CS 287 Advanced Robotics (Fall 2019)
Lecture 15
Partially Observable Markov Decision Processes
(POMDPs)

Pieter Abbeel
Introduction to POMDPs
  - Formalism
  - Exact (usually impractical) solution

Locally Optimal Solutions for POMDPs
  - Trajectory Optimization in (Gaussian) Belief Space
  - Accounting for Discontinuities in Sensing Domains

Separation Principle
Markov Decision Process (S, A, H, T, R)

Given

- S: set of states
- A: set of actions
- H: horizon over which the agent will act
- T: S x A x S x {0, 1, ..., H} → [0,1], T_t(s,a,s') = P(s_{t+1} = s' | s_t = s, a_t = a)
- R: S x A x S x {0, 1, ..., H} → R, R_t(s,a,s') = reward for (s_{t+1} = s', s_t = s, a_t = a)

Goal:

- Find \( \pi: S \times \{0, 1, ..., H\} \rightarrow A \) that maximizes expected sum of rewards, i.e.,

\[
\pi^* = \arg \max_{\pi} \mathbb{E}[\sum_{t=0}^{H} R_t(S_t, A_t, S_{t+1}) | \pi]
\]
= MDP,  BUT

don’t get to observe the state itself, instead get sensory measurements

Now: what action to take given current probability distribution rather than given current state.
POMDPs: Tiger Example

S0
“tiger-left”
Pr(o=TL | S0, listen)=0.85
Pr(o=TR | S1, listen)=0.15

S1
“tiger-right”
Pr(o=TL | S0, listen)=0.15
Pr(o=TR | S1, listen)=0.85

Actions={
0: listen,
1: open-left,
2: open-right
}

Reward Function
- Penalty for wrong opening: -100
- Reward for correct opening: +10
- Cost for listening action: -1

Observations
- to hear the tiger on the left (TL)
- to hear the tiger on the right (TR)
Belief State

- Probability of S0 vs S1 being true underlying state
- Initial belief state: $p(S0) = p(S1) = 0.5$
- Upon listening, the belief state should change according to the Bayesian update (filtering)
Policy – Tiger Example

- Policy $\pi$ is a map from $[0,1] \rightarrow \{\text{listen, open-left, open-right}\}$
- What should the policy be?
  - Roughly: listen until sure, then open
- But where are the cutoffs?

```
H=1

\begin{array}{ccc}
\text{left} & \text{listen} & \text{right} \\
[0.00, 0.10] & [0.10, 0.90] & [0.90, 1.00]
\end{array}
```

```
H=2

\begin{array}{cccc}
\text{listen} & \text{listen} & \text{listen} & \text{listen} \\
\text{TL/TR} & \text{TR} & \text{TL/TR} & \text{TR} \\
[0.00, 0.02] & [0.02, 0.39] & [0.39, 0.61] & [0.61, 0.98] & [0.98, 1.00]
\end{array}
```

Solving POMDPs

- Canonical solution method 1: Continuous state “belief MDP”
  - Run value iteration, but now the state space is the space of probability distributions
  - \( \rightarrow \) value and optimal action for every possible probability distribution
  - \( \rightarrow \) will automatically trade off information gathering actions versus actions that affect the underlying state

- Value iteration updates cannot be carried out because uncountable number of belief states
  \( \rightarrow \) need approximate methods
Each belief node has $|A|$ action node successors

Each action node has $|O|$ belief successors

Each (action, observation) pair $(a, o)$ requires predict/update step

Matrix/vector formulation:
- $b(s)$: vector $b$ of length $|S|
- p(s'|s, a)$: set of $|S| \times |S|$ matrices $T_a$
- $p(o|s)$: vector $o$ of length $|S|

$b_a = T_a b$ (predict)

$p(o|a, b) = o^\top b_a$ (probability of observation)

$b_{a,o} = \text{diag}(o) b_a / (o^\top b_a)$ (update)
Solving POMDPs

- Canonical solution method 2:
  - Search over sequences of actions with limited look-ahead
  - Branching over actions and observations

Finite horizon:

$$|\mathcal{A}| \frac{|\mathcal{O}|^{H-1}}{|\mathcal{O}|-1}$$ nodes
Solving POMDPs

- Approximate solution: becoming tractable for $|S|$ in millions
  - $\alpha$-vector point-based techniques (belief state)
  - Monte Carlo Tree Search (search over action/observation sequences from current state)
  - ...beyond scope of this course...
Solving POMDPs

- Canonical solution method 3:
  - Plan in the *MDP*
  - Probabilistic inference (filtering) to track probability distribution
  - Choose optimal action for *MDP* for currently most likely state

Note: this is computationally efficient, but fails to explicitly seek out information
Outline

- Introduction to POMDPs
- Locally Optimal Solutions for POMDPs
  - Trajectory Optimization in (Gaussian) Belief Space
  - Accounting for Discontinuities in Sensing Domains
- Separation Principle
Motivation: Cost-effective, less precise robots

- Baxter (Rethink)
- Low-cost arm (Quigley et al.)
- Raven surgical robot (Rosen et al.)
- Blue (Gealy, McKinley et al, 2019)

Cable-driven 7-DOF arms
Motors connected to joints using cables
Perception (stereo, depth)
Model Uncertainty As Gaussians

Uncertainty parameterized by mean and covariance
Accounting for Uncertainty

Problem setting

State space plan

Belief space plan

How to find this plan?

[Example from Platt, Tedrake, Kaelbling, Lozano-Perez, 2010]
(Gaussian) Belief Space Planning

• Redefine problem

"x_t" = (μ_t, Σ_t)

• Convert underlying dynamics to belief space dynamics
  – Bayesian filter (e.g., extended Kalman filter)
Belief Space Planning

- **State-space planning through optimization**

  \[
  \min_{u,x} \sum_{t=0}^{H} c(x_t, u_t) \\
  \text{s.t. } x_{t+1} = f_{\text{dynamics}}(x_t, u_t, w_t)
  \]

  Deterministic approximation

  \[
  \min_{u,x} \sum_{t=0}^{H} c(x_t, u_t) \\
  \text{s.t. } x_{t+1} = f_{\text{dynamics}}(x_t, u_t, 0)
  \]

- **Gaussian belief space planning**

  Deterministic approximation (= ML assumption)

  \[
  \min_{u,\mu,\Sigma} \sum_{t=0}^{H} c(\mu_t, \Sigma_t, u_t) \\
  \text{s.t. } (\mu_{t+1}, \Sigma_{t+1}) = \text{EKF}(\mu_t, \Sigma_t, u_t, z_{t+1})
  \]

  \[
  \min_{u,\mu,\Sigma} \sum_{t=0}^{H} c(\mu_t, \Sigma_t, u_t) \\
  \text{s.t. } (\mu_{t+1}, \Sigma_{t+1}) = \text{EKF}(\mu_t, \Sigma_t, u_t, h(f(\mu_t, u_t)))
  \]

  Solved with Sequential Convex Programming

  [Platt, Tedrake, Kaelbling, Lozano-Perez, 2010; also Roy et al; van den Berg et al.]
Dealing with Uncertainty

Problem setting

State space plan

Belief space plan
# Gaussian Belief Space Planning

<table>
<thead>
<tr>
<th>Shooting</th>
<th>Partial Collocation</th>
<th>Full Collocation</th>
</tr>
</thead>
</table>
| \[
\begin{align*}
    & \min_{\mathbf{u}_{0:T-1}} \mathcal{C}(\hat{\mathbf{x}}_0, \Sigma_0, \mathbf{u}_{0:T-1}) \\
    \text{s.t.} & \quad \tilde{\mathbf{f}}(\hat{\mathbf{x}}_0, \mathbf{u}_{0:T-1}, 0) = \hat{\mathbf{x}}_{\text{target}} \\
    & \quad \tilde{\mathbf{f}}(\hat{\mathbf{x}}_0, \mathbf{u}_{0:t-1}, 0) \in \mathcal{X}_{\text{feasible}} \\
    & \quad \mathbf{u}_t \in \mathcal{U}_{\text{feasible}}
\end{align*}
\] | \[
\begin{align*}
    & \min_{\mathbf{u}_{0:T-1}} \mathcal{C}(\hat{\mathbf{x}}_{0:T}, \Sigma_0, \mathbf{u}_{0:T-1}) \\
    \text{s.t.} & \quad \hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, 0) \\
    & \quad \hat{\mathbf{x}}_T = \hat{\mathbf{x}}_{\text{target}}, \\
    & \quad \hat{\mathbf{x}}_t \in \mathcal{X}_{\text{feasible}}, \\
    & \quad \mathbf{u}_t \in \mathcal{U}_{\text{feasible}}
\end{align*}
\] | \[
\begin{align*}
    & \min_{\mathbf{u}_{0:T-1}} \mathcal{C}(\hat{\mathbf{x}}_{0:T}, \Sigma_0, \mathbf{u}_{0:T-1}) \\
    \text{s.t.} & \quad \hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, 0), \\
    & \quad \Sigma_{t+1} = (I - K_t H_t) \Sigma_t^{-}, \\
    & \quad \hat{\mathbf{x}}_T = \hat{\mathbf{x}}_{\text{target}}, \\
    & \quad \hat{\mathbf{x}}_t \in \mathcal{X}_{\text{feasible}}, \\
    & \quad \mathbf{u}_t \in \mathcal{U}_{\text{feasible}}
\end{align*}
\]
## Gaussian Belief Space Planning

<table>
<thead>
<tr>
<th>Shooting</th>
<th>Partial Collocation</th>
<th>Full Collocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\min_{\mathbf{u}_{0:T-1}} C(\hat{\mathbf{x}}<em>0, \Sigma_0, \mathbf{u}</em>{0:T-1})$</td>
<td>$\min_{\mathbf{u}<em>{0:T-1}} C(\hat{\mathbf{x}}</em>{0:T}, \Sigma_0, \mathbf{u}_{0:T-1})$</td>
<td>$\min_{\mathbf{u}<em>{0:T-1}} C(\hat{\mathbf{x}}</em>{0:T}, \Sigma_0:0, \mathbf{u}_{0:T-1})$</td>
</tr>
<tr>
<td>s.t. $\tilde{f}(\hat{\mathbf{x}}<em>0, \mathbf{u}</em>{0:T-1}, 0) = \hat{\mathbf{x}}_{\text{target}}$</td>
<td>s.t. $\hat{\mathbf{x}}_{t+1} = f(\hat{\mathbf{x}}_t, \mathbf{u}_t, 0)$</td>
<td>s.t. $\hat{\mathbf{x}}_{t+1} = f(\hat{\mathbf{x}}<em>t, \mathbf{u}<em>t, 0)$, $\Sigma</em>{t+1} = (I - K_t H_t) \Sigma</em>{t+1}$, $\hat{\mathbf{x}}<em>T = \hat{\mathbf{x}}</em>{\text{target}}$, $\hat{\mathbf{x}}<em>t \in \mathcal{X}</em>{\text{feasible}}$, $\mathbf{u}<em>t \in \mathcal{U}</em>{\text{feasible}}$</td>
</tr>
<tr>
<td>$\tilde{f}(\hat{\mathbf{x}}<em>0, \mathbf{u}</em>{0:T-1}, 0) \in \mathcal{X}_{\text{feasible}}$</td>
<td>$\hat{\mathbf{x}}<em>T = \hat{\mathbf{x}}</em>{\text{target}}$, $\hat{\mathbf{x}}<em>t \in \mathcal{X}</em>{\text{feasible}}$, $\mathbf{u}<em>t \in \mathcal{U}</em>{\text{feasible}}$</td>
<td></td>
</tr>
<tr>
<td>$\mathbf{u}<em>t \in \mathcal{U}</em>{\text{feasible}}$</td>
<td></td>
<td>$\mathbf{u}<em>t \in \mathcal{U}</em>{\text{feasible}}$</td>
</tr>
</tbody>
</table>

**+** Much better scalability
**+** No infeasibility issues
**-** Poorly conditioned / small stepsizes / slow
**-** Can’t constrain mu and Sigma

**+** Can constrain states
**+** Bends itself into a solution
**-** Poor scalability
**-** Infeasible local optima
### Gaussian Belief Space Planning

<table>
<thead>
<tr>
<th>Shooting</th>
<th>Partial Collocation</th>
<th>Full Collocation</th>
</tr>
</thead>
</table>
| \[
\min_{\hat{u}_{0:T-1}} C(\hat{x}_{0:T}, \Sigma_0, \hat{u}_{0:T-1})
\]
| s.t. | \[
\tilde{f}(\hat{x}_0, \hat{u}_{0:T-1}, 0) = \hat{x}_{\text{target}}
\]
| \[
\tilde{f}(\hat{x}_0, \hat{u}_{0:T-1}, 0) \in \mathcal{X}_{\text{feasible}}
\]
| \[
\hat{u}_t \in \mathcal{U}_{\text{feasible}}
\] |
| \[
\min_{\hat{x}_{0:T-1}} C(\hat{x}_{0:T}, \Sigma_0, \hat{u}_{0:T-1})
\]
| s.t. | \[
\hat{x}_{t+1} = f(\hat{x}_t, \hat{u}_t, 0)
\]
| \[
\hat{x}_T = \hat{x}_{\text{target}},
\]
| \[
\hat{x}_t \in \mathcal{X}_{\text{feasible}},
\]
| \[
\hat{u}_t \in \mathcal{U}_{\text{feasible}}
\] |
| \[
\min_{\hat{x}_{0:T-1}} C(\hat{x}_{0:T}, \Sigma_0, \hat{u}_{0:T-1})
\]
| s.t. | \[
\hat{x}_{t+1} = f(\hat{x}_t, \hat{u}_t, 0),
\]
| \[
\Sigma_{t+1} = (I - K_t H_t) \Sigma_{t+1},
\]
| \[
\hat{x}_T = \hat{x}_{\text{target}},
\]
| \[
\hat{x}_t \in \mathcal{X}_{\text{feasible}},
\]
| \[
\hat{u}_t \in \mathcal{U}_{\text{feasible}}
\] |

+ Much better scalability
+ No infeasibility issues
- Poorly conditioned / small step sizes / slow
- Can’t constrain \( \mu \) and \( \Sigma \)
+ Can constrain states
+ Bends itself into a solution
- Poor scalability
- Infeasible local optima
Scalability

[Patil et al., WAFR 2014]
Active SLAM through Gaussian Belief Space Planning

[Patil et al., WAFR 2014]
Dealing with Discontinuities

Zero gradient, hence local optimum
Dealing with Discontinuities

Increasing difficulty

Noise level determined by signed distance to sensing region (computed with GJK/EPA) homotopy iteration

[Patil et al., ICRA 2014]
**Signed Distance to Sensing Discontinuity**

Field of view (FOV) discontinuity

- (a) $\text{sd}({\mathbf{b}}_t, \Pi^*) > 0$
  - Outside field of view

- (b) $\text{sd}({\mathbf{b}}_t, \Pi^*) < 0$
  - Inside field of view

Occlusion discontinuity

- (c) $\text{sd}({\mathbf{b}}_t, \Pi^*) > 0$
  - Occluded view

- (d) $\text{sd}({\mathbf{b}}_t, \Pi^*) < 0$
  - Unoccluded view
\( \delta_t^s \) vs. Signed distance

(a) \( \text{sd}(\hat{b}_t, \Pi^s) > 0 \)  
Outside field of view

(b) \( \text{sd}(\hat{b}_t, \Pi^s) < 0 \)  
Inside field of view

(c) \( \text{sd}(\hat{b}_t, \Pi^s) > 0 \)  
Occluded view

(d) \( \text{sd}(\hat{b}_t, \Pi^s) < 0 \)  
Unoccluded view

\[ \delta_t^s = \chi(\text{sd}(\hat{b}_t, \Pi^s)) \]
Modified Belief Dynamics

\[ \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t, \mathbf{q}_t), \quad \mathbf{q}_t \sim \mathcal{N}(\mathbf{0}, I), \]

\[ \mathbf{z}_t = h(\mathbf{x}_t, \mathbf{r}_t), \quad \mathbf{r}_t \sim \mathcal{N}(\mathbf{0}, I), \]

\[ \hat{\mathbf{b}}_{t+1} = g(\hat{\mathbf{b}}_t, \hat{\mathbf{u}}_t) = \left[ \begin{array}{c} \hat{\mathbf{x}}_{t+1} \\ \text{vec} \left[ \sqrt{\Sigma_{t+1}^+ - K_t H_t \Sigma_{t+1}^-} \right] \end{array} \right] \]

\[ \hat{\mathbf{x}}_{t+1} = f(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t, \mathbf{0}), \quad \Sigma_{t+1}^- = A_t \sqrt{\Sigma_t} (A_t \sqrt{\Sigma_t})^T + Q_t Q_t^T, \]

\[ A_t = \frac{\partial f}{\partial \mathbf{x}} (\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t, \mathbf{0}), \quad Q_t = \frac{\partial f}{\partial \mathbf{q}} (\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t, \mathbf{0}), \]

\[ H_t = \frac{\partial h}{\partial \mathbf{x}} (\hat{\mathbf{x}}_{t+1}, \mathbf{0}), \quad R_t = \frac{\partial h}{\partial \mathbf{r}} (\hat{\mathbf{x}}_{t+1}, \mathbf{0}), \]

\[ K_t = \Sigma_{t+1}^- H_t^T \Delta_{t+1} (\Delta_{t+1} H_t \Sigma_{t+1}^- H_t^T \Delta_{t+1} + R_t R_t^T)^{-1} \Delta_{t+1}. \]

\[ \delta_t^S : \text{Binary variable \{0,1\}} \]

\[ 0 \rightarrow \text{No measurement} \]

\[ 1 \rightarrow \text{Measurement} \]
Incorporating $\delta_t^s$ in SQP

- Binary non-convex program – difficult to solve
- Solve successively smooth approximations

$$\delta_t^s(\alpha) = \tilde{\chi}(sd(\hat{b}_t, \Pi^s), \alpha)$$

$$= 1 - \frac{1}{1 + \exp(-\alpha \cdot sd(\hat{b}_t, \Pi^s))}$$
Algorithm Overview

- While $\delta$ not within desired tolerance
  - Solve optimization problem with current value of $\alpha$
  - Increase $\alpha$
  - Re-integrate belief trajectory
  - Update $\delta$
Algorithm

Inputs:
\( \tilde{B}_{0:\ell} = [\tilde{b}_0 \ldots \tilde{b}_\ell], \tilde{U}_{0:\ell-1} = [\tilde{u}_0 \ldots \tilde{u}_{\ell-1}] \): Belief space trajectory initialization
\( \ell \): Number of time intervals
Cost and constraint definitions (Eq. (4))

Parameters:
\( \alpha \): Approximation parameter for relaxing discrete sensing constraints
\( k \): Coefficient to control rate of increase of \( \alpha \)
\( \tau \): Execution time step \((0 \leq \tau \leq \ell)\)
\( \epsilon \): Convergence tolerance parameter

Variables:
\( \tilde{B}_{0:\ell} = [\tilde{b}_0 \ldots \tilde{b}_\ell], \tilde{U}_{0:\ell-1} = [\tilde{u}_0 \ldots \tilde{u}_{\ell-1}] \): Optimization variables
\( \delta_{0:\ell} \): Binary vector to track value of continuous approximation for convergence criterion

1: \textbf{for } \tau = 0, \ldots, \ell - 1 \textbf{ do} \quad \text{\texttt{\# Re-planning loop following the MPC paradigm}}
2: \quad \alpha \leftarrow a_{\text{init}}
3: \quad \textbf{while } \delta_{\tau,\ell} \text{ not within } \epsilon \text{ tolerance of true binary values } \{0, 1\} \textbf{ do}
4: \quad \quad \text{Reset trust region size and penalty coefficient} \quad \text{\texttt{\# [25]}}
5: \quad \quad [\tilde{B}_{\tau,\ell}, \tilde{U}_{\tau,\ell-1}] \leftarrow \text{SQP-based optimization of approximation given } [\tilde{B}_{\tau,\ell}, \tilde{U}_{\tau,\ell-1}] \quad \text{\texttt{\# [25]}}
6: \quad \quad \alpha \leftarrow k * \alpha \quad \text{\texttt{\# }\alpha\texttt{-update to increase noise outside sensing region}}
7: \quad \quad \tilde{b}_{\tau+1} = g(\tilde{b}_\tau, \tilde{u}_\tau) \quad \forall \tau, \ldots, \ell - 1 \quad \text{\texttt{\# Integrate belief trajectory after }\alpha\texttt{-update}}
8: \quad \quad \text{Update } \delta_{\tau,\ell} \leftarrow \delta_{\tau,\ell}(\alpha) \quad \text{\texttt{\# Eq. (5)}}
9: \quad \quad [\tilde{B}_{\tau,\ell}, \tilde{U}_{\tau,\ell-1}] \leftarrow [\tilde{B}_{\tau,\ell}, \tilde{U}_{\tau,\ell-1}] \quad \text{\texttt{\# Update trajectory initialization}}
10: \quad \textbf{end while}
11: \quad \text{Execute } \tilde{u}_\tau \quad \text{\texttt{\# Eq. (6a)}}
12: \quad \text{Obtain measurement and update } \tilde{b}_{\tau+1} \text{ using EKF}
13: \quad \text{Truncate } \tilde{b}_{\tau+1} \text{ w.r.t sensing region boundary}
14: \quad \text{Update sensing regions for all sensors}
15: \quad \tilde{b}_{\tau+1} = g(\tilde{b}_\tau, \tilde{u}_\tau) \quad \forall \tau, 1, \ldots, \ell - 1 \quad \text{\texttt{\# Integrate belief trajectory after Kalman update}}
16: \quad \quad \text{using previously optimized control inputs } \tilde{U}_{\tau+1,\ell-1}
17: \textbf{end for}
Discontinuities in Sensing Domains

Increasing difficulty

噪声级别由到传感区域的符号距离确定

* homotopy iteration
However...
“No measurement” Belief Update

Sensing region boundary

Truncate Gaussian Belief if no measurement obtained

Truncated belief state
Effect of Truncation

Without “No measurement” Belief Update

With “No measurement” Belief Update
Experiments
Grasping: Planar 3-link Manipulator

- 6D state space: Arm joint angles + camera orientation + object position
  - 27D belief space

- **Objective**: Reliably grasp object
Robot Arm Occluding Object from Camera View

Initial belief

State space plan execution

(way-point)
Belief space plan execution

(end)
Same Scenario but Movable Camera

Initial belief  State space plan execution  (way-point) Belief space plan execution  (end)
Car and Landmarks (Active Exploration)
Collision Avoidance

- So far approximating robot geometry as points or spheres

- Articulated robots cannot be approximated as points/spheres
  - Gaussian noise in joint space
  - Need probabilistic collision avoidance w.r.t robot links
**Definition:** Convex hull of a robot link transformed (in joint space) according to sigma points

Consider sigma points lying on the standard deviation contour of uncertainty covariance (UKF)
Collision Avoidance Constraint

Consider signed distance between obstacle and sigma hulls
Continuous Collision Avoidance Constraint

- Discrete collision avoidance can lead to trajectories that collide with obstacles in between time steps

- Use convex hull of sigma hulls between consecutive time steps

  \[ sd(\text{convhull}(A_t, A_{t+1}), O) \geq d_{\text{safe}} \quad \forall \ O \in \mathcal{O} \]

- Advantages:
  - Solutions are collision-free in between time-steps
  - Discretized trajectory can have less time-steps
During execution, update the belief state based on the actual observation

Re-plan after every belief state update

Effective feedback control, provided one can re-plan sufficiently fast

Belief Space Model Predictive Control
Example: 4-DOF planar robot

State space trajectory

Initial state $x_0$

Intermediate states $x_1...T-1$

Final state $x_T$

Obstacles

Target position

$x_{sensing}$
Example: 4-DOF planar robot

1-standard deviation belief space trajectory

- Initial mean state $\hat{x}_0$
- Final mean state $\hat{x}_T$
- Narrow clearance from obstacles between consecutive time steps (sigma hull for last time step)
Example: 4-DOF planar robot

4-standard deviation belief space trajectory

$x_{sensing}$  Initial mean state $\dot{x}_0$

Final mean state $\dot{x}_T$

Wider clearance from obstacles between consecutive time steps (sigma hull at last time step)
Example: 4-DOF planar robot
Example: 4-DOF planar robot

Mean distance from target

- Belief space plan (Open-loop execution)
- Belief space plan (Feedback policy)
- Belief space re-planning
Outline

- Introduction to POMDPs
- Locally Optimal Solutions for POMDPs
  - Trajectory Optimization in (Gaussian) Belief Space
  - Accounting for Discontinuities in Sensing Domains
- Separation Principle
Separation Principle

- Assume:
  \[ x_{t+1} = Ax_t + Bu_t + w_t \quad w_t \sim \mathcal{N}(0, Q_t) \]
  \[ z_t = Cx_t + v_t \quad v_t \sim \mathcal{N}(0, R_t) \]

- Goal:
  \[
  \text{minimize } \mathbb{E} \left[ \sum_{t=0}^{H} u_t^\top U u_t + x_t^\top X_t x_t \right]
  \]

- Then, optimal control policy consists of:

  1. Offline/Ahead of time: Run LQR to find optimal control policy for fully observed case, which gives sequence of feedback matrices
     \[ K_1, K_2, \ldots \]
  2. Online: Run Kalman filter to estimate state, and apply control
     \[ u_t = K_t \mu_{t|0:t} \]
Recap

- POMDP = MDP but sensory measurements instead of exact state knowledge
- Locally optimal solutions in Gaussian belief spaces
  - Augmented state vector (mean, covariance)
  - Trajectory optimization
- Homotopy methods for dealing with discontinuities in sensing domains
- Sigma Hulls for probabilistic collision avoidance
- Separation principle