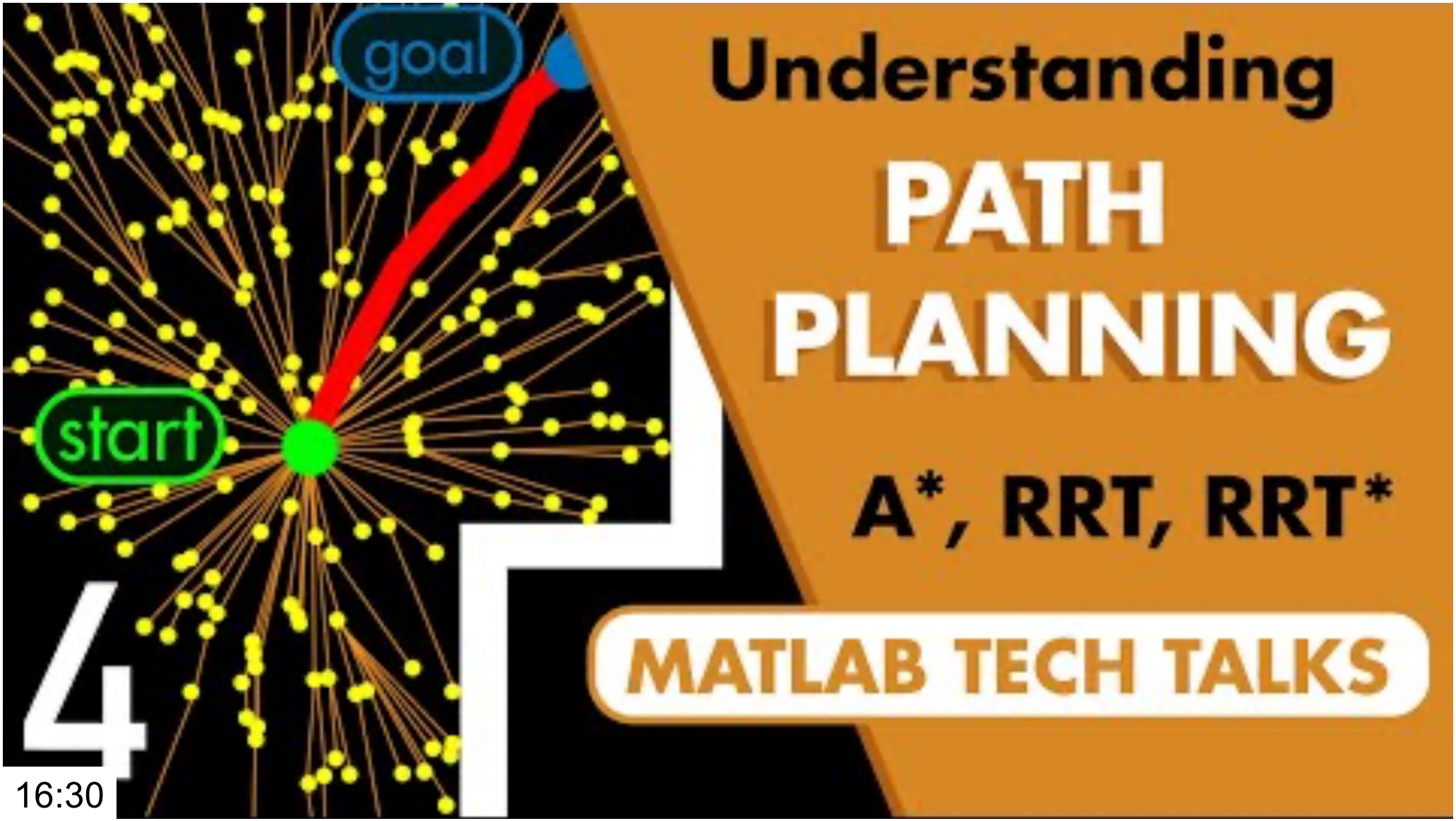
Given start position and goal position, randomly test (sample) points in the world to find shortest (lowest cost) path

e.g., RRT, RRT*, PRM

Random Trees, RRTs & RRT*

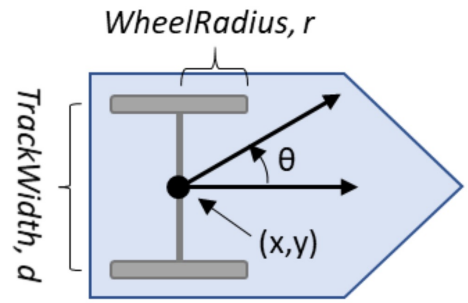


4:00


goal
start
4
Understanding
PATH
PLANNING
A*, RRT, RRT*
MATLAB TECH TALKS

16:30

What About Motion Planning



$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r}{2}\cos(\theta) & \frac{r}{2}\cos(\theta) \\ \frac{r}{2}\sin(\theta) & \frac{r}{2}\sin(\theta) \\ -r/d & r/d \end{bmatrix} \begin{bmatrix} \dot{\phi}_L \\ \dot{\phi}_R \end{bmatrix}$$

Output of path planning:
A sequence of poses.

Kinematics + odometry gives us:
Change in position given wheel velocities.

$$p' = f(x, y, \theta, \Delta s_r, \Delta s_l) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2}\cos\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r + \Delta s_l}{2}\sin\left(\theta + \frac{\Delta s_r - \Delta s_l}{2b}\right) \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix}$$

SWITCH TO ASTOUND THE CLASS GSLIDES

693H Robotic Subsystems

Motors, drive electronics, encoders, PID controllers, top-down tracking

Sensor suite (IMU, prox.), calibration code, general IO (buttons and LEDs)

Mobility

Perception

Compute and Power

Control and Planning

Battery and charging, wireless comms, GPIO and power distribution routing

microROS, ROS2, Gazebo simulation, enclosure design

Mechanics / Prototyping

Coding

PCB Design

Schedule Check-in

Week	Module	Date	Lecture Topic	In-Class Activity
Week 1	**Intro**	1/13	Course Intro	Icebreakers
		1/15	Autonomous Mobile Robots	Subsystem Group Formation
Week 2		1/20	**MLK Day**	
	Design Phase 1	1/22	Action 1	
Week 3		1/27	Action 2	Astound the Class! 1
		1/29	Perception 1	
Week 4		2/3	Perception 2	Subsystem Check-ins 1
		2/5	Intelligence 1	
Week 5		2/10	Design Review Presentations	
		2/12		
		2/17	**President's Day**	

2/3: Parts order, and ~40 minutes in-class work time for check-ins!

<u>Ask questions about Design Review 1 NOW</u>