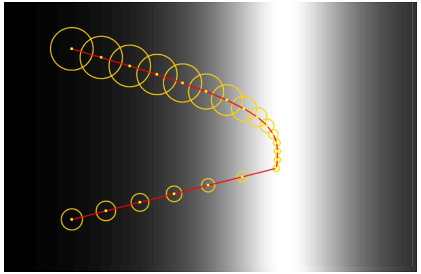


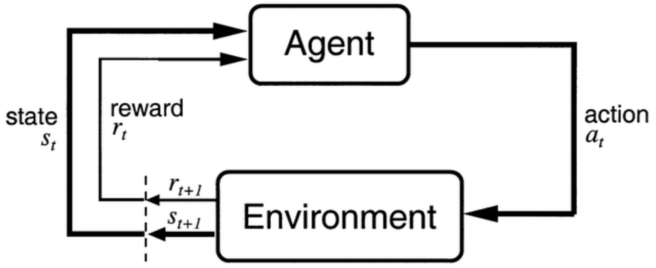
CS 287 Advanced Robotics (Fall 2019)
Lecture 15
Partially Observable Markov Decision Processes (POMDPs)

Pieter Abbeel

Outline

- 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] , $T_t(s,a,s') = P(s_{t+1} = s' \mid s_t = s, a_t = a)$
- R: S x A x S x {0, 1, …, H} → $\mathbb{R}$, $R_t(s,a,s')$ = reward for $(s_{t+1} = s', s_t = s, a_t = a)$

Goal:

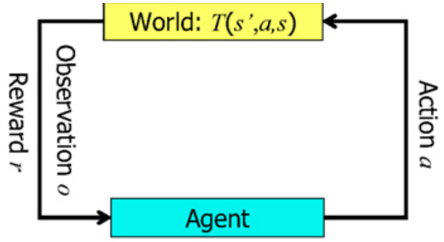
- Find π: S x {0, 1, …, H} → A that maximizes expected sum of rewards, i.e.,

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

POMDP – Partially Observable MDP

= MDP, BUT

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

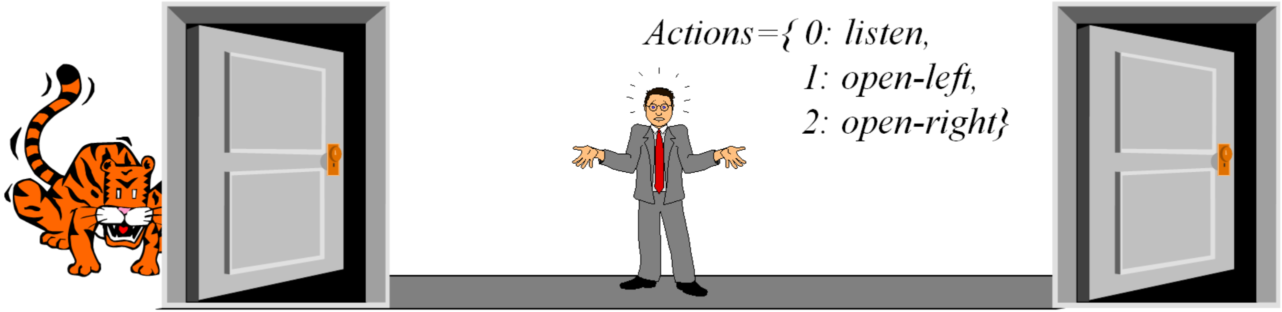


Now: what action to take given current probability distribution rather than given current state.

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



Reward Function

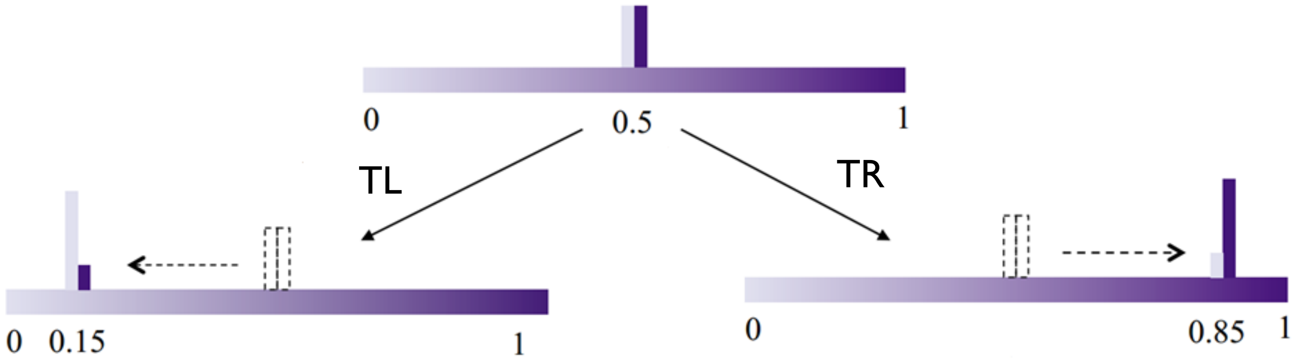
- *Penalty for wrong opening: -100*
- *Reward for correct opening: +10*
- *Cost for listening action: -1*

Observations

- *to hear the tiger on the left (TL)*
- *to hear the tiger on the right(TR)*

Belief State

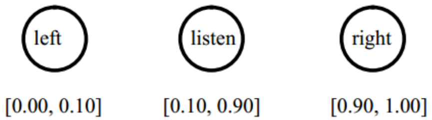
- Probability of S0 vs S1 being true underlying state
- Initial belief state: p(S0)=p(S1)=0.5
- Upon listening, the belief state should change according to the Bayesian update (filtering)



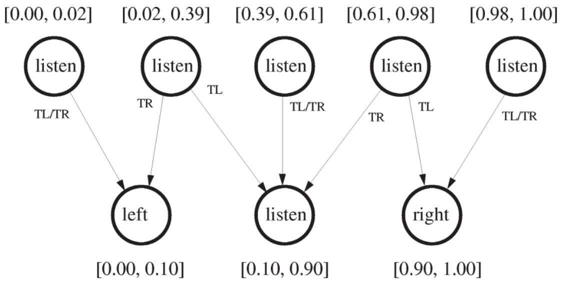
Policy – Tiger Example

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

H=1



H=2



Solving POMDPs

- Canonical solution method 1: Continuous state "belief MDP"
  - Run value iteration, but now the state space is the space of probability distributions
  - → value and optimal action for every possible probability distribution
  - → will <u>automatically trade off information gathering actions versus actions</u> that affect the underlying state
  - Value iteration updates cannot be carried out because uncountable number of belief states
    –> need approximate methods

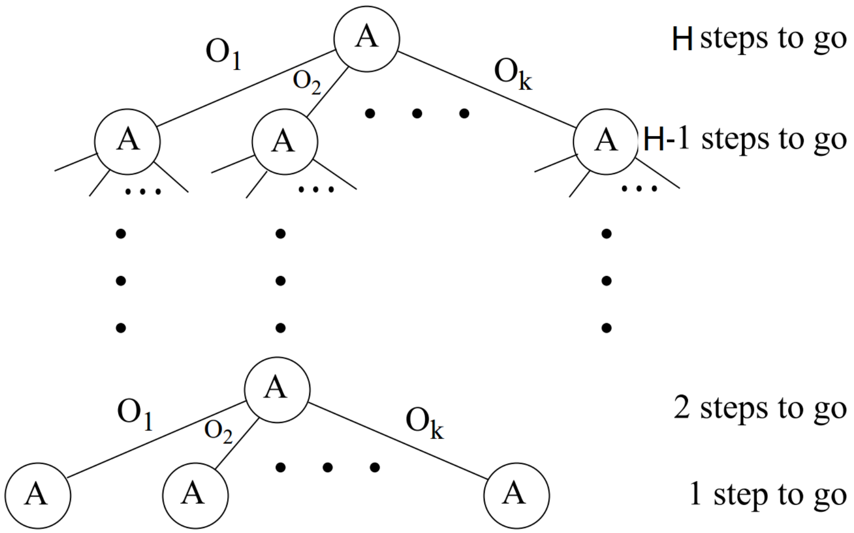
Belief State Update

- 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|x|S| matrices T_a
  - p(o|s): vector **o** of length |S|
- $\mathbf{b}_a = T_a\mathbf{b}$ (predict)
- $p(o|a,b) = \mathbf{o}^T \mathbf{b}_a$ (probability of observation)
- $\mathbf{b}_{a,o} = \mathrm{diag}(\mathbf{o})\, \mathbf{b}_a / (\mathbf{o}^T \mathbf{b}_a)$ (update)

$$\equiv b_{a,o}(s') = \frac{p(o \mid s', a) \sum_{s_i \in S} p(s' \mid s_i, a) b(s_i)}{p(o \mid a, b)}$$

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
  - α-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

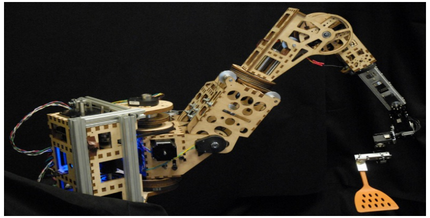
Note: this is computationally efficient, but fails to explicitly seek out information

Outline

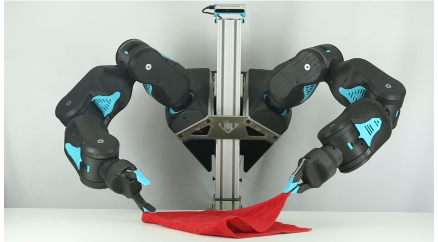
- Introduction to POMDPs
- Locally Optimal Solutions for POMDPs
  - Trajectory Optimization in (Gaussian) Belief Space
  - Accounting for Discontinuities in Sensing Domains
- Separation Principle

Motivation: Cost-effective, less precise robots

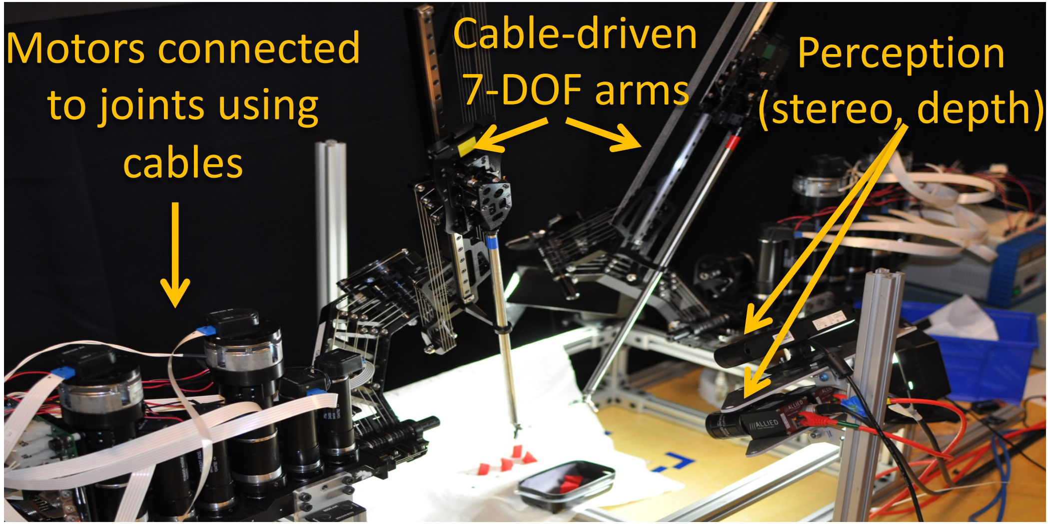
Baxter (Rethink)



Low-cost arm (Quigley et al.)

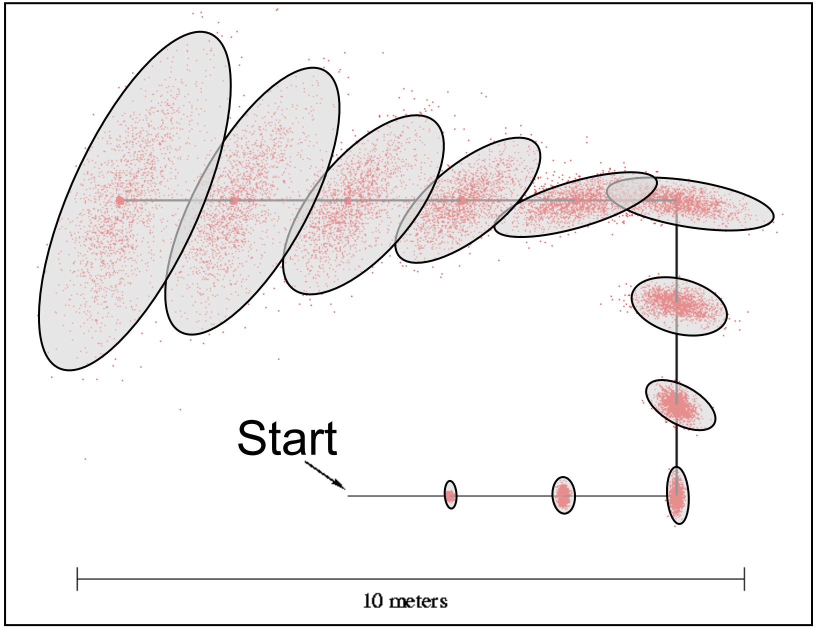


Blue (Gealy, McKinley et al, 2019)



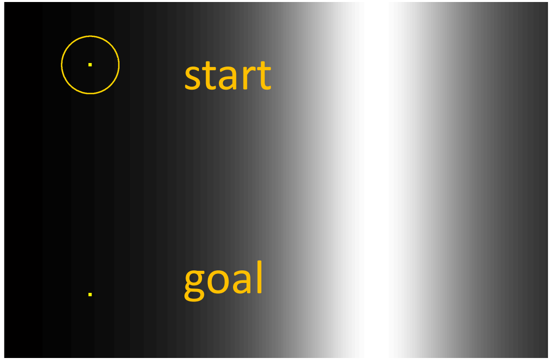
Raven surgical robot (Rosen et al.)

Model Uncertainty As Gaussians

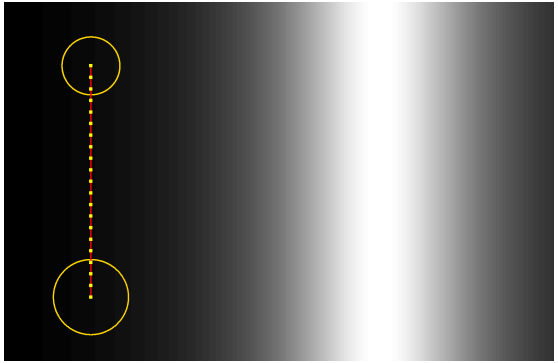


Uncertainty parameterized by
mean and covariance

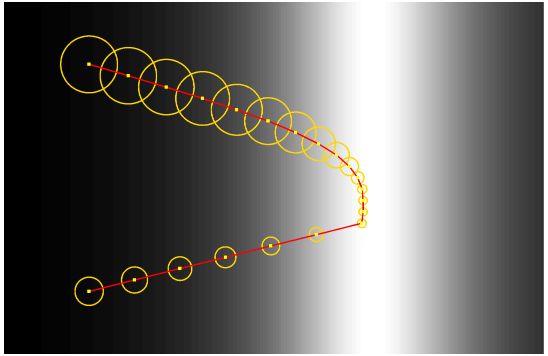
Accounting for Uncertainty



Problem setting



State space plan



Belief space plan

How to find this plan?

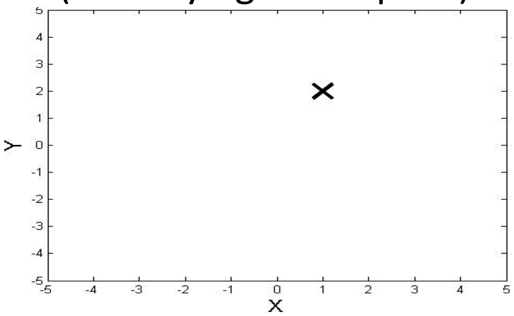
[Example from Platt, Tedrake, Kaelbling, Lozano-Perez, 2010]

(Gaussian) Belief Space Planning

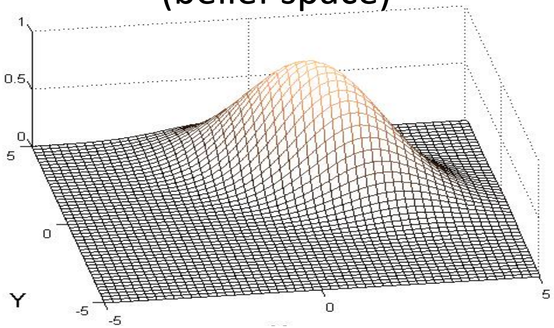
- Redefine problem

$$"x_t" = (\mu_t, \Sigma_t)$$

(underlying state space)



(belief space)



- Convert underlying dynamics to belief space dynamics
  - Bayesian filter (e.g., extended Kalman filter)

Belief Space Planning

- State-space planning through optimization

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

Deterministic approximation

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

- Gaussian belief space planning

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

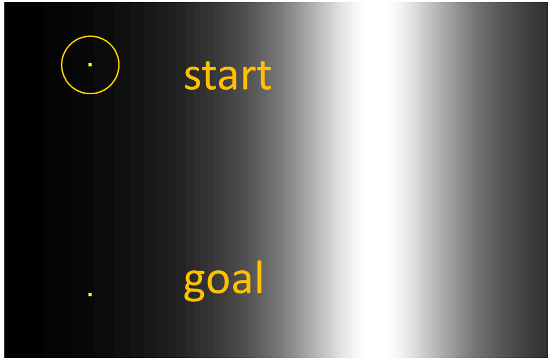
Deterministic approximation (= ML assumption)

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

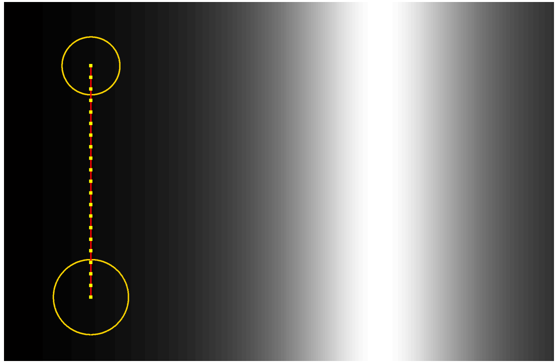
Solved with Sequential Convex Programming

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

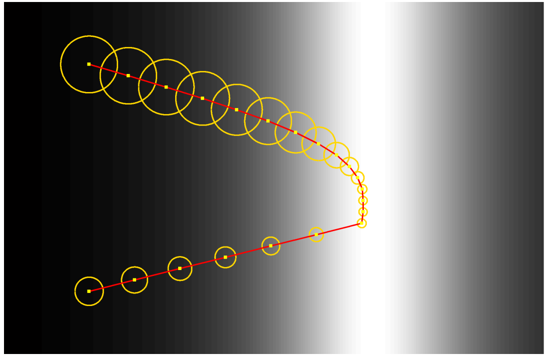
Dealing with Uncertainty



Problem setting



State space plan



Belief space plan

Gaussian Belief Space Planning

Shooting	Partial Collocation	Full Collocation
$\min_{\mathbf{u}_{0:T-1}} \quad \mathcal{C}(\hat{\mathbf{x}}_0, \Sigma_0, \mathbf{u}_{0:T-1})$ s.t $\quad \tilde{\mathbf{f}}(\hat{\mathbf{x}}_0, \mathbf{u}_{0:T-1}, \mathbf{0}) = \hat{\mathbf{x}}_{\text{target}}$ $\tilde{\mathbf{f}}(\hat{\mathbf{x}}_0, \mathbf{u}_{0:t-1}, \mathbf{0}) \in \mathcal{X}_{\text{feasible}}$ $\mathbf{u}_t \in \mathcal{U}_{\text{feasible}}$	$\min_{\substack{\mathbf{u}_{0:T-1} \\ \hat{\mathbf{x}}_{0:T}}} \quad \mathcal{C}(\hat{\mathbf{x}}_{0:T}, \Sigma_0, \mathbf{u}_{0:T-1})$ s.t $\quad \hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0})$ $\hat{\mathbf{x}}_T = \hat{\mathbf{x}}_{\text{target}},$ $\hat{\mathbf{x}}_t \in \mathcal{X}_{\text{feasible}},$ $\mathbf{u}_t \in \mathcal{U}_{\text{feasible}}$	$\min_{\substack{\mathbf{u}_{0:T-1} \\ \hat{\mathbf{x}}_{0:T} \\ \Sigma_{0:T}}} \quad \mathcal{C}(\hat{\mathbf{x}}_{0:T}, \Sigma_{0:T}, \mathbf{u}_{0:T-1})$ s.t $\quad \hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}),$ $\Sigma_{t+1} = (I - K_t H_t)\Sigma_{t+1}^{-},$ $\hat{\mathbf{x}}_T = \hat{\mathbf{x}}_{\text{target}},$ $\hat{\mathbf{x}}_t \in \mathcal{X}_{\text{feasible}},$ $\mathbf{u}_t \in \mathcal{U}_{\text{feasible}}$

Gaussian Belief Space Planning

Shooting	Partial Collocation	Full Collocation
$\min_{\mathbf{u}_{0:T-1}} \; \mathcal{C}(\hat{\mathbf{x}}_0, \Sigma_0, \mathbf{u}_{0:T-1})$ s.t $\tilde{\mathbf{f}}(\hat{\mathbf{x}}_0, \mathbf{u}_{0:T-1}, \mathbf{0}) = \hat{\mathbf{x}}_{\text{target}}$ $\tilde{\mathbf{f}}(\hat{\mathbf{x}}_0, \mathbf{u}_{0:t-1}, \mathbf{0}) \in \mathcal{X}_{\text{feasible}}$ $\mathbf{u}_t \in \mathcal{U}_{\text{feasible}}$	$\min_{\substack{\mathbf{u}_{0:T-1} \\ \hat{\mathbf{x}}_{0:T}}} \; \mathcal{C}(\hat{\mathbf{x}}_{0:T}, \Sigma_0, \mathbf{u}_{0:T-1})$ s.t $\hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0})$ $\hat{\mathbf{x}}_T = \hat{\mathbf{x}}_{\text{target}}$, $\hat{\mathbf{x}}_t \in \mathcal{X}_{\text{feasible}}$, $\mathbf{u}_t \in \mathcal{U}_{\text{feasible}}$	$\min_{\substack{\mathbf{u}_{0:T-1} \\ \hat{\mathbf{x}}_{0:T} \\ \Sigma_{0:T}}} \; \mathcal{C}(\hat{\mathbf{x}}_{0:T}, \Sigma_{0:T}, \mathbf{u}_{0:T-1})$ s.t $\hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0})$, $\Sigma_{t+1} = (I - K_t H_t)\Sigma_{t+1}^-$, $\hat{\mathbf{x}}_T = \hat{\mathbf{x}}_{\text{target}}$, $\hat{\mathbf{x}}_t \in \mathcal{X}_{\text{feasible}}$, $\mathbf{u}_t \in \mathcal{U}_{\text{feasible}}$

+ 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

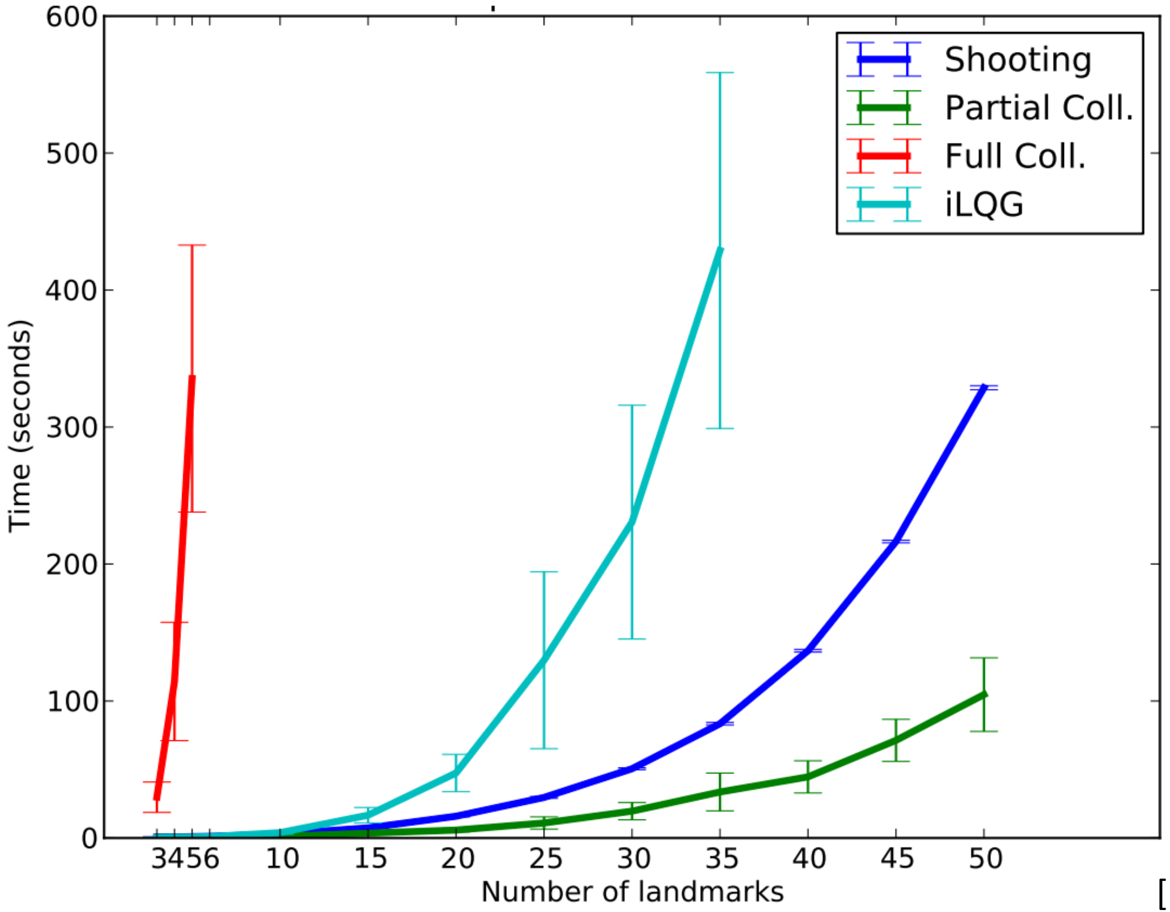
Gaussian Belief Space Planning

Shooting	Partial Collocation	Full Collocation
$\min_{\mathbf{u}_{0:T-1}} \; \mathcal{C}(\hat{\mathbf{x}}_0, \Sigma_0, \mathbf{u}_{0:T-1})$ s.t $\tilde{\mathbf{f}}(\hat{\mathbf{x}}_0, \mathbf{u}_{0:T-1}, \mathbf{0}) = \hat{\mathbf{x}}_{\text{target}}$ $\tilde{\mathbf{f}}(\hat{\mathbf{x}}_0, \mathbf{u}_{0:t-1}, \mathbf{0}) \in \mathcal{X}_{\text{feasible}}$ $\mathbf{u}_t \in \mathcal{U}_{\text{feasible}}$	$\min_{\mathbf{u}_{0:T-1}, \hat{\mathbf{x}}_{0:T}} \; \mathcal{C}(\hat{\mathbf{x}}_{0:T}, \Sigma_0, \mathbf{u}_{0:T-1})$ s.t $\hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0})$ $\hat{\mathbf{x}}_T = \hat{\mathbf{x}}_{\text{target}},$ $\hat{\mathbf{x}}_t \in \mathcal{X}_{\text{feasible}},$ $\mathbf{u}_t \in \mathcal{U}_{\text{feasible}}$	$\min_{\mathbf{u}_{0:T-1}, \hat{\mathbf{x}}_{0:T}, \Sigma_{0:T}} \; \mathcal{C}(\hat{\mathbf{x}}_{0:T}, \Sigma_{0:T}, \mathbf{u}_{0:T-1})$ s.t $\hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \mathbf{u}_t, \mathbf{0}),$ $\Sigma_{t+1} = (I - K_t H_t)\Sigma_{t+1}^-,$ $\hat{\mathbf{x}}_T = \hat{\mathbf{x}}_{\text{target}},$ $\hat{\mathbf{x}}_t \in \mathcal{X}_{\text{feasible}},$ $\mathbf{u}_t \in \mathcal{U}_{\text{feasible}}$

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

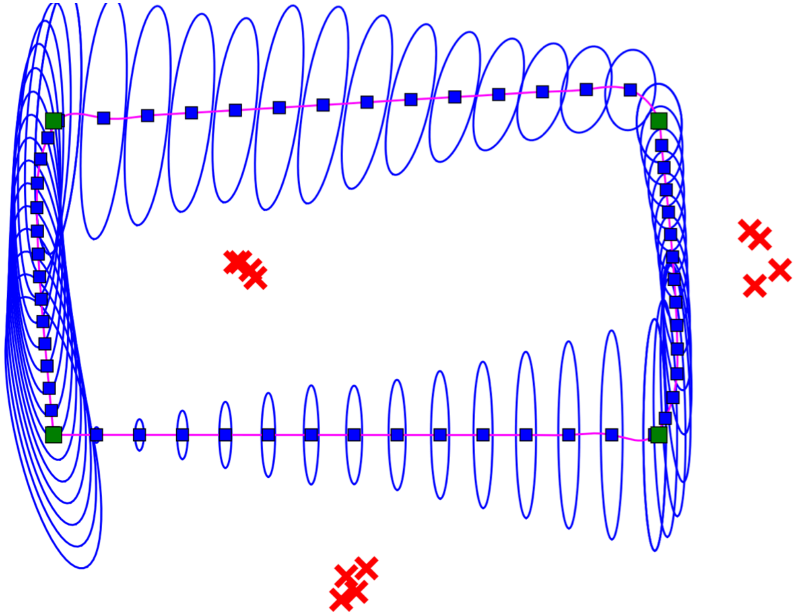
+ Can constrain states
+ Bends itself into a solution
- Poor scalability
- Infeasible local optima

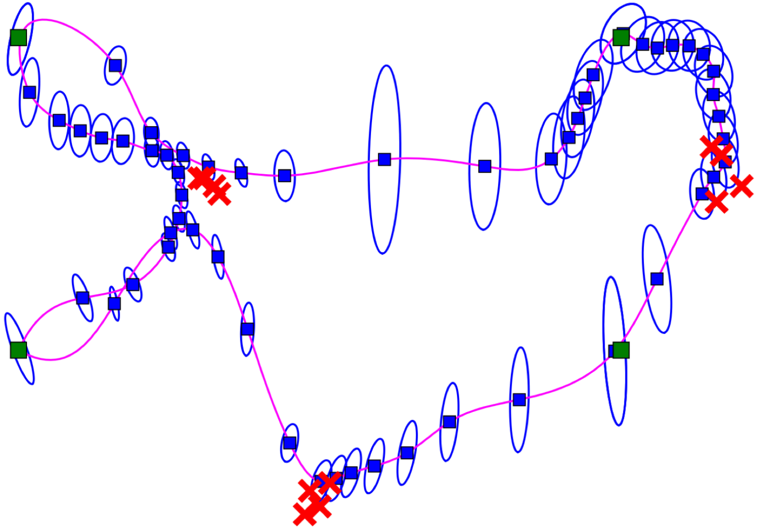
Scalability



[Patil et al., WAFR 2014]

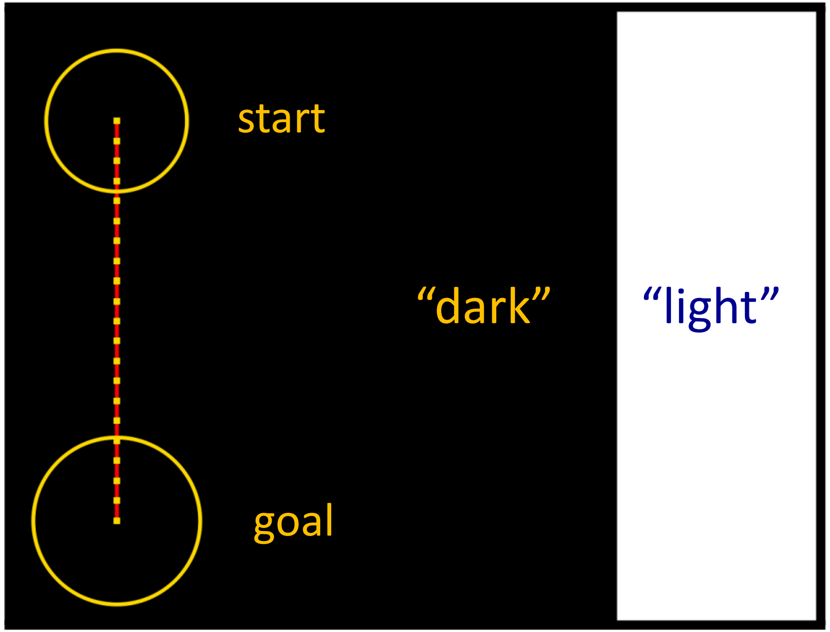
Active SLAM through Gaussian Belief Space Planning





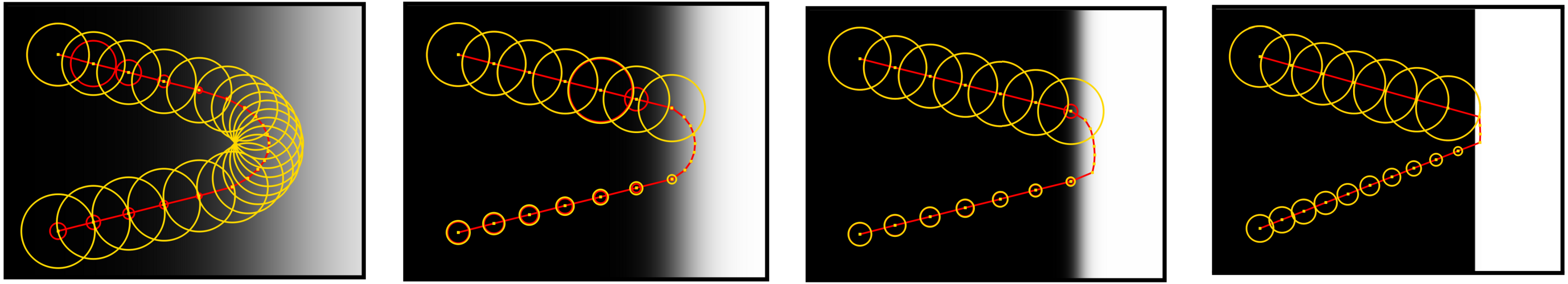
[Patil et al., WAFR 2014]

Dealing with Discontinuities



Zero gradient, hence local optimum

Dealing with Discontinuities



Increasing difficulty →

Noise level determined by

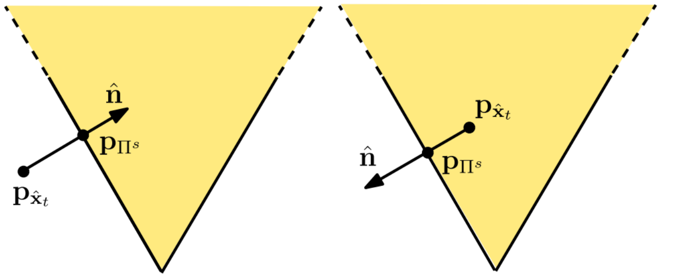
signed distance to sensing region (computed with GJK/EPA)

homotopy iteration

[Patil et al., ICRA 2014]

Signed Distance to Sensing Discontinuity

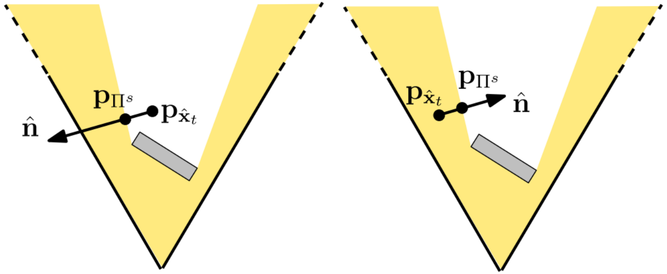
Field of view (FOV) discontinuity



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

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

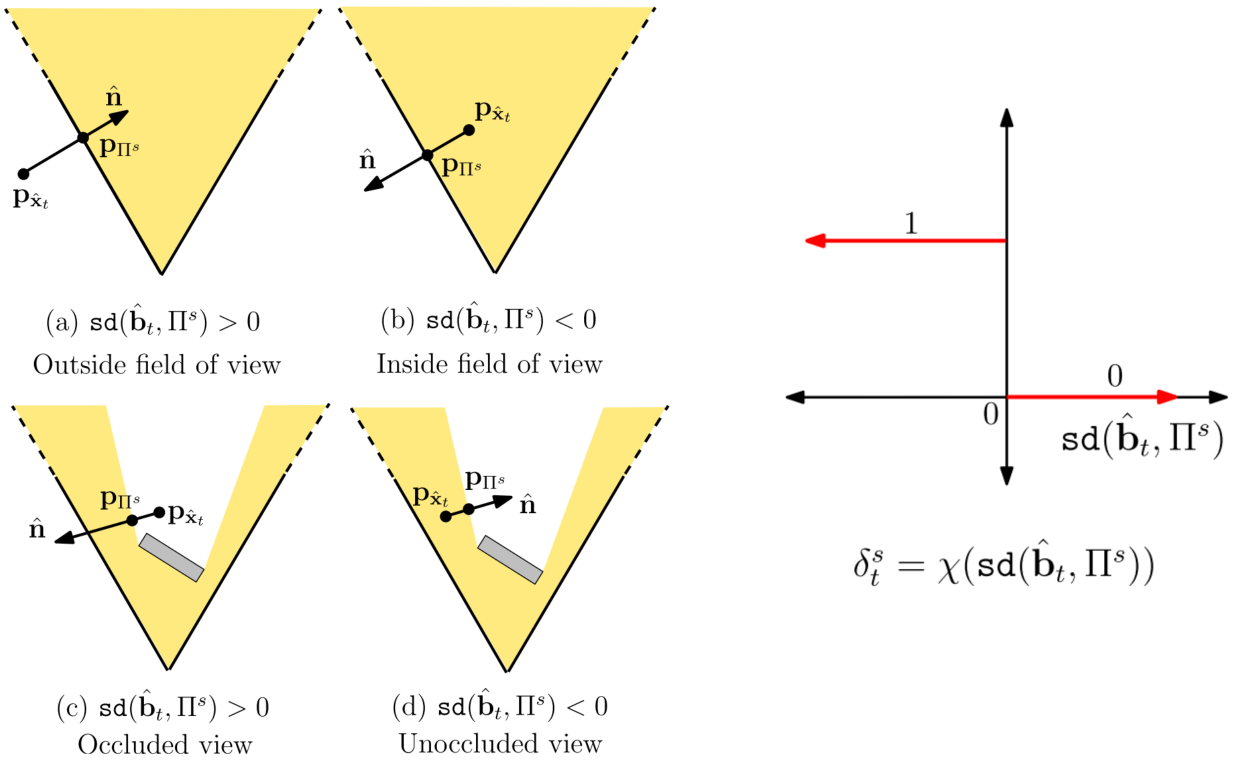
Occlusion discontinuity



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

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

δ_t^s vs. Signed distance



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

Modified Belief Dynamics

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t, \mathbf{q}_t), \quad \mathbf{q}_t \sim \mathcal{N}(\mathbf{0}, I),$$
$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t, \mathbf{r}_t), \qquad \mathbf{r}_t \sim \mathcal{N}(\mathbf{0}, I),$$

$$\hat{\mathbf{b}}_{t+1} = \mathbf{g}(\hat{\mathbf{b}}_t, \hat{\mathbf{u}}_t) = \begin{bmatrix} \hat{\mathbf{x}}_{t+1} \\ \mathrm{vec}[\sqrt{\Sigma^-_{t+1} - K_t H_t \Sigma^-_{t+1}}] \end{bmatrix}$$

$$\hat{\mathbf{x}}_{t+1} = \mathbf{f}(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t, \mathbf{0}), \qquad \Sigma^-_{t+1} = A_t \sqrt{\Sigma_t}(A_t \sqrt{\Sigma_t})^T + Q_t Q_t^T,$$
$$A_t = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t, \mathbf{0}), \qquad Q_t = \frac{\partial \mathbf{f}}{\partial \mathbf{q}}(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t, \mathbf{0}),$$
$$H_t = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}(\hat{\mathbf{x}}_{t+1}, \mathbf{0}), \qquad R_t = \frac{\partial \mathbf{h}}{\partial \mathbf{r}}(\hat{\mathbf{x}}_{t+1}, \mathbf{0}),$$
$$\boxed{K_t = \Sigma^-_{t+1} H_t^T \Delta_{t+1} (\Delta_{t+1} H_t \Sigma^-_{t+1} H_t^T \Delta_{t+1} + R_t R_t^T)^{-1} \Delta_{t+1}.}$$

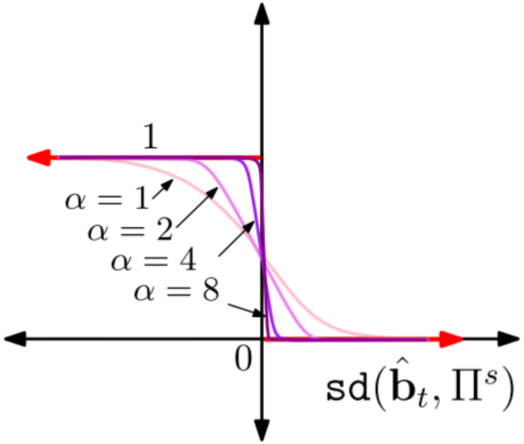
δ_t^s : Binary variable {0,1}

0 -> No measurement

1 -> Measurement

Incorporating δ_t^s in SQP

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



$$\begin{aligned}\delta_t^s(\alpha) &= \tilde{\chi}(\mathtt{sd}(\hat{\mathbf{b}}_t, \Pi^s), \alpha) \\ &= 1 - \frac{1}{1+\exp(-\alpha \cdot \mathtt{sd}(\hat{\mathbf{b}}_t, \Pi^s))}\end{aligned}$$

Algorithm Overview

- While δ not within desired tolerance
  - Solve optimization problem with current value of α
  - Increase α
  - Re-integrate belief trajectory
  - Update δ

Algorithm

Inputs:

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

Parameters:

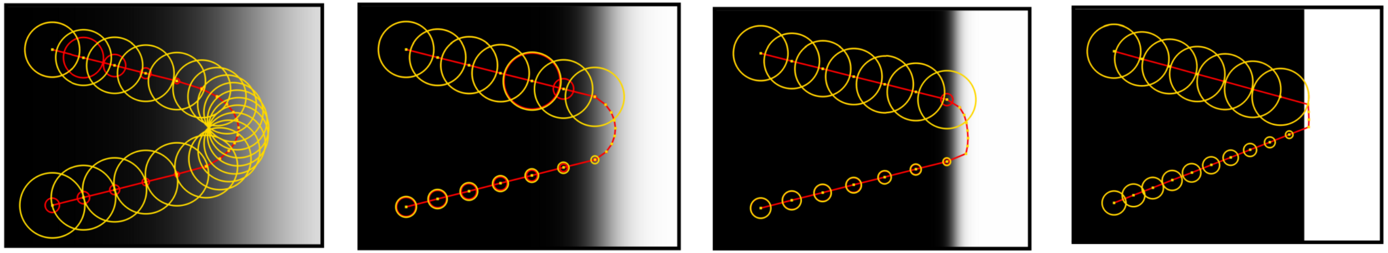
α: Approximation parameter for relaxing discrete sensing constraints
k: Coefficient to control rate of increase of α
τ: Execution time step $(0 \leq \tau \leq \ell)$
ε: Convergence tolerance parameter

Variables:

$\hat{\mathcal{B}}_{0:\ell} = [\hat{\mathbf{b}}_0 \ldots \hat{\mathbf{b}}_\ell]$, $\hat{\mathcal{U}}_{0:\ell-1} = [\hat{\mathbf{u}}_0 \ldots \hat{\mathbf{u}}_{\ell-1}]$: Optimization variables
$\boldsymbol{\delta}_{0:\ell}$: Binary vector to track value of continuous approximation for convergence criterion

for $\tau = 0, \ldots, \ell-1$ **do** ▷ Re-planning loop following the MPC paradigm
  $\alpha \leftarrow \alpha_{\text{init}}$
  while $\boldsymbol{\delta}_{\tau:\ell}$ not within ε tolerance of true binary values $\{0,1\}$ **do**
    Reset trust region size and penalty coefficient ▷ [25]
    $[\hat{\mathcal{B}}_{\tau:\ell}, \hat{\mathcal{U}}_{\tau:\ell-1}] \leftarrow$ SQP-based optimization of approximation given $[\bar{\mathcal{B}}_{\tau:\ell}, \bar{\mathcal{U}}_{\tau:\ell-1}]$ ▷ [25]
    $\alpha \leftarrow k * \alpha$ ▷ α-update to increase noise outside sensing region
    $\hat{\mathbf{b}}_{t+1} = \mathbf{g}(\hat{\mathbf{b}}_t, \hat{\mathbf{u}}_t)\ \forall\ t = \tau, \ldots, \ell-1$ ▷ Integrate belief trajectory after α-update
    Update $\boldsymbol{\delta}_{\tau:\ell} \leftarrow \boldsymbol{\delta}_{\tau:\ell}(\alpha)$ ▷ Eq. (5)
    $[\bar{\mathcal{B}}_{\tau:\ell}, \bar{\mathcal{U}}_{\tau:\ell-1}] \leftarrow [\hat{\mathcal{B}}_{\tau:\ell}, \hat{\mathcal{U}}_{\tau:\ell-1}]$ ▷ Update trajectory initialization
  end while
  Execute $\bar{\mathbf{u}}_\tau$
  Obtain measurement and update $\bar{\mathbf{b}}_{\tau+1}$ using EKF ▷ Eq. (6a)
  Truncate $\bar{\mathbf{b}}_{\tau+1}$ w.r.t sensing region boundary
  Update sensing regions for all sensors
  $\bar{\mathbf{b}}_{t+1} = \mathbf{g}(\bar{\mathbf{b}}_t, \bar{\mathbf{u}}_t)\ \forall\ t = \tau+1, \ldots, \ell-1$ ▷ Integrate belief trajectory after Kalman update
  ▷ using previously optimized control inputs $\bar{\mathcal{U}}_{\tau+1:\ell-1}$
end for

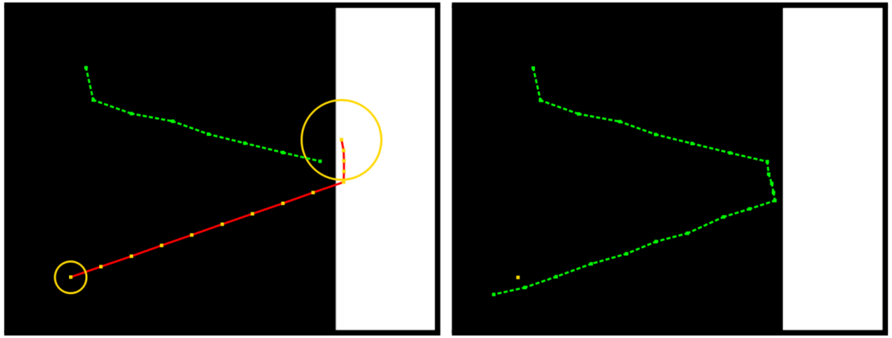
Discontinuities in Sensing Domains



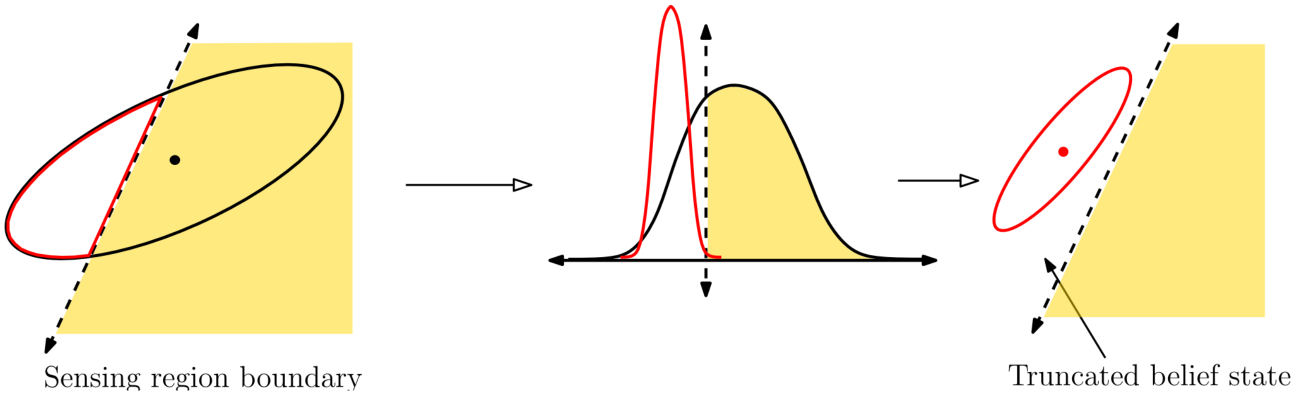
Increasing difficulty →

Noise level determined by signed distance to sensing region * homotopy iteration

However…

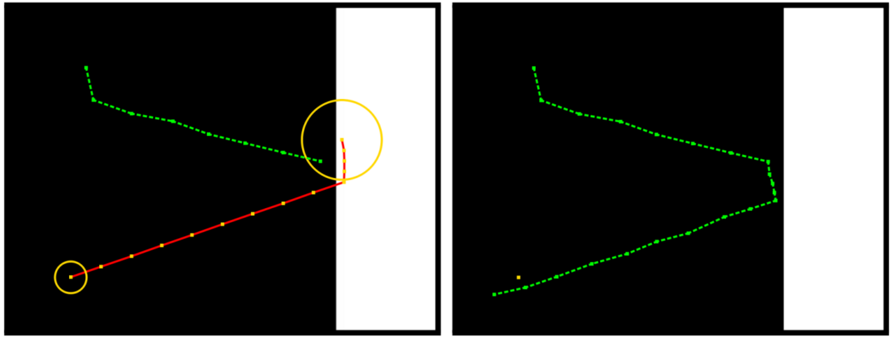


“No measurement” Belief Update

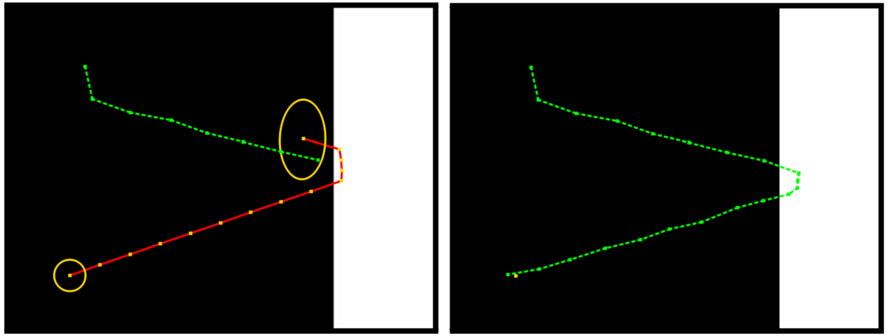


Truncate Gaussian Belief if no measurement obtained

Effect of Truncation

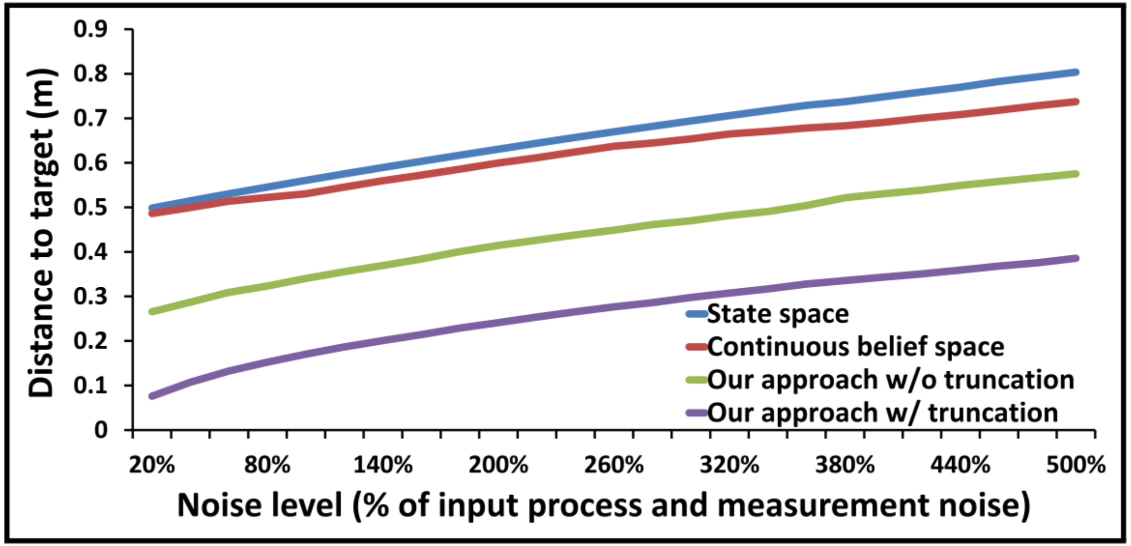


Without "No measurement" Belief Update



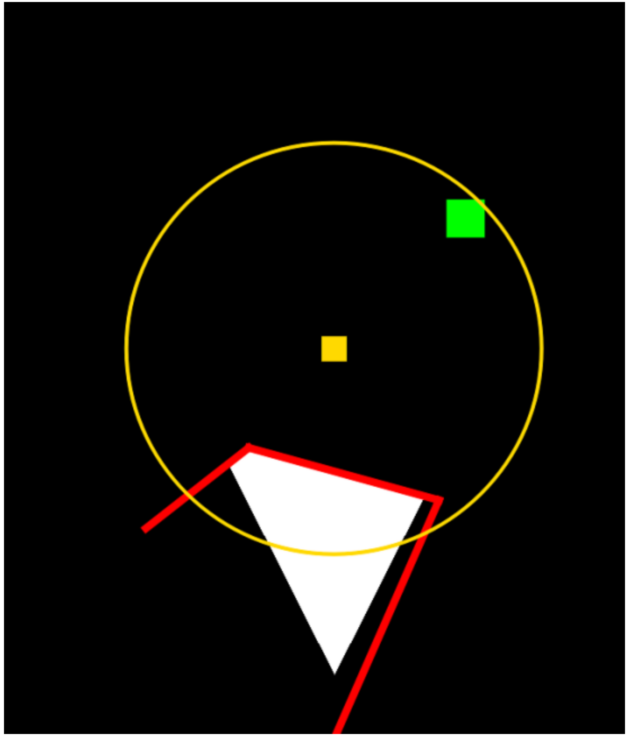
With "No measurement" Belief Update

Experiments

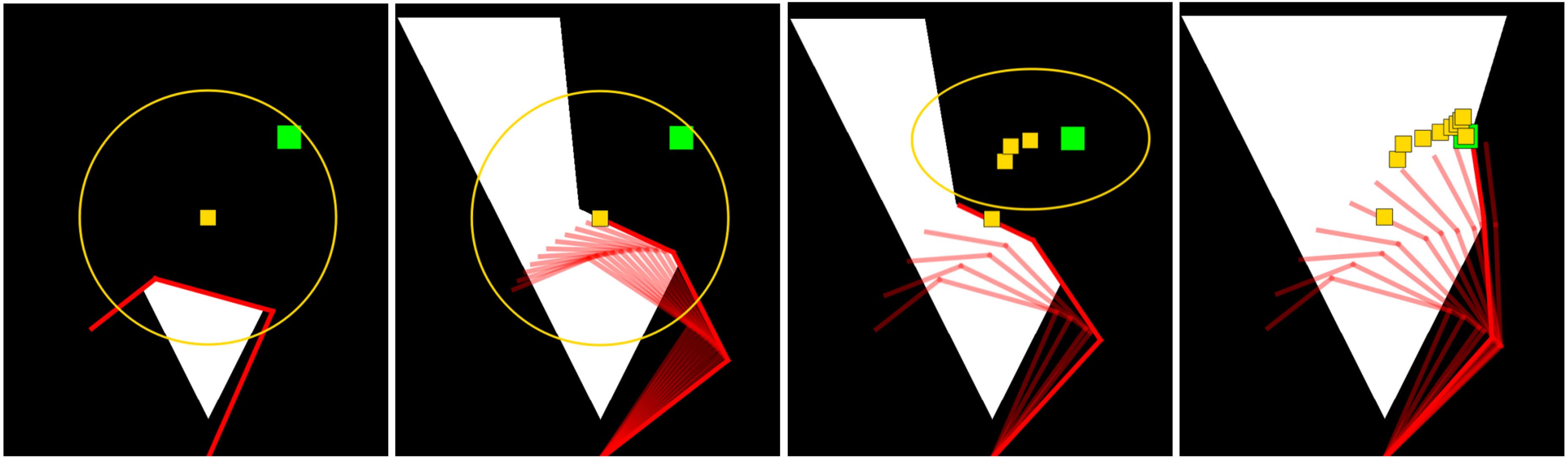


Grasping: Planar 3-link Manipulator

- 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

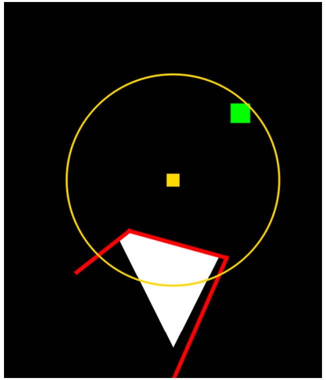
State space plan execution

(way-point)

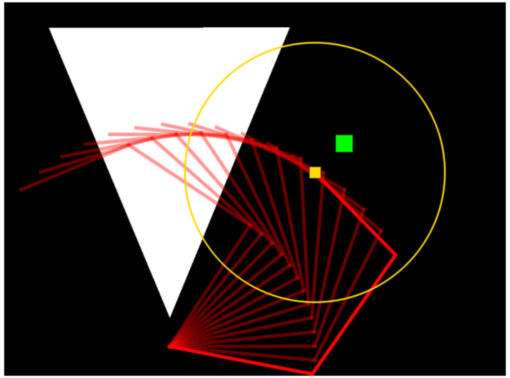
(end)

Belief space plan execution

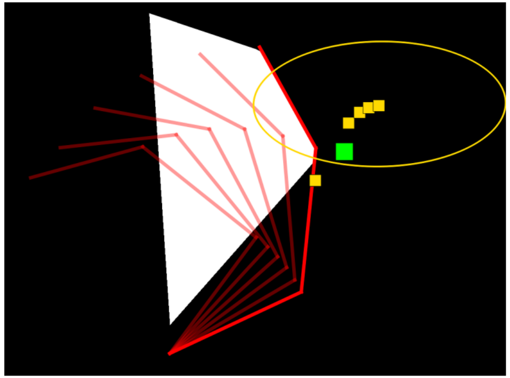
Same Scenario but Movable Camera



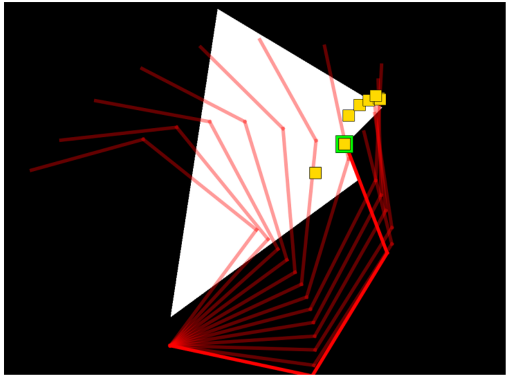
Initial belief



State space
plan execution



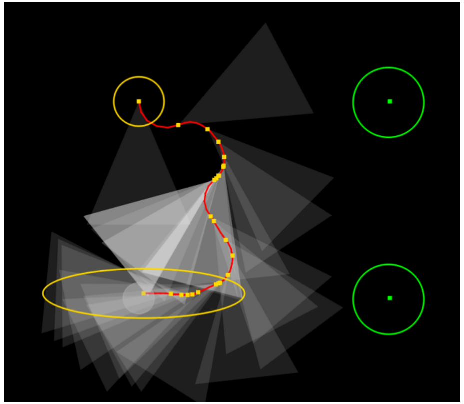
(way-point)

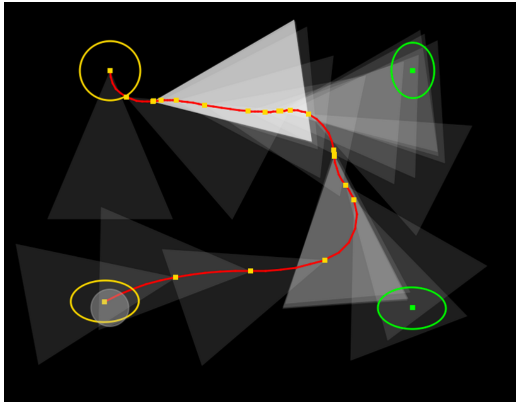


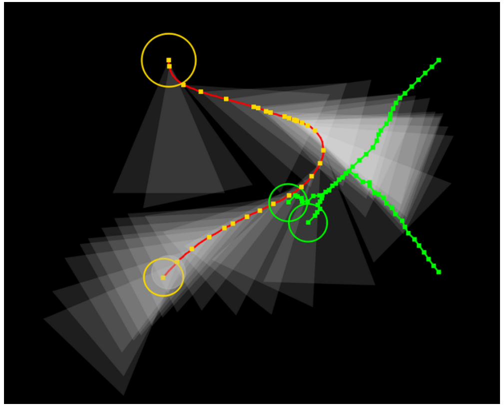
(end)

Belief space plan execution

Car and Landmarks (Active Exploration)

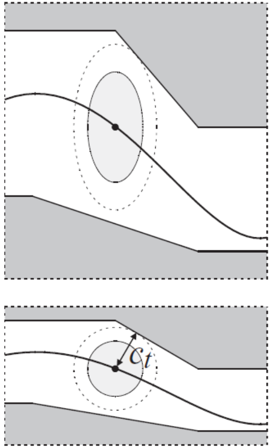






Collision Avoidance

- So far approximating robot geometry as points or spheres

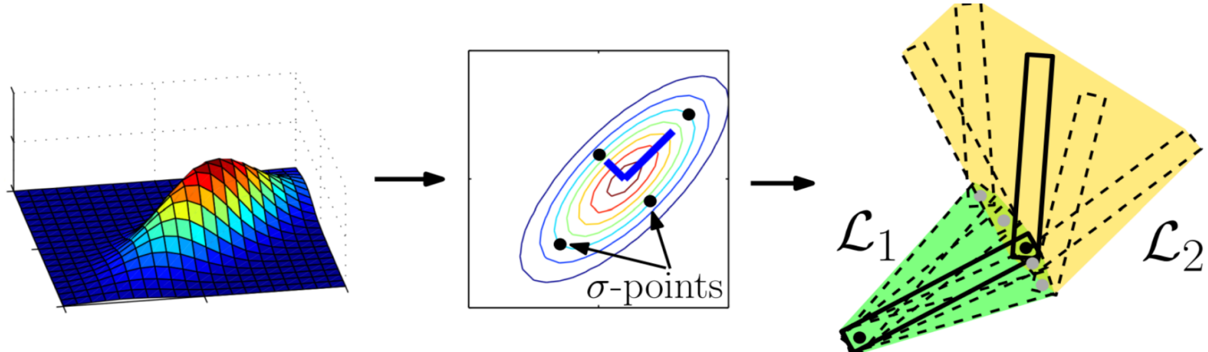


Van den Berg et al.

- Articulated robots cannot be approximated as points/spheres
  - Gaussian noise in joint space
  - Need probabilistic collision avoidance w.r.t robot links

Sigma Hulls

- 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)

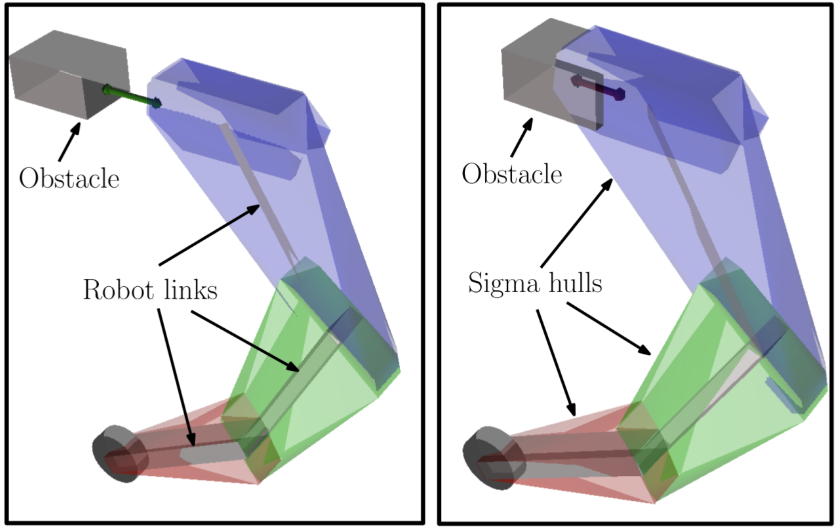


Gaussian belief $\quad$ σ-stddev contours $\quad$ Sigma Hulls (per link)

$$\mathcal{X} = [\hat{\mathbf{x}} \dots \hat{\mathbf{x}}] + \lambda[\mathbf{0}\ \sqrt{\Sigma}\ \ -\sqrt{\Sigma}]$$

Collision Avoidance Constraint

Consider signed distance between obstacle and sigma hulls



(a) Obstacle outside sigma hulls (b) Obstacle overlaps sigma hulls

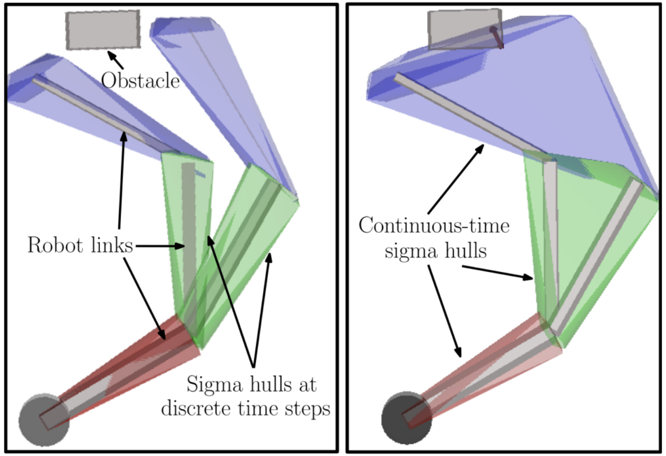
Continuous Collision Avoidance Constraint

- 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

$$\mathtt{sd}(\mathtt{convhull}(\mathcal{A}_t, \mathcal{A}_{t+1}), O) \geq d_{\text{safe}} \quad \forall\, O \in \mathcal{O}$$

- Advantages:
  - Solutions are collision-free in between time-steps
  - Discretized trajectory can have less time-steps



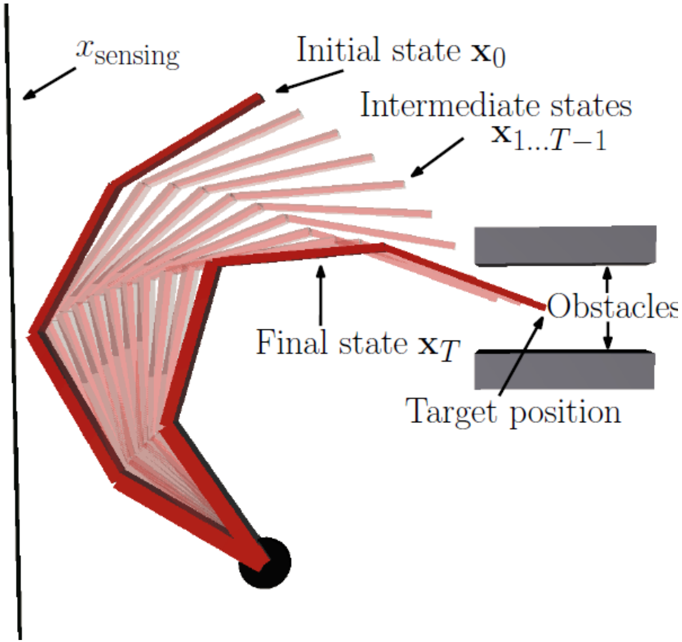
(a) Obstacle does not collide with discrete-time sigma hulls
(b) Obstacle overlaps with continuous-time sigma hulls

Belief Space Model Predictive Control

- 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

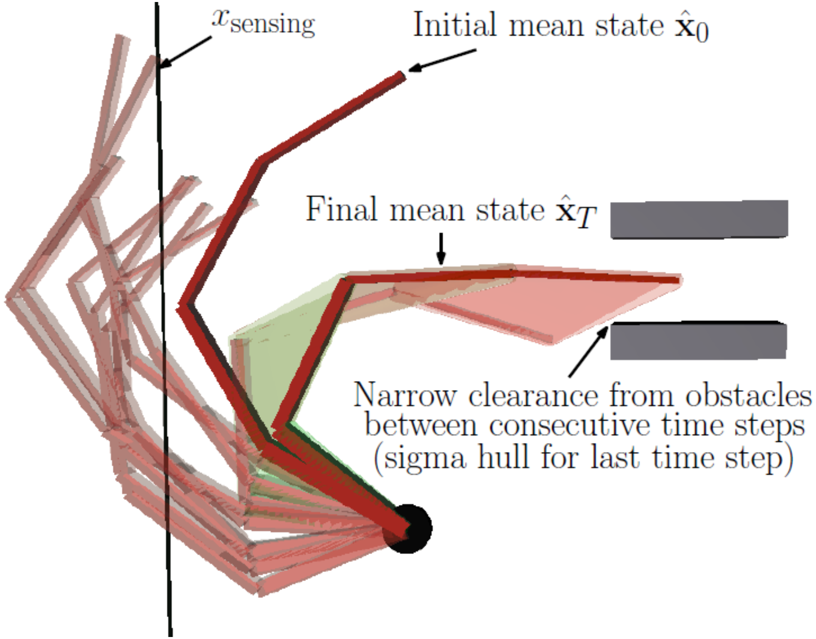
Example: 4-DOF planar robot

State space trajectory



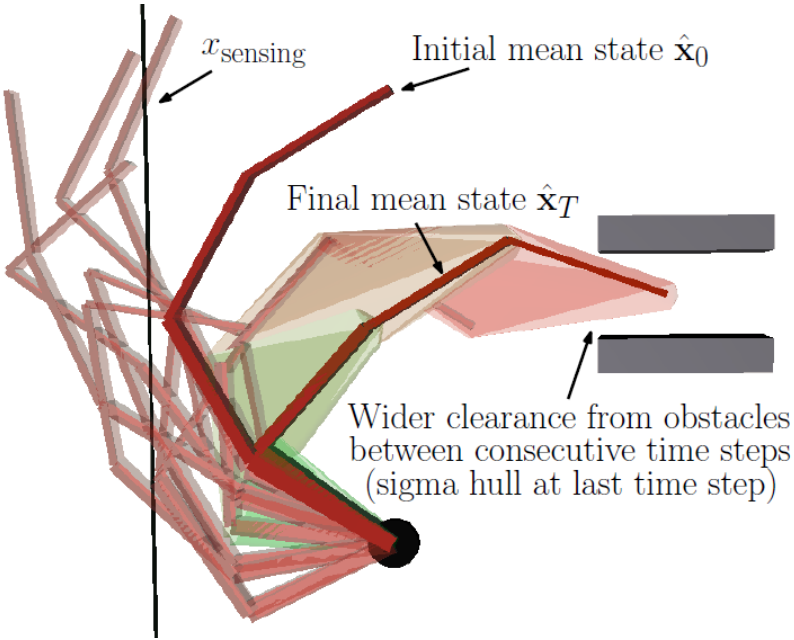
Example: 4-DOF planar robot

1-standard deviation belief space trajectory

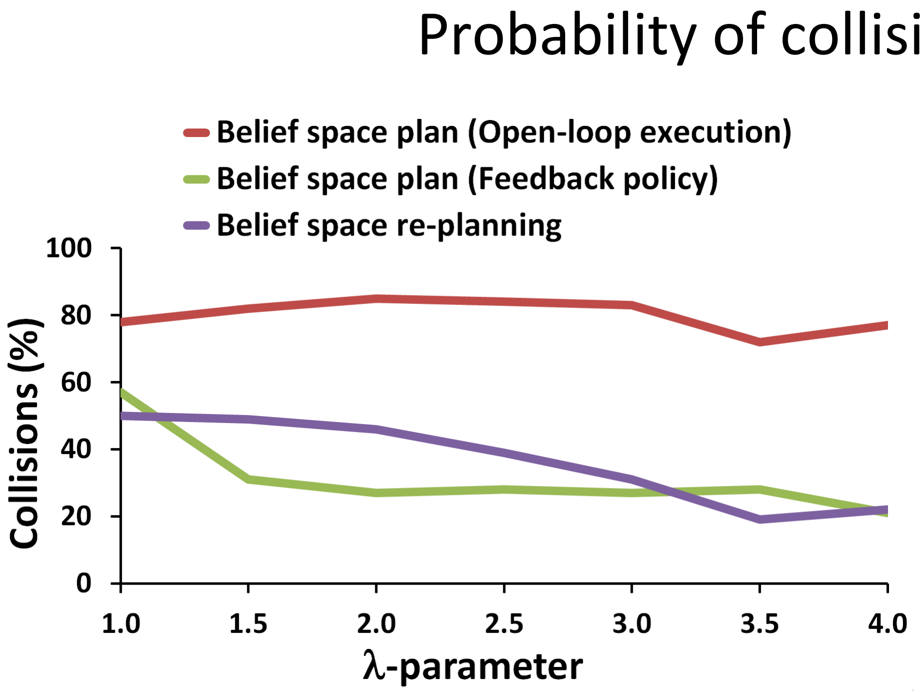


Example: 4-DOF planar robot

4-standard deviation belief space trajectory

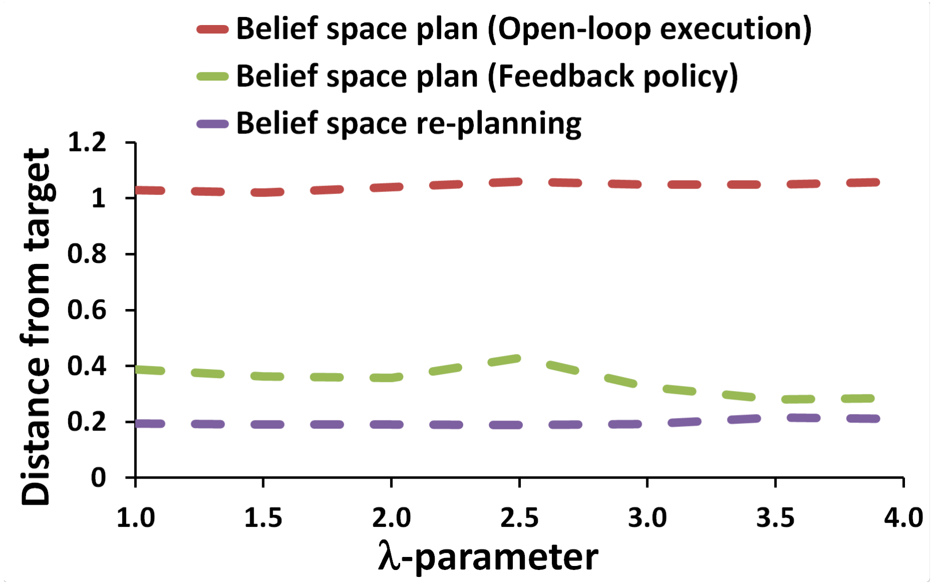


Example: 4-DOF planar robot



Example: 4-DOF planar robot

Mean distance from target



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:
$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + w_t \qquad & w_t \sim \mathcal{N}(0, Q_t) \\ z_t &= Cx_t + v_t & v_t \sim \mathcal{N}(0, R_t) \end{aligned}$$

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

- 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