Georgia Institute of Technology

Dr. Magnus Egerstedt
Professor
School of Electrical and
Computer Engineering

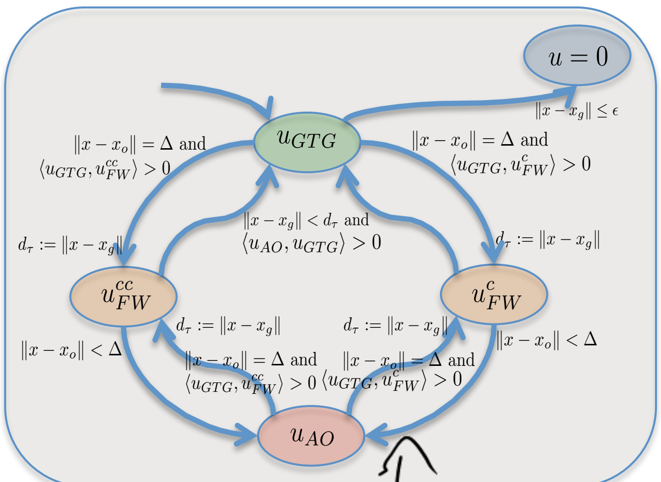
Control of Mobile Robots

Module 7
Putting It All Together

How make mobile robots move in effective, safe, predictable, and collaborative ways using modern control theory?

Lecture 7.1 – Approximations and Abstractions

- We need to understand when and how our models are relevant!



Assume $\dot{x} = u$

?

Lecture 7.1 – Approximations and Abstractions

- We need to understand when and how our models are relevant!

"Slow and steady wins the race"

Don't rush it when doing the quizzes…

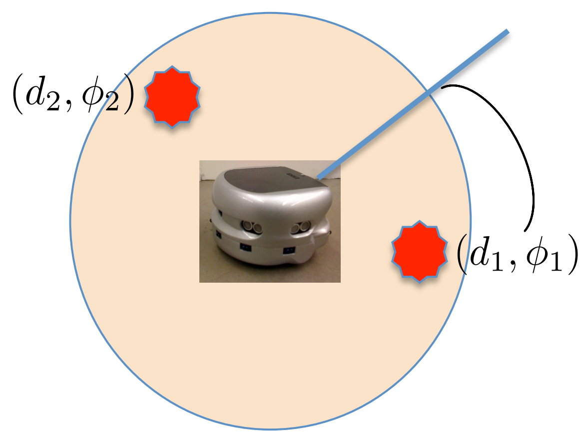
Main Assumptions Made So Far

- Dynamics:

$$\dot{x} = u, \;\; x \in \Re^2$$

Not even close to being reasonable!

- Sensors:



sensor only parts of the circle

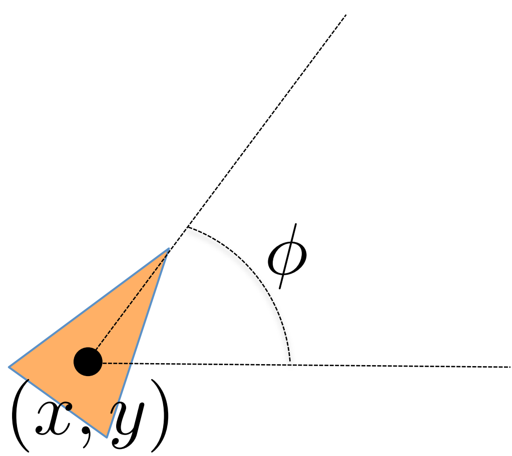
More or less ok…

What's The Problem?

They do not act the same

$$\dot{x} = u, \;\; x \in \Re^2$$

- Recall the unicycle model (e.g., for describing differential drive mobile robots)



$$\begin{cases} \dot{x} = v \cos \phi \\ \dot{y} = v \sin \phi \\ \dot{\phi} = \omega \end{cases}$$

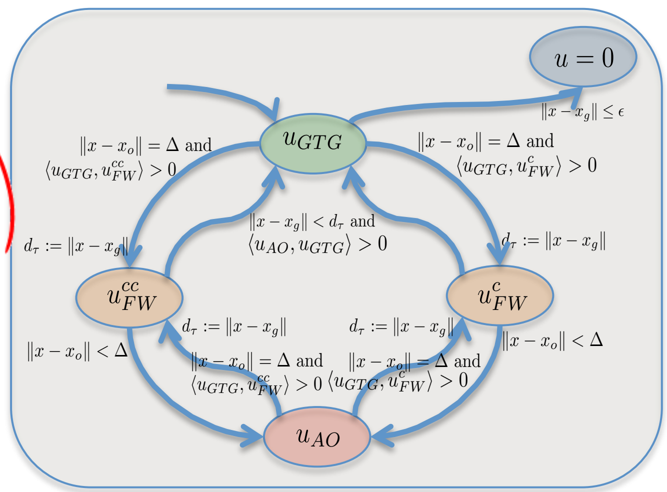
Next Few Lectures…

- How do we make a unicycle robot "act" like $\dot{x} = u$?

Lecture 7.2 – A Layered Architecture

- We have a problem: Even with a simple robot model, the navigation architecture becomes rather involved

$\dot{x} = u$
(Basic model)



- Would like to be able to reuse this while allowing for more realistic robot models

Solution: Layered architecture

All Good Things Come in Threes

- Standard navigation systems are typically decoupled along three different levels of abstraction:
  - Strategic Level: Where to go (high-level, long-term)?
  - Operational Level: Where to go (low-level, short-term)?
  - Tactical Level: How to go there?

All Good Things Come in Threes

- Or slightly less militaristic:
  - High-Level Planning: Where should the (intermediary) goal points be? Not in this course!
  - Low-Level Planning: Which "direction" to move in-between goal points? Use the navigation architecture!
  - Execution: How make the robot move in those directions? Control design with reference signal!

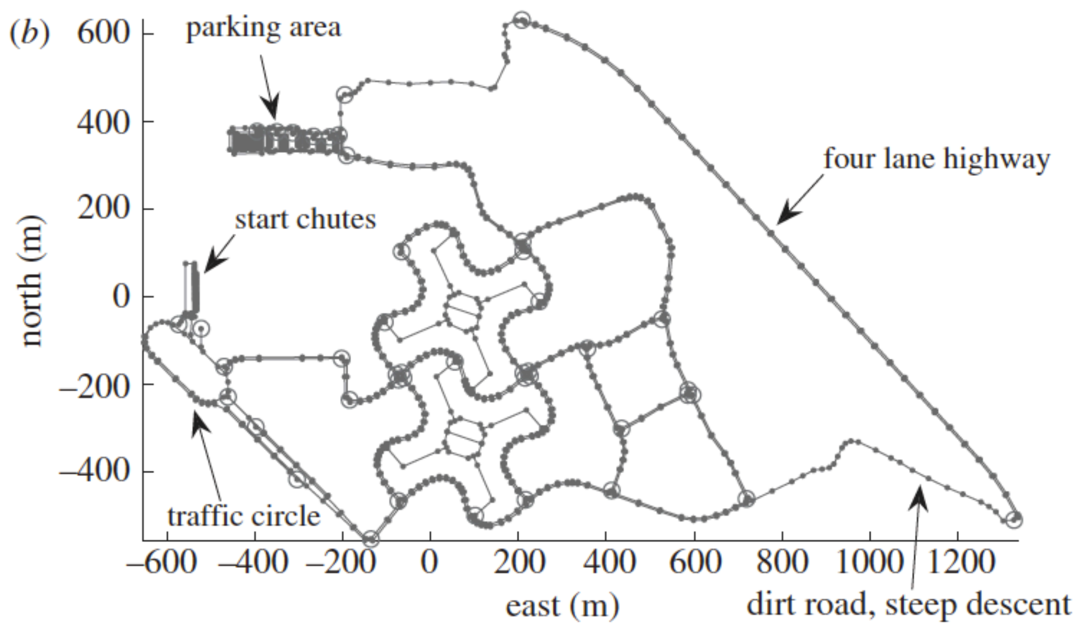
AI produces goal point

$\dot{x} = u$

Robot specific

High-Level Planning

- There are many AI methods (e.g., Dijkstra, Dynamic Programming, A*, D*, RRT…) for doing this!



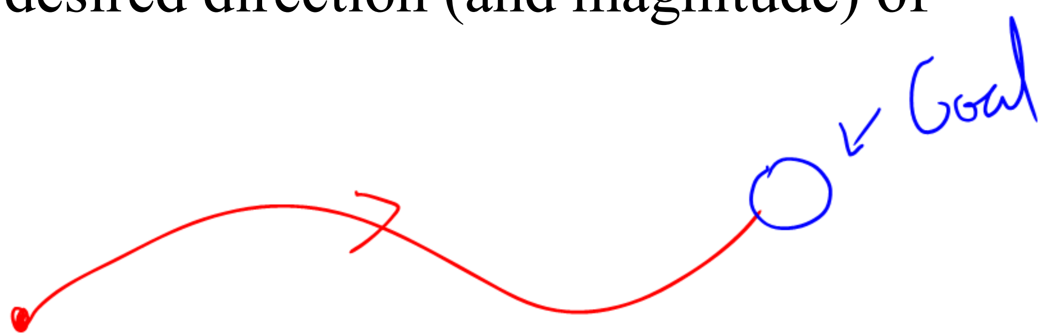
Low-Level Planning

- We already know how to do this! Assume that

$$\dot{x} = u, \;\; x \in \Re^2$$

and get to work!

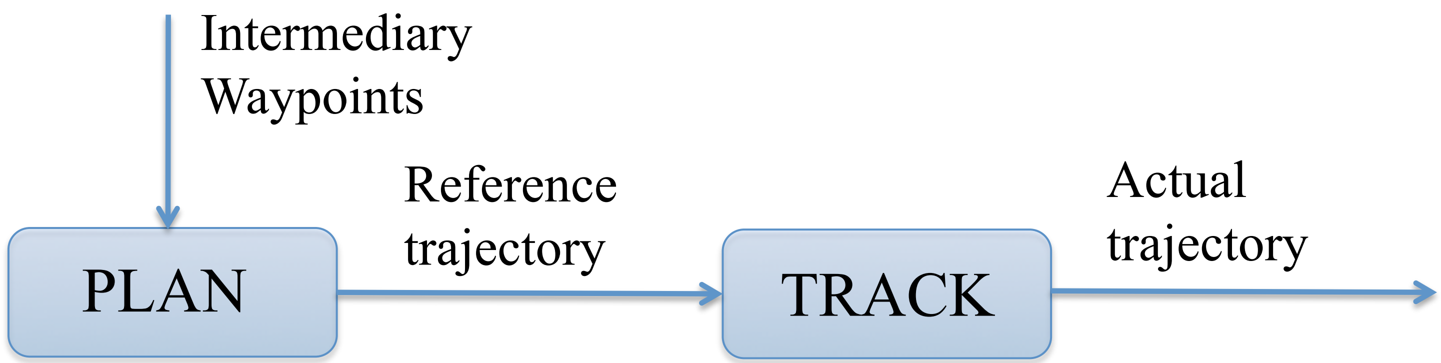
- The "output" is a desired direction (and magnitude) of travel

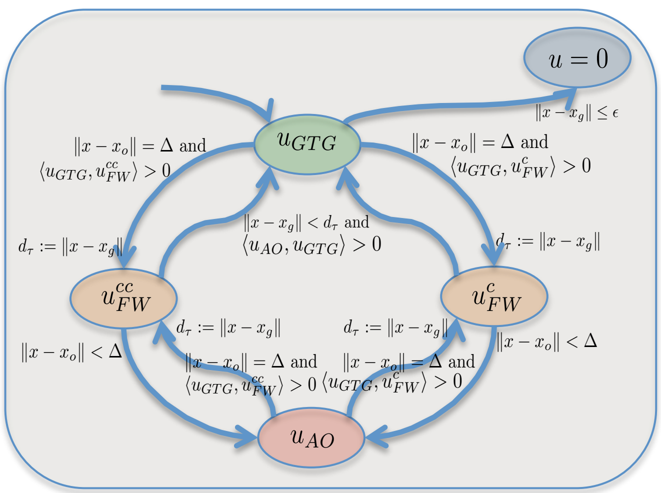


Execution-Level

- This is where we make the unicycle (or any other mobile robot) act like a simpler system over which we are performing the low-level planning

The Architecture

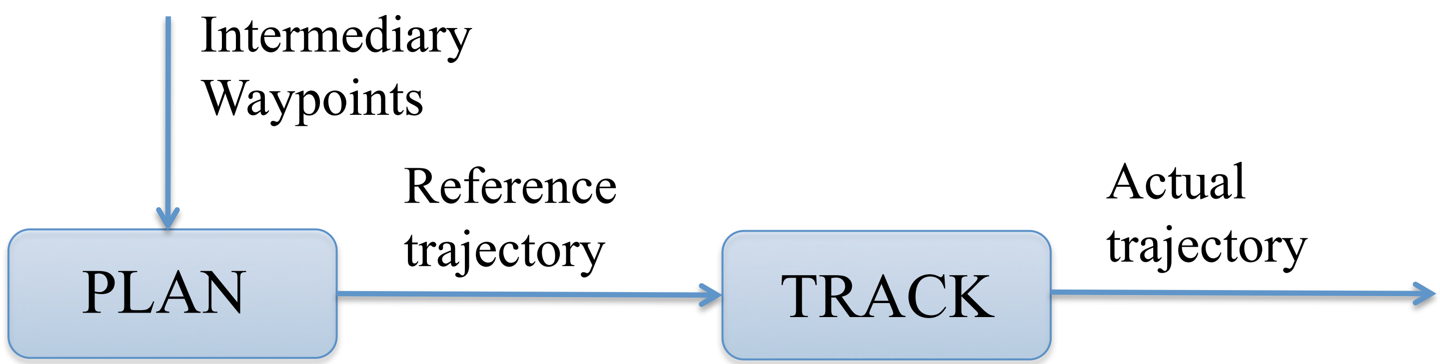




$$\dot{x} = f(x, u), \;\; u = g(x, r)$$

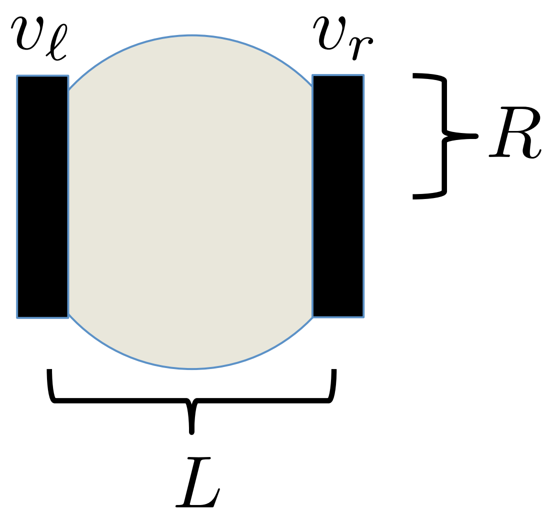
- Next time: Let's do this for the differential-drive mobile robot

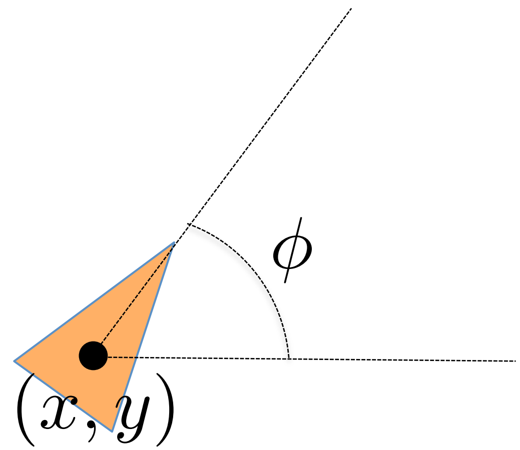
Lecture 7.3 – Differential-Drive Trackers



- How should we design the tracker when the robot is a differential-drive mobile robot?

Recap: The Model





$$v_r = \frac{2v + \omega L}{2R}$$

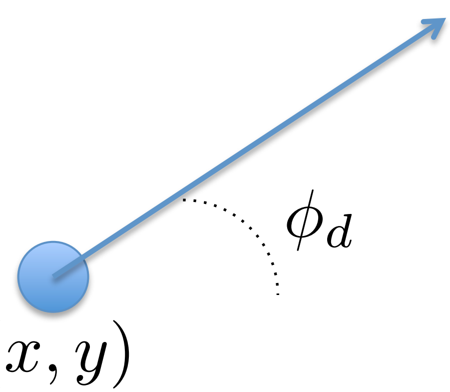
$$v_\ell = \frac{2v - \omega L}{2R}$$

$$\begin{cases} \dot{x} = \frac{R}{2}(v_r + v_\ell)\cos\phi \\ \dot{y} = \frac{R}{2}(v_r + v_\ell)\sin\phi \\ \dot{\phi} = \frac{R}{L}(v_r - v_\ell) \end{cases} \qquad \begin{cases} \dot{x} = v\cos\phi \\ \dot{y} = v\sin\phi \\ \dot{\phi} = \omega \end{cases}$$

Recap: Dealing With Angles

- How drive the robot in a specific direction?

$$\begin{cases} \dot{x} = v\cos\phi \\ \dot{y} = v\sin\phi \\ \dot{\phi} = \omega \end{cases}$$



$$e = \arctan\left(\frac{\sin(\theta_d - \theta)}{\cos(\theta_d - \theta)}\right)$$

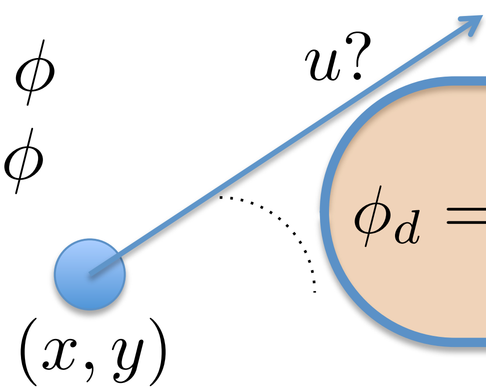
$$e = \phi_d - \phi, \quad \omega = \text{PID}(e)$$

$$PID(e) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \dot{e}(t)$$

Adding In The Speed Component

- Let the output from the planner be $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ $(\dot{x} = u)$

$$\begin{cases} \dot{x} = v\cos\phi \\ \dot{y} = v\sin\phi \\ \dot{\phi} = \omega \end{cases}$$



$$\boxed{\phi_d = \operatorname{atan}\left(\frac{u_2}{u_1}\right)}$$

$\omega = PID(e)$
$e = \phi_d - \phi$

$$\sqrt{\dot{x}^2 + \dot{y}^2} = \sqrt{v^2\cos^2\phi + v^2\sin^2\phi} = v$$

$$v = \|u\| \;\Rightarrow\; \boxed{v = \sqrt{u_1^2 + u_2^2}}$$

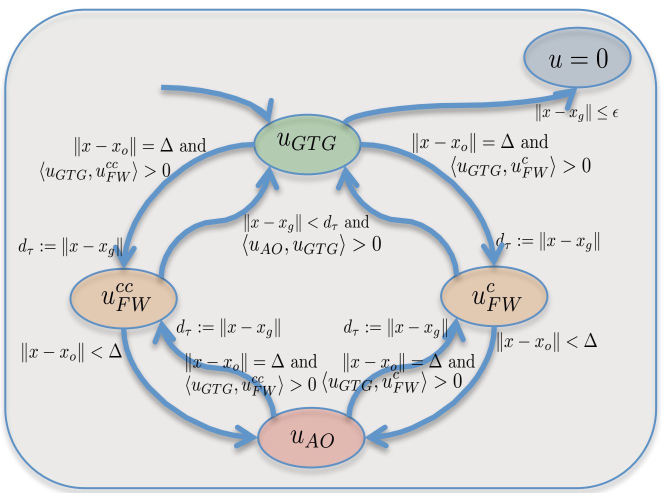
$\sqrt{v^2 \underbrace{(\cos^2 + \sin^2)}_{1}} = v$

The Complete Differential-Drive Architecture

Intermediary Waypoints

$$\downarrow$$

$$\text{PLAN} \xrightarrow{(u_1, u_2)} \text{TRACK} \xrightarrow{(v_r, v_\ell)}$$



$$\phi_d = \operatorname{atan}\left(\frac{u_2}{u_1}\right)$$

$$\omega = \text{PID}(\phi_d - \phi)$$

$$v = \sqrt{u_1^2 + u_2^2}$$

$$v_r = \frac{2v + \omega L}{2R}$$

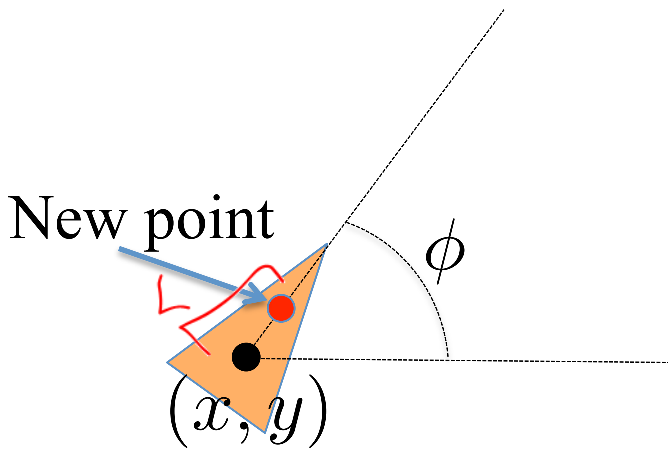
$$v_\ell = \frac{2v - \omega L}{2R}$$

Lecture 7.4 – A Clever Trick

- We can use a layered architecture for making differential drive robots act like $\dot{x} = u$
- Key idea: Plan using the simple dynamics, then track using some clever controller (PID?)

- Today: We can be even more clever!

Transforming the Unicycle



$$\begin{cases} \dot{x} = v\cos\phi \\ \dot{y} = v\sin\phi \\ \cancel{\dot{\phi} = \omega} \end{cases}$$

ignored

- What if we ignored the orientation and picked a different point on the robot as the point we care about?

$$\begin{cases} \tilde{x} = x + \ell\cos\phi \\ \tilde{y} = y + \ell\sin\phi \end{cases}$$

New Dynamics

$$\begin{cases} \dot{x} = v\cos\phi \\ \dot{y} = v\sin\phi \\ \dot{\phi} = \omega \end{cases}$$

New point

$$\begin{cases} \tilde{x} = x + \ell\cos\phi \\ \tilde{y} = y + \ell\sin\phi \end{cases}$$

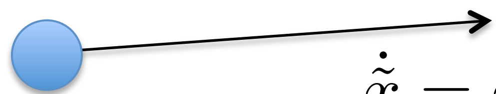
$\dot{x}$ $\dot{\phi}$

$$\dot{\tilde{x}} = \dot{x} - \ell\dot{\phi}\sin\phi \quad = v\cos\phi - \ell\omega\sin\phi$$

$$\dot{\tilde{y}} = \dot{y} + \ell\dot{\phi}\cos\phi \quad = v\sin\phi + \ell\omega\cos\phi$$

New Inputs

- Let's assume that we can control the new point directly



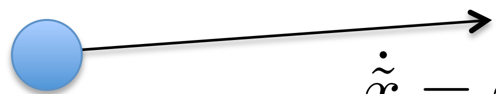
$$\dot{\tilde{x}} = u_1,\ \ \dot{\tilde{y}} = u_2$$

$$\dot{\tilde{x}} = v\cos\phi - \ell\omega\sin\phi = u_1$$
$$\dot{\tilde{y}} = v\sin\phi + \ell\omega\cos\phi = u_2$$

$$\underbrace{\begin{bmatrix}\cos\phi & -\sin\phi\\ \sin\phi & \cos\phi\end{bmatrix}}_{R(\phi)}\underbrace{\begin{bmatrix} v\\ \ell\omega\end{bmatrix}}_{\begin{bmatrix}1&0\\0&\ell\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix}} = \begin{bmatrix}u_1\\ u_2\end{bmatrix}$$

New Inputs

- Let's assume that we can control the new point directly



$$\dot{\tilde{x}} = u_1, \;\; \dot{\tilde{y}} = u_2$$

Rotating matrix

$$R(\phi) \begin{bmatrix} 1 & 0 \\ 0 & \ell \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

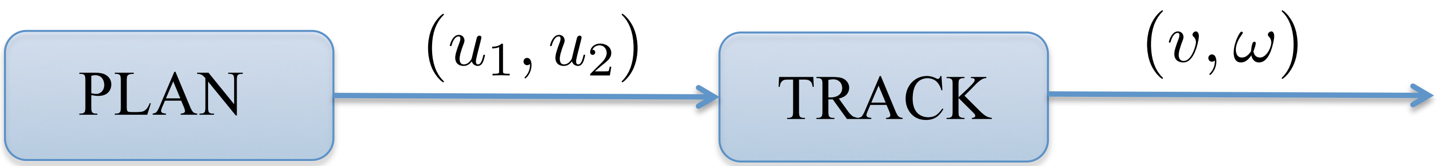
$$\boxed{\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{\ell} \end{bmatrix} R(-\phi) \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}}$$

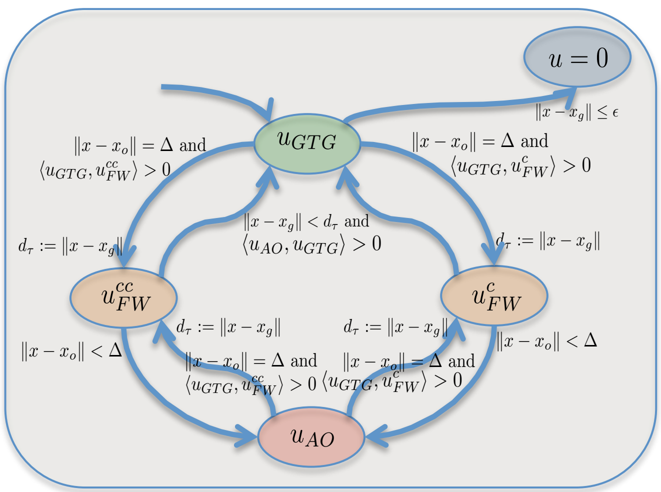
$$R^{-1}(\phi) = R(-\phi)$$

$$\begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{l} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & l \end{bmatrix}^{-1}$$

What's The Point?

- Before:





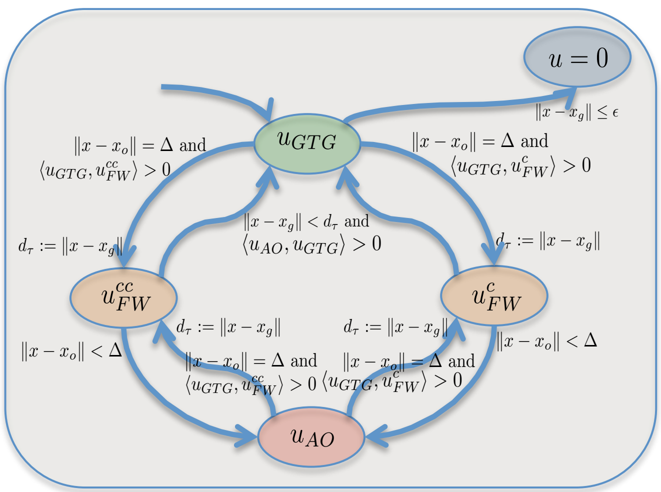
$$e = \phi_d - \phi, \quad \omega = \mathrm{PID}(e)$$

$$v = \|u\|$$

What's The Point?

- Now:

$$\text{PLAN} \xrightarrow{(u_1,u_2)} \text{TRANSFORM} \xrightarrow{(v,\omega)}$$



$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{\ell} \end{bmatrix} R(-\phi) \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

↳ L has to be small

Lecture 7.5 – Other Robot Classes

- Last time: It is indeed possible to make differential drive mobile robots “act” like $\dot{x} = u$
  - If we are willing to ignore orientation
  - And accept a small offset error
- Today: Does this generalize to other types of robots? And, what other types are there?






Other Models

- There are lots and lots of different types of robotic systems
- We cannot cover them all. Instead, we will focus on what they have in common:

UNICYLE

$$\dot{x} = v \cos \phi$$
$$\dot{y} = v \sin \phi$$
$$\dot{\phi} = \omega$$

States: position and orientation
Inputs: Angular and transl. velocities

Other Models

- There are lots and lots of different types of robotic systems
- We cannot cover them all. Instead, we will focus on what they have in common:

CAR-LIKE ROBOT

steering angle

$$\dot{x} = v\cos(\phi + \psi)$$
$$\dot{y} = v\sin(\phi + \psi)$$
$$\dot{\phi} = \frac{v}{\ell}\sin(\psi)$$
$$\dot{\psi} = u$$

States: position, orientation, steering angle
Inputs: Transl. vel. and steering angular vel.

Other Models

- There are lots and lots of different types of robotic systems
- We cannot cover them all. Instead, we will focus on what they have in common:

SEGWAY ROBOT

base: unicycle
pendulum:

$$3(m_w + m_b)\dot{v} - m_b d \cos\phi \ddot{\phi} + m_b d \sin\phi(\dot{\phi}^2 + \dot{\psi}^2) = \frac{1}{R}(\tau_L + \tau_R)$$

$$((3L^2 + \frac{1}{2R^2})m_w + m_b d^2 \sin^2\phi + I_2)\ddot{\psi} + m_b d^2 \sin\phi \cos\phi \dot{\psi}\dot{\phi} = \frac{L}{R}(\tau_L - \tau_R)$$

$$m_b d \cos\phi \dot{v} + (-m_b d^2 - I_3)\ddot{\phi} + m_b d^2 \sin\phi \cos\phi \dot{\psi}^2 + m_b g d \sin\phi = \tau_L + \tau_R$$

States: position, orientation, tilt angle, and velocities
Inputs: Wheel torques

Other Models

- There are lots and lots of different types of robotic systems
- We cannot cover them all. Instead, we will focus on what they have in common:

FIXED-WING AIRCRAFT

$$\boxed{\begin{aligned}\dot{x} &= v\cos(\phi)\\ \dot{y} &= v\sin(\phi)\\ \dot{\phi} &= \omega\end{aligned}}$$

Unicycle

$$\dot{z} = u$$

altitude

States: position, orientation, altitude
Inputs: Transl., angular, vertical vel.

Other Models

- There are lots and lots of different types of robotic systems
- We cannot cover them all. Instead, we will focus on what they have in common:

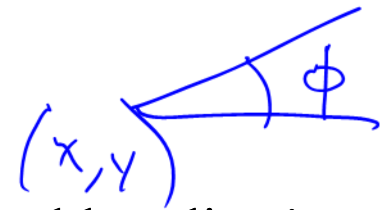
UNDERWATER GLIDER

$$\dot{x} = v\cos(\phi)$$
$$\dot{y} = v\sin(\phi)$$
$$\dot{\phi} = \omega$$
$$\dot{z} = u$$

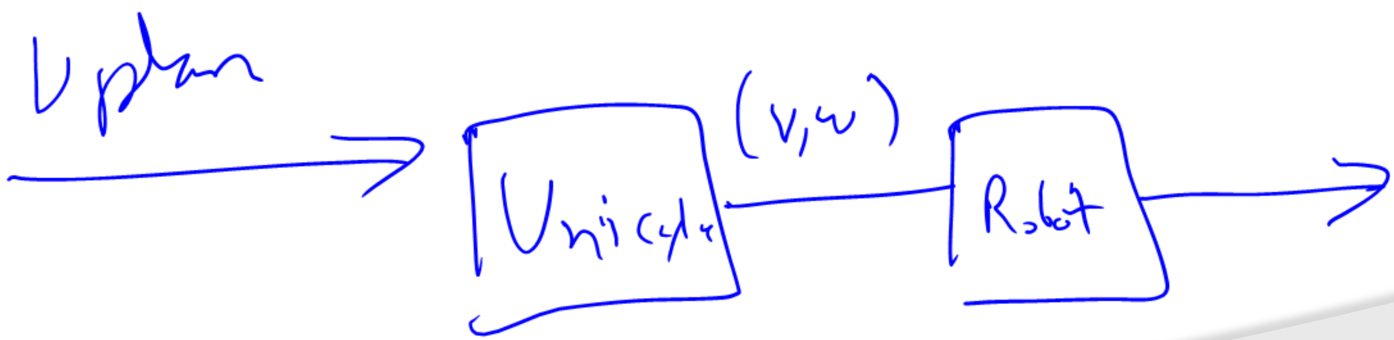
Same as airplane

States: position, orientation, altitude
Inputs: Transl., angular, vertical vel.

Punchline



- Everything (almost) involves POSE = position and heading!
- Everything (almost) with pose is almost a unicycle!
- So we can (almost) use what we have already done and then make the actual model class fit the unicycle – Just add a layer
  - Next lecture: Do this for the car robot!



Adding Constraints

- A lot of times we actually need constraints, which unfortunately make it harder to control the robots (not in this class)

UNICYLE

$$\dot{x} = v \cos \phi$$
$$\dot{y} = v \sin \phi$$
$$\dot{\phi} = \omega$$

DUBINS

$$\dot{x} = v \cos(\phi)$$
$$\dot{y} = v \sin(\phi)$$
$$\dot{\phi} = \omega$$
$$v = 1, \ \omega \in [-1, 1]$$

V_0 $-\omega_{max}$, ω_{max}

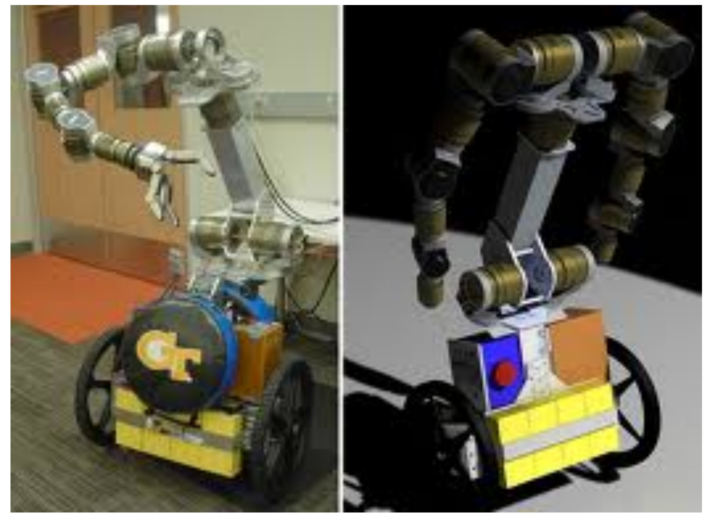
REEDS-SHEPP

$$\dot{x} = v \cos(\phi)$$
$$\dot{y} = v \sin(\phi)$$
$$\dot{\phi} = \omega$$
$$|v| = 1, \ \omega \in [-1, 1]$$

Dubbins vehicle but you can go backwards

When is POSE Not Reasonable?

- Humanoids
- Snakes
- Mobile manipulators



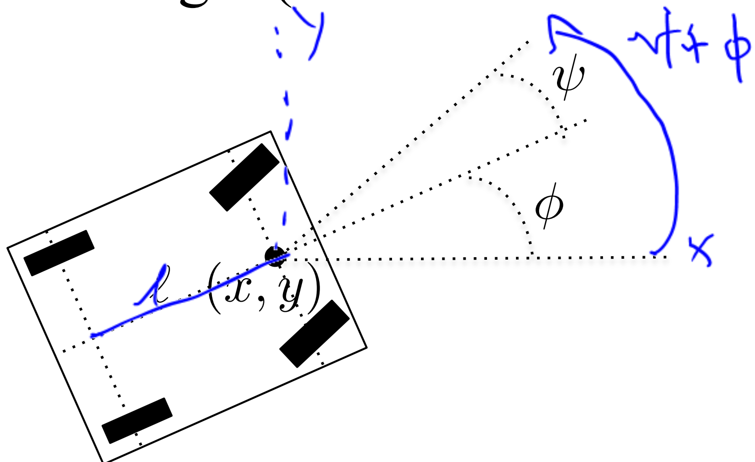
- Next time: Cars (when it is ok…)

Lecture 7.6 – Car-Like Robots

- Claims:
  - Pose (position and heading) is central
  - Other "pose-based" models can be made to look like a unicycle
- Today: Car-like robots

Car Kinematics

- What's different about the car is that it has four wheels.
- Only the front wheels turn, which means that the steering wheel angle ("=" front wheel angle) becomes important



$$\dot{x} = v\cos(\phi + \psi)$$
$$\dot{y} = v\sin(\phi + \psi)$$
$$\dot{\phi} = \frac{v}{\ell}\sin(\psi)$$
$$\dot{\psi} = \sigma$$

states:

(x, y)	position
ϕ	heading
ψ	steering angle

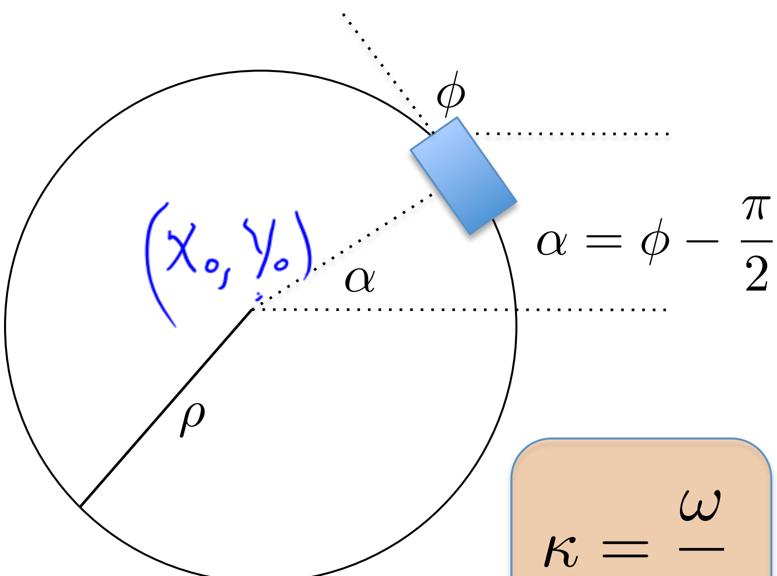
inputs:

v	speed
σ	angular steering velocity

Curvature Control

- How do we make this act like a unicycle?
- Assume a unicycle is driving along a circular arc

$$\dot{x} = v\cos(\phi + \psi)$$
$$\dot{y} = v\sin(\phi + \psi)$$
$$\dot{\phi} = \frac{v}{\ell}\sin(\psi)$$
$$\dot{\psi} = \sigma$$



$$\kappa = \frac{\omega}{v}$$

$$x = x_0 + \rho\cos(\alpha)$$
$$= x_0 + \rho\sin(\phi)$$
$$\dot{x} = \omega\rho\cos(\phi)$$
$$= v\cos(\phi)$$
$$\rho = \frac{v}{\omega}$$

$\dot{x}_0 = 0$

$\sin(\phi) = \phi\cos(\phi)$

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

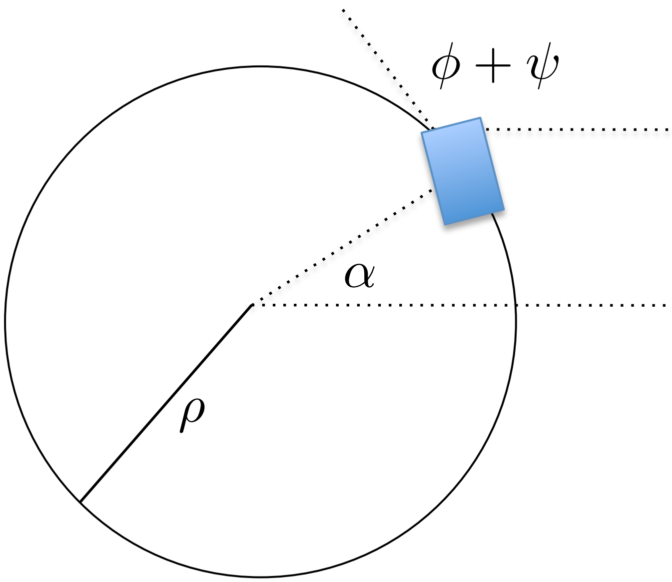
then

$\omega\rho = v$

$\rho = \frac{v}{\omega}$ $K = \frac{1}{\rho}$

Curvature Control

- Let's redo this for the car



$$\alpha = \phi + \psi - \frac{\pi}{2}$$

$$x = x_0 + \rho \sin(\phi + \psi)$$

$$\dot{x} = \rho(\dot{\phi} + \dot{\psi}) \cos(\phi + \psi)$$

$$\dot{\phi} = \frac{v}{\ell}\sin(\psi) \quad \dot{\psi} = 0$$

← since we are driving in a circle

$$= v\cos(\phi + \psi)$$

$$\rho = \frac{\ell}{\sin(\psi)}$$

$$v = \rho\dot{\phi}$$

$$\rho = \frac{v}{\dot{\phi}} = \frac{l}{\sin(\psi)}$$

$$\boxed{\kappa = \frac{\sin(\psi)}{\ell}}$$

$$\kappa = \frac{1}{\rho}$$

Lining Up The Curvatures

UNICYCLE CAR

$$\kappa = \frac{\omega}{v} \qquad \kappa = \frac{\sin(\psi)}{\ell}$$

set them equal

$\psi = \arcsin\left(\frac{\omega}{v}\right)$

$$\sin(\psi) = \frac{\omega\ell}{v} \quad \Rightarrow \psi_d = \arcsin\left(\frac{\omega\ell}{v}\right)$$

But we can actually stay with sinus instead of dealing with arcsin!

An Almost P-Regulator

$$\dot{\psi} = \sigma$$

$$\sigma = C(\psi_d - \psi)$$

$$\sigma = C\left(\frac{\omega\ell}{v} - \sin(\psi)\right)$$

$\sin(\psi_d)$ $\sin\psi$

$\shortparallel$

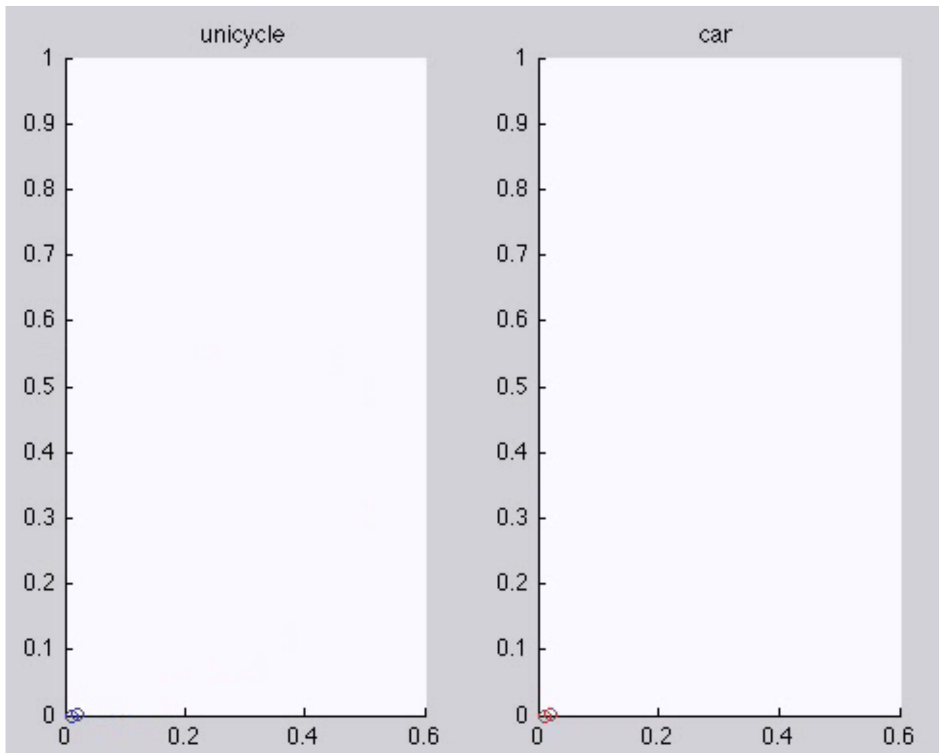
$\frac{\omega\ell}{v}$

An Almost P-Regulator

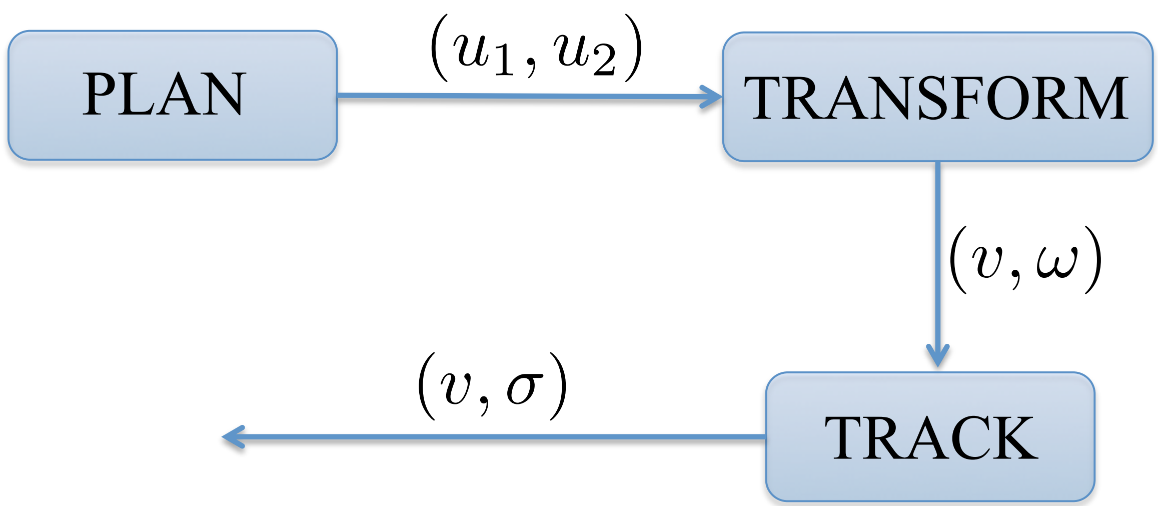
$$\dot{\psi} = \sigma$$

$$\cancel{\sigma = C(\psi_d - \psi)}$$

$$\sigma = C\left(\frac{\omega \ell}{v} - \sin(\psi)\right)$$



Summing It Up



$$\sigma = C\left(\frac{\omega\ell}{v} - \sin(\psi)\right)$$

Lecture 7.7 – To Probe Further

- Believe it or not – there are lots of things not covered in this course!

Nonlinear and Optimal Control

AI, machine learning w

$\dot{x} = Ax + Bu$

↳ linear

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

$$\min_u \int_0^T L(x(t), u(t))dt + \Psi(x(T))$$

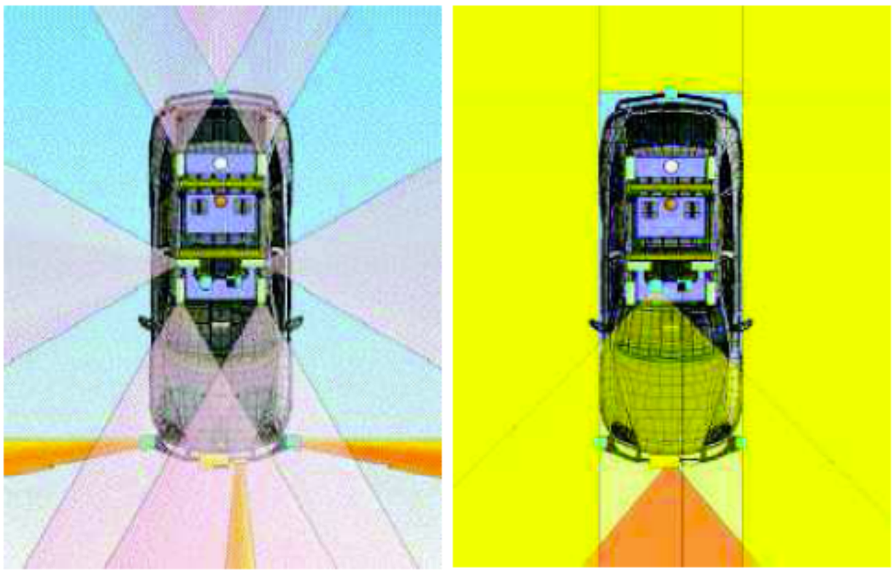
stability
Performance
Robustness
Cost (Optimal Control)

Machine Learning

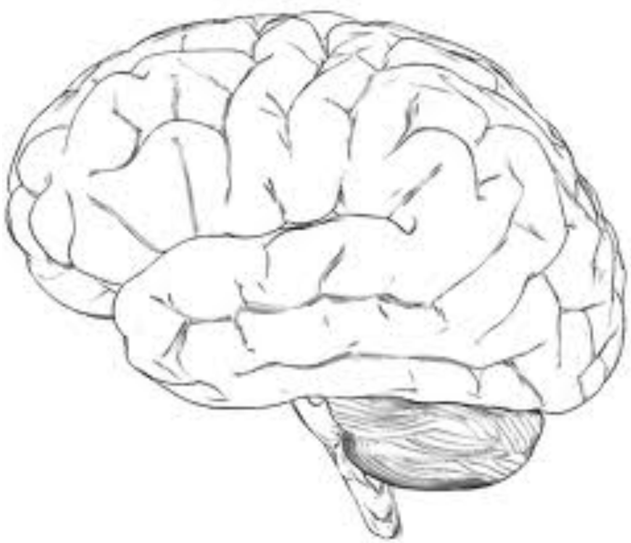
$$V^{\pi}(x_0) = \sum_{k=0}^{\infty} \gamma^k c(x_k, \pi(x_k))$$

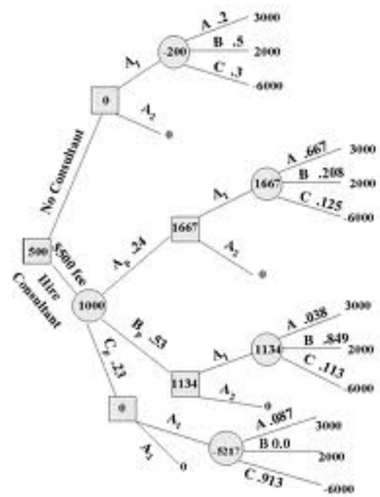
$$V^{\star}(x) = \min_u \{c(x,u) + \gamma V^{\star}(f(x,u))\}$$

Perception and Mapping



High-Level AI



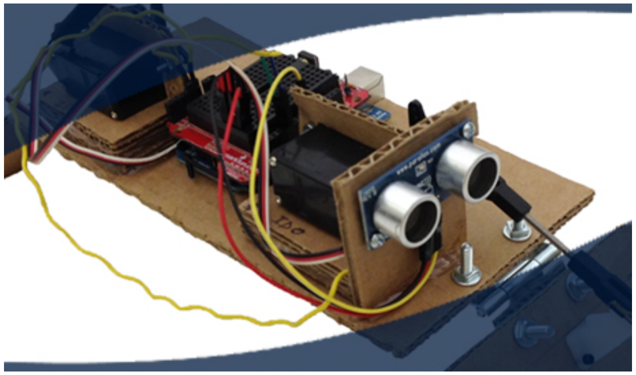


To Probe Further

- Not only are there things not covered in the class, there are lots of things we don't know yet!

Lecture 7.8 – In Conclusion

- *That's it folks!*



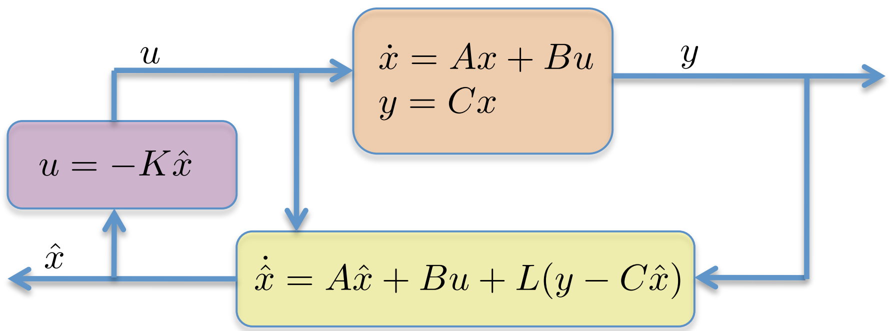
- Ambition with the course:
  - Learn how to make mobile robots move in effective and safe ways using modern control theory
  - Appreciate the value of systematic thinking/design
  - Bridge the theory-practice gap
  - Have fun and spark further investigations

High-Level Punchline #1 – The Model

- Without a model, we cannot say much about how the system will behave:
  - Need models to predict behavior forward in time
  - Need models to be able to derive control laws in a systematic manner
- The model should be rich enough to be relevant yet simple enough to be useful
- Bananas vs. Non-bananas

High-Level Punchline #2 – Feedback

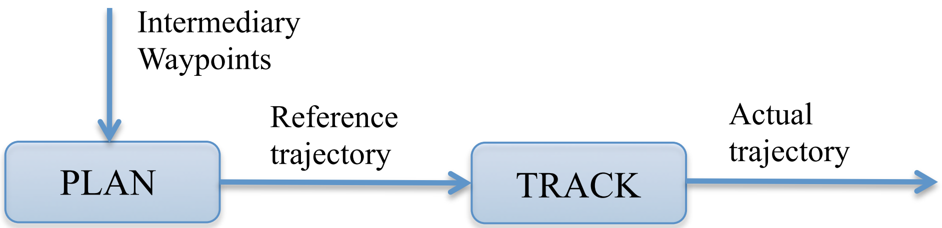
- Given a model, feedback control should be used to make the system behave the way we want it to (if possible)

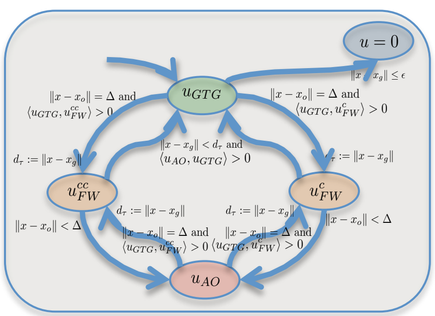


- Stability, Tracking, Robustness
- State feedback and observers

High-Level Punchline #3 – Architectures

- Plan for simple systems, execute on the "real" system



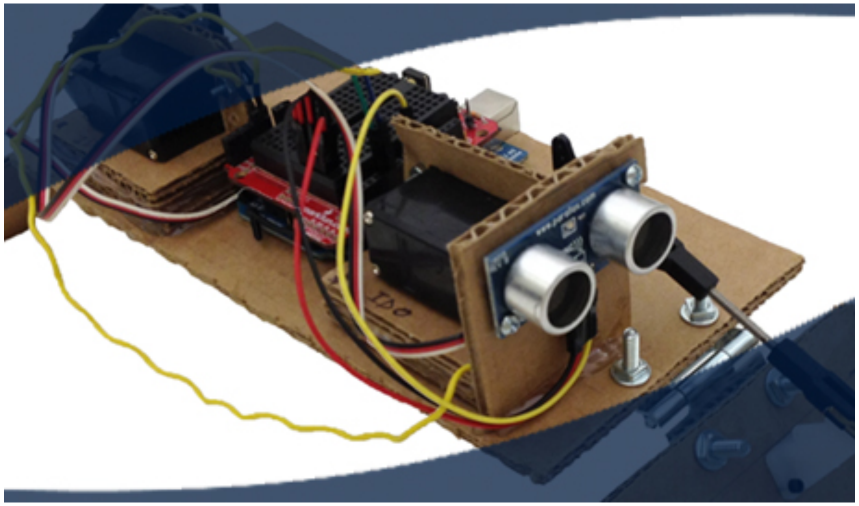


$$\dot{x} = f(x, u), \;\; u = g(x, r)$$

- Three abstraction layers
- Differential-drive robots

High-Level Punchline #4 – Whatever…

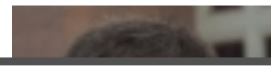
- Don't take my word for it
- Experiment and tweak
- The field is certainly not done yet



THANKS!

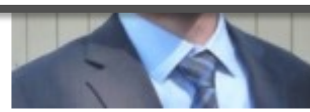
Amy LaVi

g Droge



All of you!

Smriti Chopra



Rowland O'flaherty