Autonomous Robotics
Winter 2024
Abhishek Gupta
TAs: Karthikeya Vemuri, Arnav Thareja
Marius Memmel, Yunchu Zhang

Slides borrowed from many sources – Sidd Srinivasa, Sanjiban Choudhury, Russ Tedrake
Class Outline

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

Control
- Feedback Control
- PID Control
- MPC
- LQR

Planning
- Search
- Heuristic Search

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

Motion Planning
Lazy Search
Logistics

- HW 2 due on Feb 2
- Reading responses due on Ed by Monday Jan 29
- Start hardware HW early!
- Suggest guest lectures early 😊
Lecture Outline

Where does control fit in the roadmap

Why control problems are hard

How to formulate control problems
Let’s zoom back out

Our knowledge of how the world works (Modeling)

World Model

Planning

Helicopter Models

How do we decide how to act to accomplish task? (Planning/Control)

Control

How do we decide where we are in the world? (Estimation)

Sensors

Robot interacts with environment

Laser
Camera
GPS

Actuators

Stick
Lever
Pedals
The Sense-Plan-Act Paradigm

Estimate robot state

Plan sequence of motions

Control robot to follow plan

Solved over last 3 weeks

Assume to be solved for now

??
From perception to control ...
When I think about control …
What is Control?

“PLAN”  →  CONTROL  →  ACTUATOR COMMANDS
What is a Plan?

Can express this problem as **tracking a reference** trajectory

\[ x(t), y(t), \theta(t) \]
Why Feedback Control?

What if we send out controls $u(t)$ from kinematic car model?

Open-loop control leads to **accumulating errors**!
1. Measure error between reference and current state.
2. Take actions to minimize this error.
Useful to think of control laws as vector fields

\[ x(t), y(t), \theta(t) \]
Is this still a research problem?
Assumptions made by such controllers

1. Fully actuated: There exists an inverse mapping from reference to control actions

\[ \sigma(t) \rightarrow u(t) \]

2. Almost no execution error or state estimation error

3. Enough control authority to clamp down errors / overcome disturbances
The Atlas robot hard at … play?

https://www.youtube.com/watch?v=fRj34o4hN4I
Challenge 1: Underactuated systems

Fully actuated: There exists an inverse mapping from reference to control actions

\[ \sigma(t) \rightarrow u(t) \]

We don’t have full authority to move the system along arbitrary trajectories
Challenge 1: Underactuated systems

What affects the error between robot state and reference?

Some initial motor thrust ... Whole lot of gravity!

... some precise control adjustments
Question: If we know the model of our robot, can’t we solve a huge optimization problem to figure out control?
Doing backflips with a helicopter

Redbull Eurocopter BO-105

https://www.youtube.com/watch?v=RGu45s1_QPU
And just what is this model?!?

Unpredictable drag forces!

Chaotic vortex around blades!

Nothing countering gravity!

Hopeless to assume we know exactly how the helicopter will behave upside down...
Challenge 2: Choosing good closed-loop models

Closed-loop system = Point mass with a planar thrust vector

Chaotic dynamics + Feedback control law = Well-behaved system
Challenge 3: Model changing on the fly!

Run real-time estimators for wheel characteristics

Need control laws for all possible model parameters
Ok let’s control racecars!
Reference Parameterizations

Option 1: **Time**-parameterized trajectory

*Pro:* Useful if we want the robot to respect time constraints

*Con:* Sometimes we only care about deviation from reference
Option 2: **Index**-parameterized geometric path (untimed)

Pro: Useful for conveying shape for the robot to follow
Con: Can’t control when robot will reach a point
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 2: Pick a reference (desired) state

\[
\begin{bmatrix}
x_{\text{ref}} \\
y_{\text{ref}} \\
\theta_{\text{ref}}
\end{bmatrix}
\]

\[
\begin{bmatrix}
x \\
y \\
\theta
\end{bmatrix}
\]
How do we choose a reference state?

For an index-parameterized path, there are multiple options.

Closest point

$$\tau_{\text{ref}} = \arg \min_\tau \| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x(\tau) \\ y(\tau) \end{bmatrix} \|$$
How do we choose a reference state?

For an index-parameterized path, there are multiple options.

Lookahead

\[ \tau_{\text{ref}} = \arg \min_{\tau} \left( \| [x \quad y]^\top - [x(\tau) \quad y(\tau)]^\top \| - \ell \right)^2 \]
Step 3: Compute error to reference state

Along-track error \( e_{at} \)

Cross-track error \( e_{ct} \)

Heading error \( \theta_e \)
Step 3: Compute error to reference state

\[
\begin{bmatrix}
x \\
y \\
\theta
\end{bmatrix}_G \rightarrow \begin{bmatrix}
e_{at} \\
e_{ct} \\
\theta_e
\end{bmatrix}_{\text{ref}}
\]
Aside: Rotation Matrices (Plane)

\[ R = R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \]
Step 3: Compute error to reference state
Step 3: Compute error to reference state

\[ A_e = \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{ref} \\ y_{ref} \end{bmatrix} \]
Step 3: Compute error to reference state

We want position in frame B

\[ B_e = A \begin{bmatrix} R \\ A \end{bmatrix} A e = R(-\theta_{ref}) \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{ref} \\ y_{ref} \end{bmatrix} \]

(rotation of A w.r.t B)

(rotation of A w.r.t B)
Step 3: Compute error to reference state

We want position in frame B

\[
B_e = \begin{bmatrix} e_{at} \\ e_{ct} \end{bmatrix} = \begin{bmatrix} \cos(\theta_{ref}) & \sin(\theta_{ref}) \\ -\sin(\theta_{ref}) & \cos(\theta_{ref}) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_{ref} \\ y_{ref} \end{bmatrix}
\]
Step 3: Compute error to reference state

\[ \theta_e = \theta - \theta_{ref} \]
Step 3: Compute error to reference state

\[ e_{at} = \cos(\theta_{ref})(x - x_{ref}) + \sin(\theta_{ref})(y - y_{ref}) \]

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

\[ \theta_e = \theta - \theta_{ref} \]
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.

\[ u = K(e) \]
Different Control Laws

- Proportional-integral-derivative (PID) control
- Pure-pursuit control
- Model-predictive control (MPC)
- Linear-quadratic regulator (LQR)
- And many many more!
Bang-bang control

Simple control law - choose between hard left and hard right

\[ u = \begin{cases} 
  u_{max} & \text{if } e_{ct} < 0 \\
  -u_{max} & \text{otherwise}
\end{cases} \]
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! Come back on Monday to find out more