Lecture 16: Planning III (Decision-Making I)

Professor Katie Driggs-Campbell

April 9, 2024

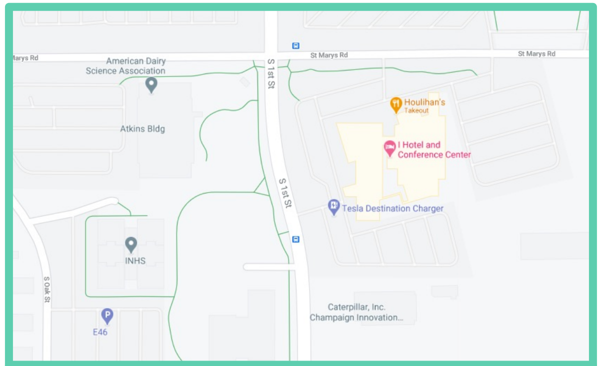
ECE484: Principles of Safe Autonomy

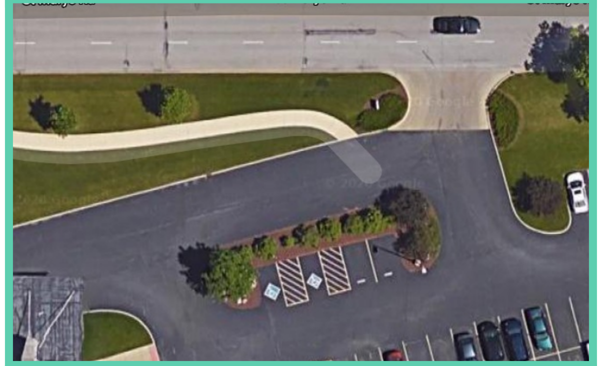
Administrivia

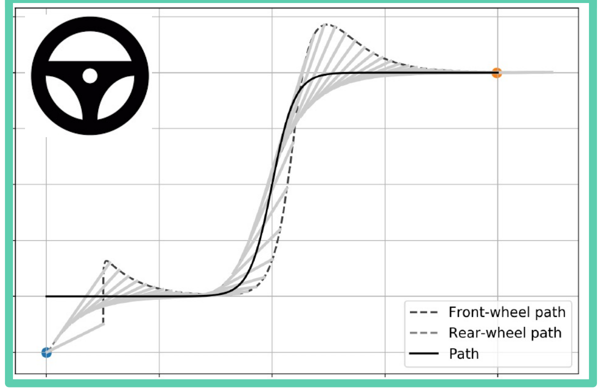
- Upcoming due dates:
  - Final Presentations in class on 4/23 and 4/25
  - Final Video due 5/3
- Exam on 4/18 at 7pm
  - Email me ASAP about conflict exams
  - Make reservation in testing center for DRES accommodations
  - Practice questions will be posted on CampusWire today
  - No cheatsheet will be needed
  - CA review session on Friday 4/19
  - In-class review session on Tuesday 4/16
- Prof. DC OH by appointment next week (4/16)
  - Otherwise in 260 CSL

Typical planning and control modules

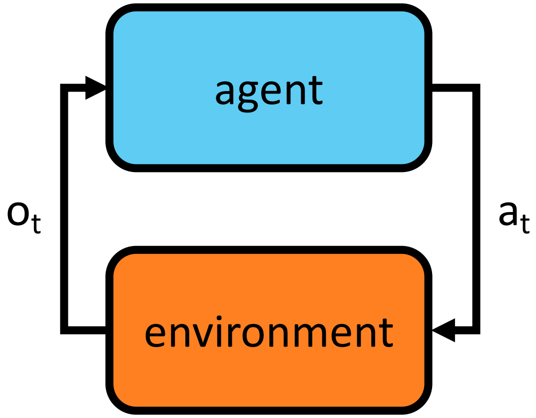
- Global navigation and planner
  - Find paths from source to destination with static obstacles
  - Algorithms: Graph search, Dijkstra, Sampling-based planning
  - Time scale: Minutes
  - Look ahead: Destination
  - Output: reference center line, semantic commands
- Local planner
  - Dynamically feasible trajectory generation
  - Dynamic planning w.r.t. obstacles
  - Time scales: 10 Hz
  - Look ahead: Seconds
  - Output: Waypoints, high-level actions, directions / velocities
- Controller
  - Waypoint follower using steering, throttle
  - Algorithms: PID control, MPC, Lyapunov-based controller
  - Lateral/longitudinal control
  - Time scale: 100 Hz
  - Look ahead: current state
  - Output: low-level control actions

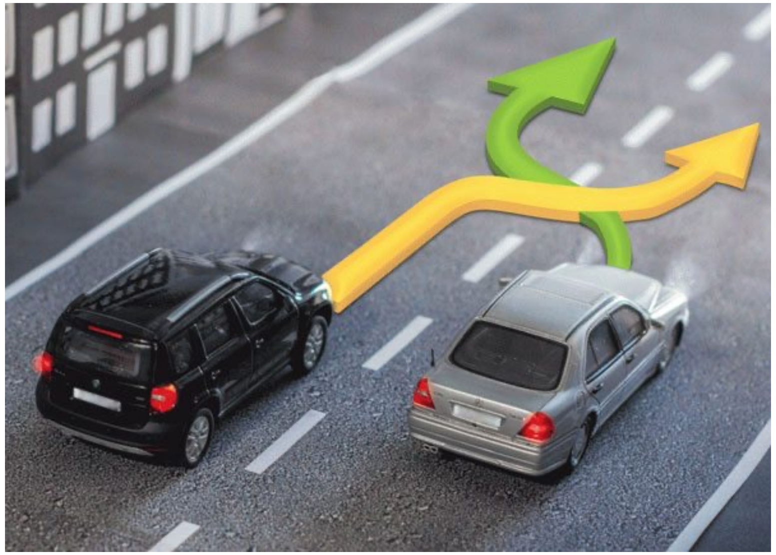


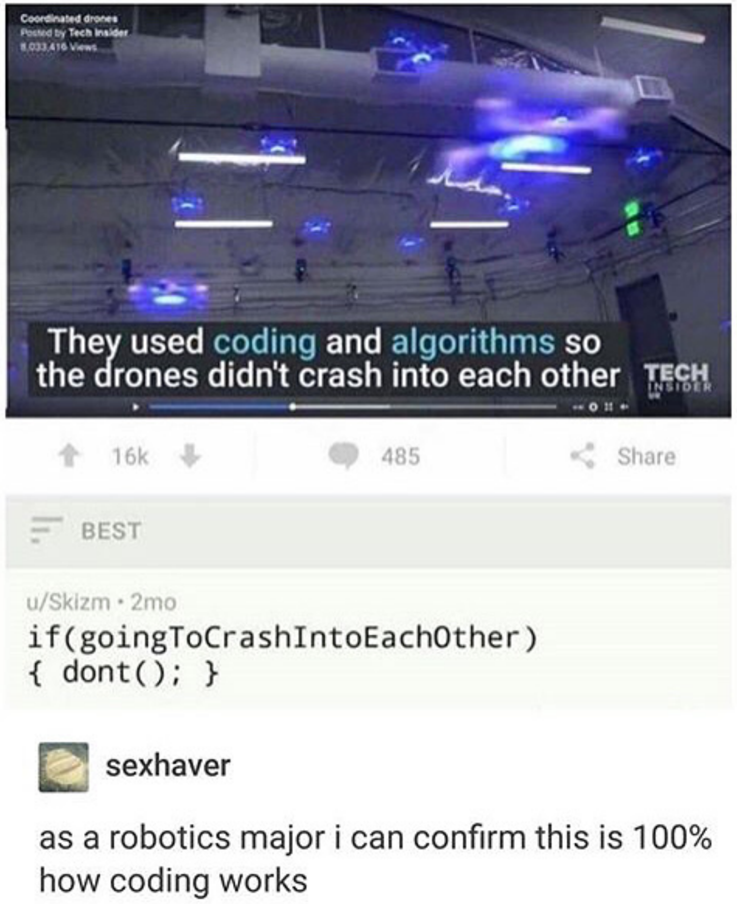




High-Level Decision-Making

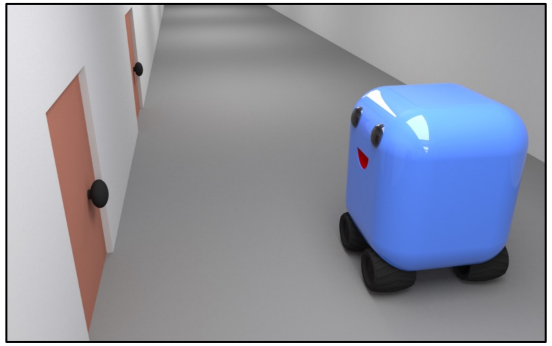


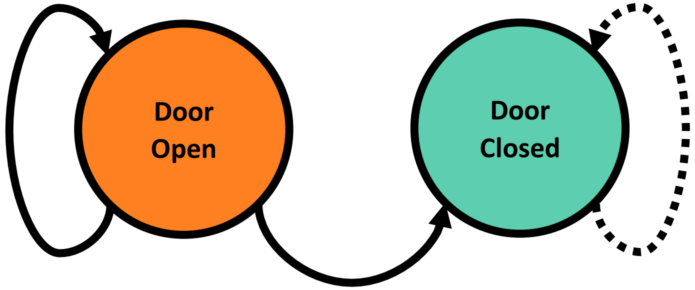




From Filtering to Decision-Making

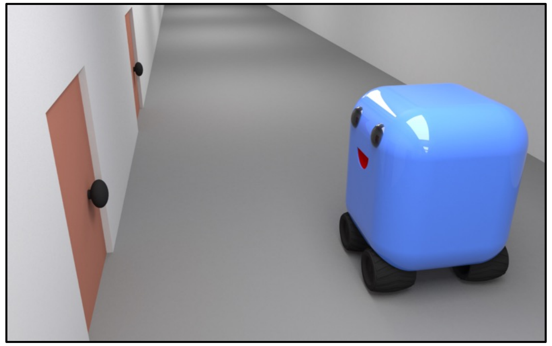
Recall: Filtering allows us to recursively update our belief about some state

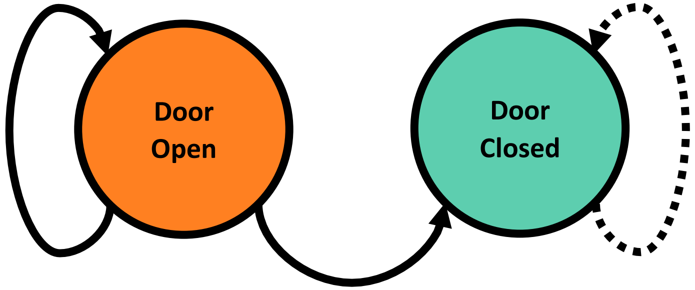


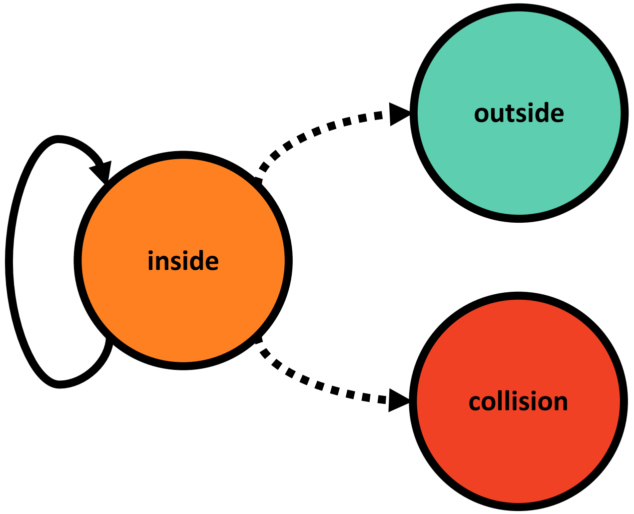


From Filtering to Decision-Making

Recall: Filtering allows us to recursively update our belief about some state







Decision-making helps us reason about what actions we should take

Today's Plan

- Possible solutions for decision-making
- Markov Decision Processes
- MDP Policies and Value Iteration

Today's Plan

- Possible solutions for decision-making
- Markov Decision Processes
- MDP Policies and Value Iteration

Decision-Making Methods

1. Explicit programming
   - Ex: if/then statements
   - → Heavy burden on designer

Heuristic Method for Lane Changing: MOBIL

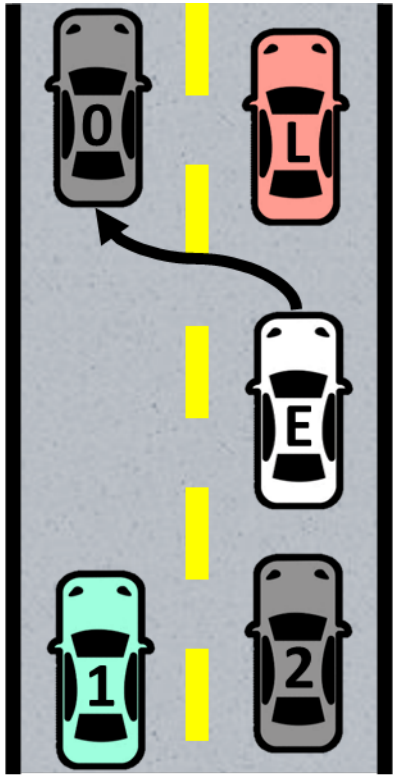
- Safety criterion:

$$\tilde{a}_E \geq -b_{safe}$$

- Decision rule:

$$\tilde{a}_E - a_E + p(\tilde{a}_1 - a_1 + \tilde{a}_2 - a_2) > \Delta a_{th}$$

- Politeness factor, p: 0.35
- Safe braking limit, b_{safe}: $2\ {}^{m}/_{s^2}$
- Acceleration threshold: $0.1\ {}^{m}/_{s^2}$
- Look-ahead horizon: $30m$



Decision-Making Methods

1. Explicit programming
   - Ex: if/then statements

   → Heavy burden on designer
2. Supervised learning
   - Ex: imitation learning

   → Generalizing is often a challenge
3. Optimization / optimal control
   - Ex: MPC

   → Requires a high-fidelity model and lots of computation
4. Planning
   - Given a stochastic model, how to algorithmically determine best policy?
5. Reinforcement Learning
   - If model is unknown (or very complex), learn policy through experience

Today's Plan

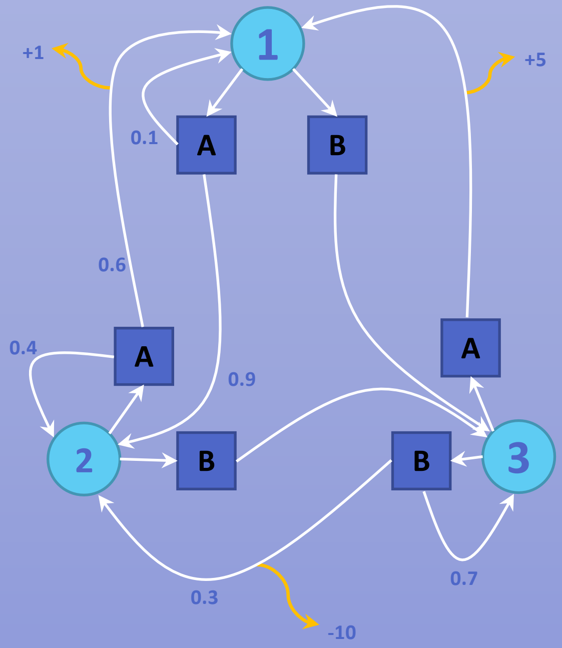
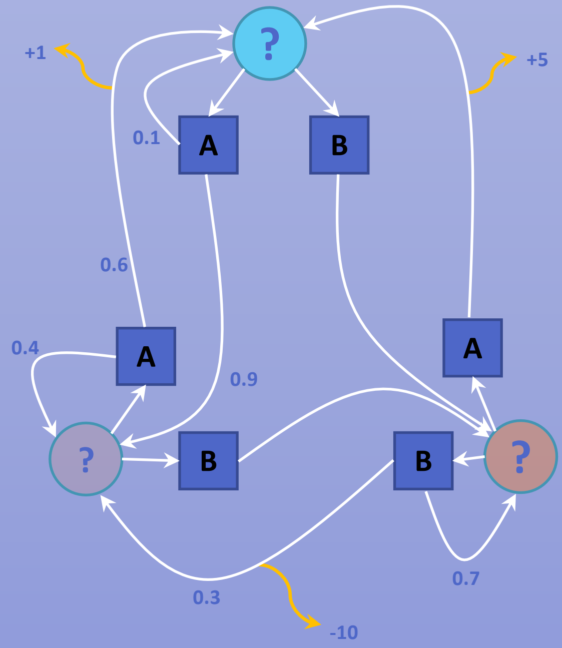
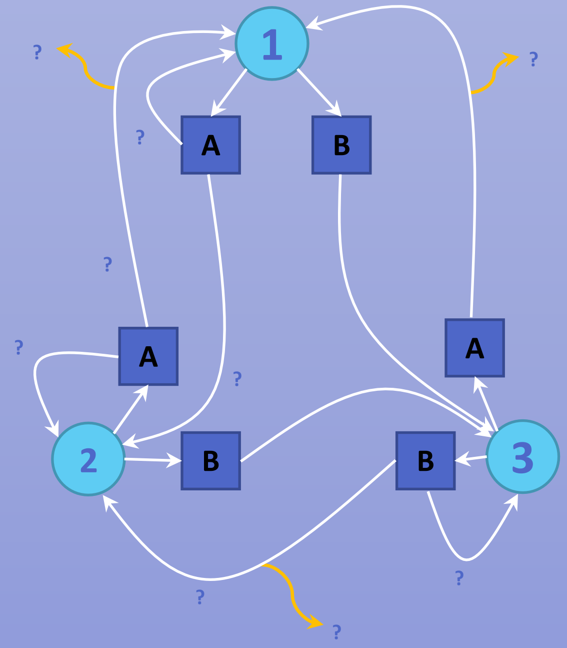
- Introduction to decision-making
- Markov Decision Processes
- MDP Policies and Value Iteration
- Simple Example

Markov Decision Processes (MDPs)

Uncertainty in Motion

- **Markov Decision Processes (MDPs)** model the AV and environment assuming full observability
  - $P(z|x)$: *deterministic* and bijective
  - $P(x'|x,u)$: may be nondeterministic
  - Must incorporate uncertainty into the planner and generate actions for each state
- A policy for action selection is defined for all states

Markov Models

<table>
<tr><th>Markov Decision Processes (MDP)</th><th>Partially Observable MDP</th><th>Reinforcement Learning</th></tr>
<tr><td>
</td><td>
</td><td>
</td></tr>
<tr><td>Uncertainty in effects of actions</td><td>Uncertainty in current state</td><td>Uncertainty in model</td></tr>
</table>

Markov Assumptions and Common Violations

Markov Assumption postulates that past and future data are independent if you know the current state.

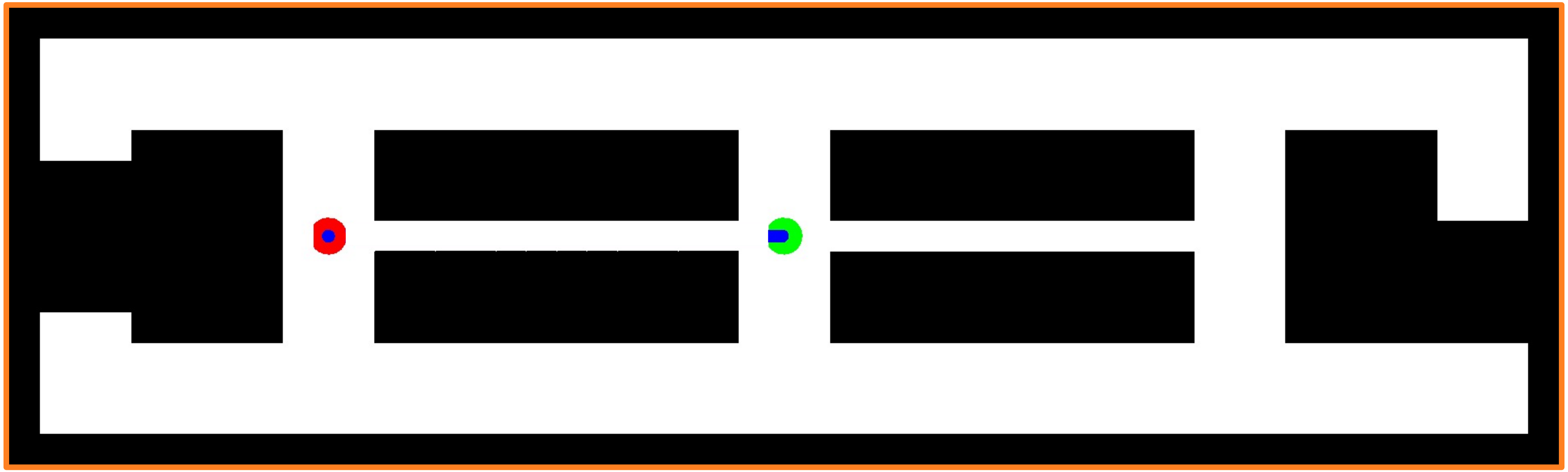
Markov Assumptions and Common Violations

Markov Assumption postulates that past and future data are independent if you know the current state.

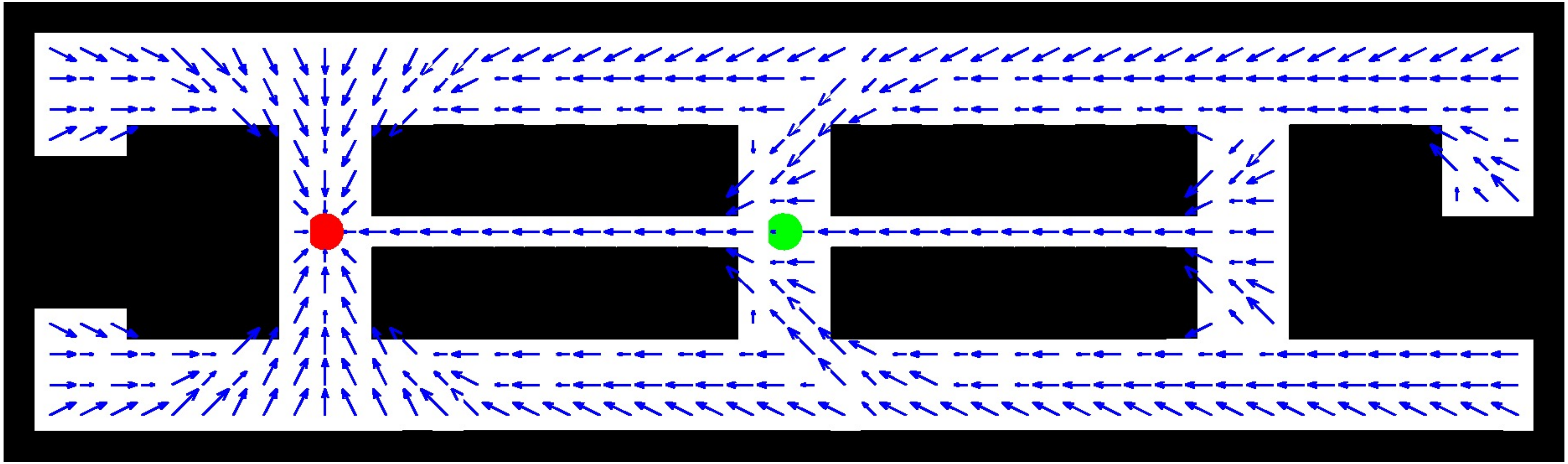
What are some common violations?

- Unmodeled dynamics in the environment not included in state
  - E.g., moving people and their effects on sensor measurements in localization
- Inaccuracies in the probabilistic model
  - E.g., error in the map of a localizing agent or incorrect model dynamics
- Approximation errors when using approximate representations
  - E.g., discretization errors from grids, Gaussian assumptions
- Variables in control scheme that influence multiple controls
  - E.g., the goal or target location will influence an entire sequence of control commands

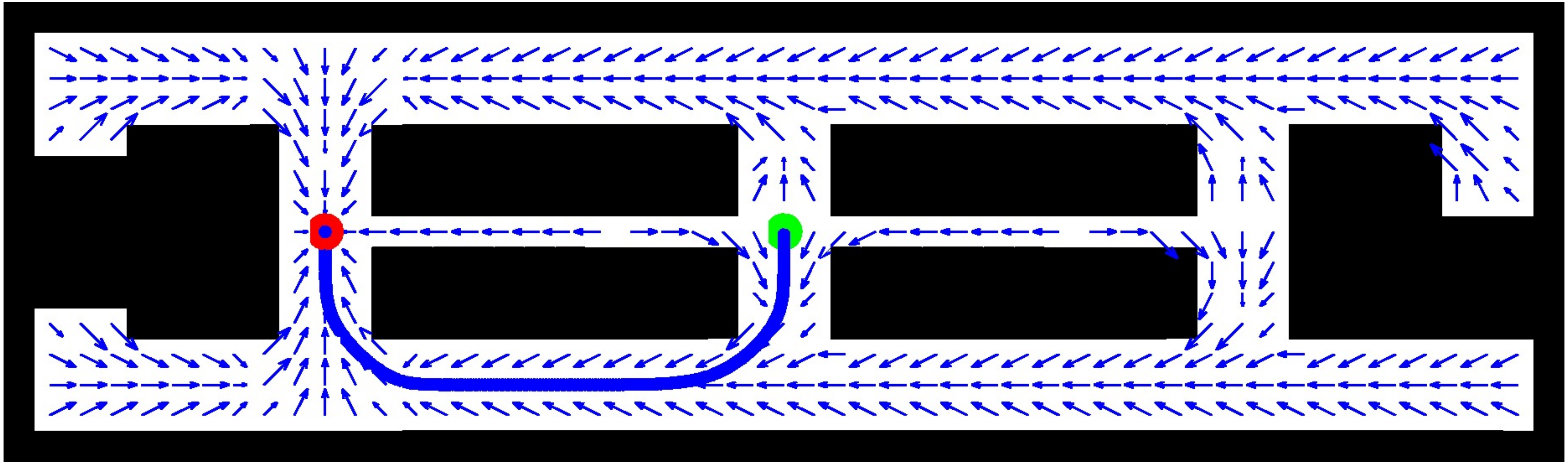
Grid World Example



Deterministic actions



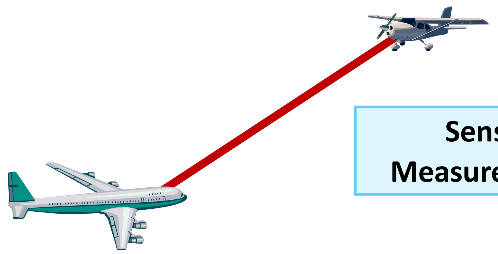
Nondeterministic actions



Defining Values

- Actions are driven by goals
  - E.g., reach destination, stay in lane
- Often, we want to reach goal while optimizing some cost
  - E.g., minimize time / energy consumption, obstacle avoidance
- We express both costs and goals in a single function, called the payoff function

Traffic Alert and Collision Avoidance System (TCAS)



Sensor Measurements

```
IF (ITF.A LT G.ZTHR)
 THEN IF(ABS(ITF.VMD) LT G.ZTHR)
  THEN SET ZHIT;
  ELSE CLEAR ZHIT;
 ELSE IF (ITF.ADOT GE P.ZDTHR)
  THEN CLEAR ZHIT
  ELSE
   ITF.TAUV = -ITF.A/ITF.ADOT;
   IF (ITF.TAUV LT TVTHR AND
    ((ABS(ITF.VMD) LT G.ZTHR) OR
    (ITF.TAUV LT ITF.TRTRU))
    THEN SET ZHIT
    ELSE CLEAR ZHIT
IF (ZHIT EQ $TRUE AND
 ABS(ITF.ZDINT) GT P.MAXZDINT
 THEN CLEAR ZHIT
```

Resolution Advisory

Surveillance

Advisory Logic

Display

Slide Credit: Mykel Kochenderfer

ACAS X: Simplified MDP

Own aircraft
ACAS X

Intruder Aircraft

State space	Action space
• Relative altitude • Own vertical rate • Intruder vertical rate • Time to lateral NMAC • State of advisory	• Clear of conflict • Climb > 1500 ft/min • Climb > 2500 ft/min • Descend > 1500 ft/min • Descend > 2500 ft/min
Dynamic model	**Reward model**
• Head-on, constant closure • Random vertical acceleration • Pilot response delay (5 s) • Pilot response strength (1/4 g) • State of advisory	• NMAC (-1) • Alert (-0.01) • Reversal (-0.01) • Strengthen (-0.009) • Clear of conflict (0.0001)

ACAS X: Simplified MDP

State space	Action space
• Relative altitude • Own vertical rate • Intruder vertical rate • Time to lateral NMAC • State of advisory	• Clear of conflict • Climb > 1500 ft/min • Climb > 2500 ft/min • Descend > 1500 ft/min • Descend > 2500 ft/min

Dynamic model	Reward model
• Head-on, constant closure • Random vertical acceleration • Pilot response delay (5 s) • Pilot response strength (1/4 g) • State of advisory	• NMAC (-1) • Alert (-0.01) • Reversal (-0.01) • Strengthen (-0.009) • Clear of conflict (0.0001)

Today's Plan

- Possible solutions for decision-making
- Markov Decision Processes
- MDP Policies and Value Iteration

Decision-Making Policies

- We want to devise a scheme that generates actions to optimize the future payoff *in expectation*

Decision-Making Policies

- We want to devise a scheme that generates actions to optimize the future payoff *in expectation*
- Policy: $\pi : x_t \rightarrow u_t$
  - Maps states to actions
  - Can be low-level reactive algorithm or a long-term, high-level planner
  - May or may not be deterministic

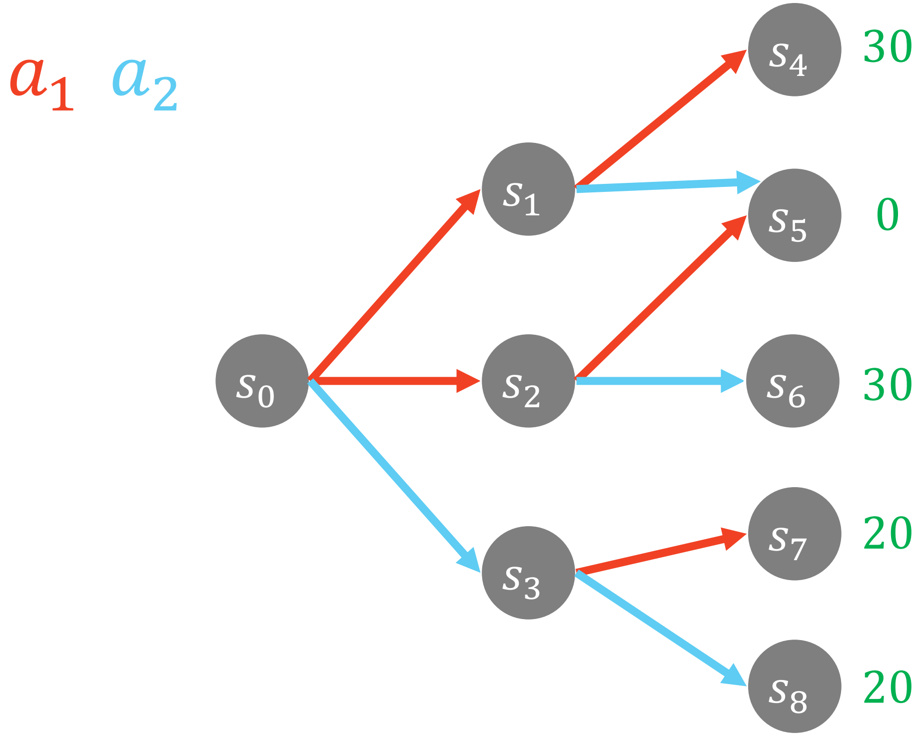
Decision-Making Policies

- We want to devise a scheme that generates actions to optimize the future payoff *in expectation*
- Policy: $\pi : x_t \rightarrow u_t$
  - Maps states to actions
  - Can be low-level reactive algorithm or a long-term, high-level planner
  - May or may not be deterministic
- Typically, we want a policy that optimizes <u>future</u> payoff, considering optimal actions over a <u>planning (time) horizon</u>

Open vs. Closed Loop Planning

- Closed-Loop Planning: accounts for future information in planning. This creates a reactive plan (policy) that can react to different outcomes over time
- Open-Loop Planning: path panning algorithms develop a static sequence of actions

Open Loop vs. Closed Loop Planning



MDP Policies

- Policies map states to actions

$$\pi : x \rightarrow u$$

- We want to find a policy that maximizes future pay off
  - Suppose $T = 1$: $\pi_1(x) = \text{argmax}_u \; r(x, u)$

MDP Policies

- Policies map states to actions

  $$\pi: x \rightarrow u$$

- We want to find a policy that maximizes future pay off
  - Suppose $T = 1$: $\pi_1(x) = \text{argmax}_u \; r(x, u)$
- We write the Value Function for given π:

  $$V_1(x) = \gamma \max_u r(x, u)$$

- Generally, we want to find the sequence of actions that optimize the *expected cumulative discounted future payoff*

Expected Cumulative Payoff

$$R_T = \mathbb{E}\left[\sum_{\tau=0}^{T} \gamma^{\tau}\, r_{t+\tau}\right]$$

Expected Cumulative Payoff

$$R_T = \mathbb{E}\left[\sum_{\tau=0}^{T} \gamma^{\tau} r_{t+\tau}\right]$$

1. Greedy case: $T = 1$

   → Optimize next payoff

Expected Cumulative Payoff

$$R_T = \mathbb{E}\left[\sum_{\tau=0}^{T} \gamma^{\tau} r_{t+\tau}\right]$$

1. Greedy case: $T = 1$

   → Optimize next payoff

2. Finite Horizon: $1 \leq T < \infty, (\gamma \leq 1)$

   → Optimize R_T for set time window

Expected Cumulative Payoff

$$R_T = \mathbb{E}\left[\sum_{\tau=0}^{T} \gamma^{\tau} r_{t+\tau}\right]$$

1. Greedy case: $T = 1$
   → Optimize next payoff
2. Finite Horizon: $1 \leq T < \infty, (\gamma \leq 1)$
   → Optimize R_T for set time window
3. Infinite Horizon: $T = \infty, (\gamma < 1)$
   → Optimize R_∞ for all time
   If $|r| \leq r_{max}$, discounting guarantees R_∞ is finite

$$R_\infty \leq r_{max} + \gamma r_{max} + \gamma^2 r_{max} + \cdots = \frac{r_{max}}{1-\gamma}$$

Value Functions

For longer time horizons (T), we define V(x) recursively:

Recall: $V_1(x) = \gamma \max_u r(x, u)$

Value Functions

- In the infinite time horizon, we tend to reach equilibrium:

$$V_\infty(x) = \gamma \max_u \left[r(x,u) + \int V_\infty(x') p(x'|x,u)\, dx' \right]$$

- This is the *Bellman Equation*
  - Satisfying this is necessary and sufficient for an optimal policy

Computing the (Approximate) Value Function

- Initial guess for $\hat{V}$
  - $\hat{V}(x) \leftarrow r_{min}, \forall x$

Computing the (Approximate) Value Function

- Initial guess for $\hat{V}$
  - $\hat{V}(x) \leftarrow r_{min}, \forall x$
- Successively update for increasing horizons
  - $\hat{V}(x) \leftarrow \gamma \max_u \left[r(x,u) + \int \hat{V}(x') p(x' | x, u) dx' \right]$
- Value iteration converges if $\gamma < 1$

Computing the (Approximate) Value Function

- Initial guess for $\hat{V}$
  - $\hat{V}(x) \leftarrow r_{min}, \forall x$
- Successively update for increasing horizons
  - $\hat{V}(x) \leftarrow \gamma \max_u \left[r(x,u) + \int \hat{V}(x')p(x'|x,u)dx' \right]$
- Value iteration converges if $\gamma < 1$
- Given estimate $\hat{V}(x)$, policy is found:
  - $\pi(x) = \text{argmax}_u \left[r(x,u) + \int \hat{V}(x')p(x'|x,u)dx' \right]$

Computing the (Approximate) Value Function

- Initial guess for $\hat{V}$
  - $\hat{V}(x) \leftarrow r_{min}, \forall x$
- Successively update for increasing horizons
  - $\hat{V}(x) \leftarrow \gamma \max_u \left[r(x,u) + \int \hat{V}(x')p(x'|x,u)dx' \right]$
- Value iteration converges if $\gamma < 1$
- Given estimate $\hat{V}(x)$, policy is found:
  - $\pi(x) = \mathrm{argmax}_u \left[r(x,u) + \int \hat{V}(x')p(x'|x,u)dx' \right]$
- Often, we use the discrete version:
  - $\pi(x) = \mathrm{argmax}_u \left[r(x,u) + \sum_x' \hat{V}(x')p(x'|x,u) \right]$

Summary

- Discussed a different form of planning (often referred to as **decision-making**) schemes and how they fit into the AV stack
- Defined the **MDP** model for decision-making, including **goals, costs, payoff,** and **policies**
- Defined **Expected Cumulative Payoff,** which plays a key role in **optimizing actions over planning horizons**
- **Next time:**
  - Examples of computing policies for MDPs
  - Course wrap-up