CSE-571
AI-based Mobile Robotics

Planning and Control:

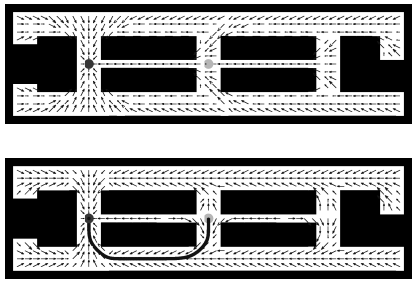
Markov Decision Processes

Problem Classes

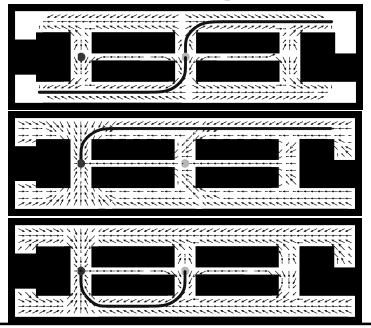
- Deterministic vs. stochastic actions
- Full vs. partial observability

Deterministic, fully observable

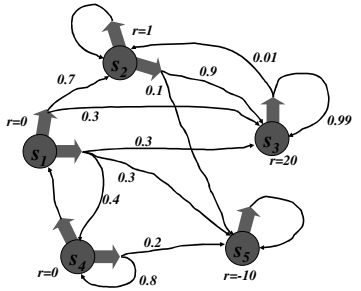
Stochastic, Fully Observable



Stochastic, Partially Observable



Markov Decision Process (MDP)



Markov Decision Process (MDP)

- **Given:**
- States x
- Actions u
- Transition probabilities $p(x'|u,x)$
- Reward / payoff function $r(x,u)$

- **Wanted:**
- Policy $\pi(x)$ that maximizes the future expected reward

Rewards and Policies

- Policy (general case):

$$\pi: \; z_{1:t-1}, u_{1:t-1} \rightarrow u_t$$

- Policy (fully observable case):

$$\pi: \; x_t \rightarrow u_t$$

- Expected cumulative payoff:

$$R_T = E\left[\sum_{\tau=1}^{T} \gamma^{\tau} r_{t+\tau}\right]$$

  - T=1: greedy policy
  - T>1: finite horizon case, typically no discount
  - T=infty: infinite-horizon case, finite reward if discount < 1

Policies contd.

- Expected cumulative payoff of policy:

$$R_T^{\pi}(x_t) = E\left[\sum_{\tau=1}^{T} \gamma^{\tau} r_{t+\tau} \,|\, u_{t+\tau} = \pi(z_{1:t+\tau-1}, u_{1:t+\tau-1})\right]$$

- Optimal policy:

$$\pi^* = \operatorname*{argmax}_{\pi} R_T^{\pi}(x_t)$$

- 1-step optimal policy:

$$\pi_1(x) = \operatorname*{argmax}_{u} \; r(x,u)$$

- Value function of 1-step optimal policy:

$$V_1(x) = \gamma \max_{u} r(x,u)$$

2-step Policies

- Optimal policy:

$$\pi_2(x) = \operatorname*{argmax}_{u} \left[r(x,u) + \int V_1(x') p(x'|u,x) dx'\right]$$

- Value function:

$$V_2(x) = \gamma \max_{u} \left[r(x,u) + \int V_1(x') p(x'|u,x) dx'\right]$$

T-step Policies

- Optimal policy:

$$\pi_T(x) = \operatorname*{argmax}_{u} \left[r(x,u) + \int V_{T-1}(x') p(x'|u,x) dx'\right]$$

- Value function:

$$V_T(x) = \gamma \max_{u} \left[r(x,u) + \int V_{T-1}(x') p(x'|u,x) dx'\right]$$

Infinite Horizon

- Optimal policy:

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

- Bellman equation
- Fix point is optimal policy
- Necessary and sufficient condition

Value Iteration

- for all *x* do

$$\hat{V}(x) \leftarrow r_{\min}$$

- endfor
- repeat until convergence
  - for all *x* do

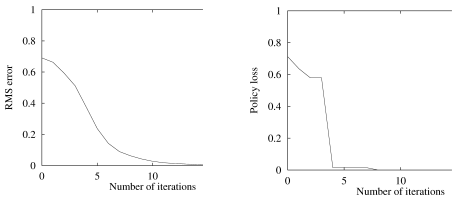
$$\hat{V}(x) \leftarrow \gamma \max_{u} \left[r(x,u) + \int \hat{V}(x') p(x'|u,x) dx' \right]$$

  - endfor
- endrepeat

$$\pi(x) = \arg\max_{u} \left[r(x,u) + \int \hat{V}(x') p(x'|u,x) dx' \right]$$

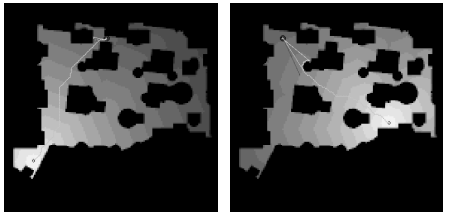
Value Function and Policy

- Value function error and policy error



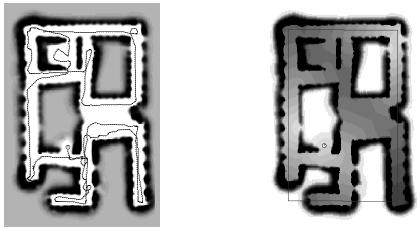
Value Iteration for Motion Planning

(assumes knowledge of robot's location)

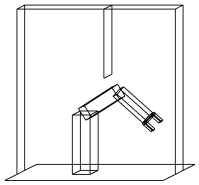


Frontier-based Exploration

- Every unknown location is a target point.



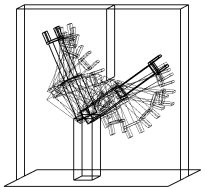
Manipulator Control



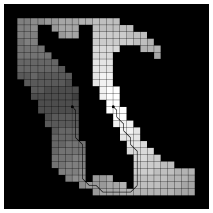
Arm with two joints

Configuration space

Manipulator Control Path

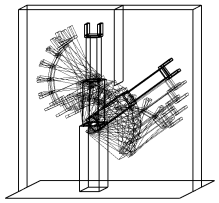


State space

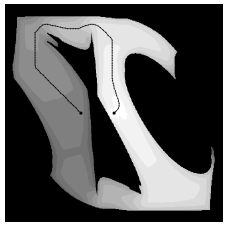


Configuration space

Manipulator Control Path



State space



Configuration space

Collision Avoidance via Planning

- Potential field methods have local minima
- Perform efficient path planning in the local perceptual space
- Path costs depend on length and closeness to obstacles

[Konolige, Gradient method]

Paths and Costs

- Path is list of points $P=\{p_1, p_2, \ldots p_k\}$
- p_k is only point in goal set
- Cost of path is separable into intrinsic cost at each point along with adjacency cost of moving from one point to the next

$$F(P) = \sum_i I(p_i) + \sum_i A(p_i, p_{i+1})$$

  - Adjacency cost typically Euclidean distance
  - Intrinsic cost typically occupancy, distance to obstacle

Navigation Function

- Assignment of potential field value to every element in configuration space [Latombe, 91].
- Goal set is always downhill, no local minima.
- Navigation function of a point is cost of minimal cost path that starts at that point.

$$N_k = \min_{P_k} F(P_k)$$

Computation of Navigation Function

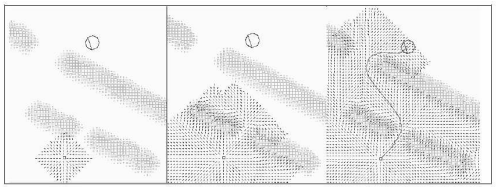
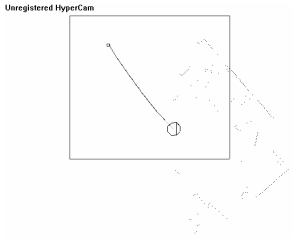


Figure 2. Three stages of the LPN algorithm, starting from a single goal point. The rectangles indicate the intrinsic cost of a point. The gradient direction at each point is shown as a short black line. The images shown are at 10, 30, and 70 iterations. The interpolated path from the robot to the goal is shown in the last image.

Further Details

- Does not consider uncertainty in motion
- Keeps paths away from obstacles by setting intrinsic cost based on distance from obstacles.
- Control is extracted from path by lookahead.
- Local perceptual space depends on quality of sensors and odometry.
- Can easily achieve 10Hz update rate on low-end PC (10 x 10m LPS, 10cm resol.).
- Performs superior to human operator.

Example Run