CS 545 – Optimal Control

- Optimal Control
- Definitions
- Markov Decision Processes

Additional Readings

- *Probabilistic Robotics* – Thrun, Burgard, Fox
- *Reinforcement Learning: A Survey* – Kaelbling, Littman, Moore (1995)
- *Planning and Acting in Partially Observable Stochastic Domains* – Kaelbling, Littman, Cassandra (1998)

Decision-making with Uncertainty

- Before: Addressed uncertainty in robot motion.
  - Kalman Filters
  - Particle Filters
  - Methods for **State Estimation (Tracking)**

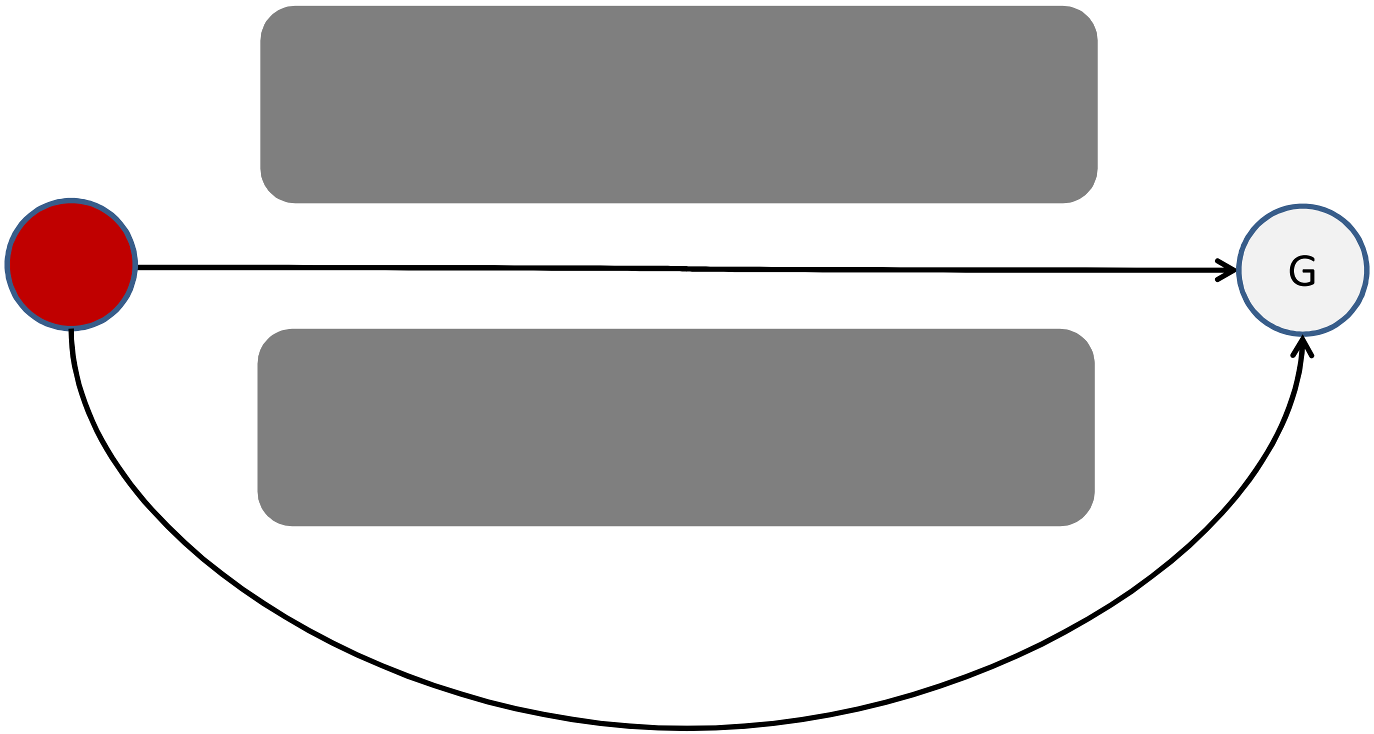
- What about for control?

- How do I know what to do if I don't know where I am and what's going to happen?!

Scenarios

- Industrial robot arm has to pick up pieces which could show up in any orientation
- Mobile robot has to follow shorter, more dangerous route (hallway), or longer, safer router
- Robot arm needs to maneuver through potentially tight area

Navigation Example



Which way is better?

Markov Property

- Markov Property – Conditional probability of future states only depends on current state
  – Don’t need history to make decisions (First-order Markov model)

- *n*th-order Markov model– need history from *n* time steps in past

- Can convert any problem to first order Markov model

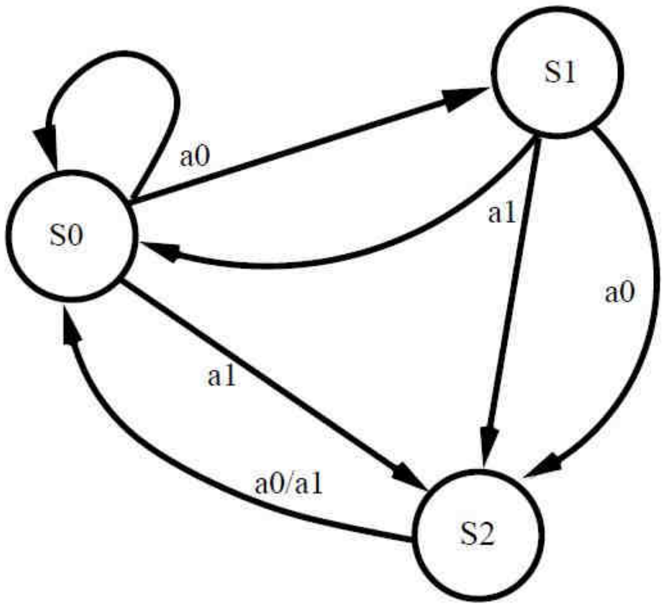
Markov Decision Processes

- $S = \{s_0, s_1, \dots, s_n\}$ – set of states
- $A = \{a_0, a_1, \dots, a_m\}$ – set of **stochastic** actions
- *T(s,a,s')* – transition function *P(s'|s,a)*

$$P(s_{t+1} = s' | s_t = s, a_t = a)$$

- $R(s, a): S \times A \rightarrow \mathbb{R}$ – reward function
- *ϒ* – discount factor
- *π* – policy: *π(s)* → *a*

Markov Decision Processes



$T(s, a_0, s')$		s'		
		s_0	s_1	s_2
s	s_0	0.2	0.8	0.0
	s_1	0.0	0.0	1.0
	s_2	1.0	0.0	0.0

$T(s, a_1, s')$		s'		
		s_0	s_1	s_2
s	s_0	0.0	0.0	1.0
	s_1	0.4	0.0	0.6
	s_2	1.0	0.0	0.0

	$R(s, a_0)$	$R(s, a_1)$
s_0	4	2
s_1	5	7
s_2	10	10

Discount Factor

- $0 \leq \gamma < 1$
- Keeps the total reward finite
- Necessary for planning with infinite horizon
- Idea: Rewards **now** better than rewards **later**

- Total Reward = $r_1 + \gamma r_2 + \gamma^2 r_3 + \ \ldots.$

Models of Optimal Control

- What are we trying to optimize?
- Finite Horizon – for *h* steps

$$E(\sum_{t=0}^{h} r_t)$$

- Infinite Horizon

$$E(\sum_{t=0}^{\infty} \gamma^t r_t)$$

- Goal: Find a policy which maximizes total reward
- Solution is set of simultaneous equations

Bellman Equations

Immediate reward for taking that action

All the states you *could* end up

- $V^*(s) = \max_a [R(s,a) + \gamma(\sum_{s' \in S} T(s,a,s')V^*(s'))]$

The chance of ending up there

How “good” is it to be there?

- $\pi(s) =$
  $argmax_a[\ R(s,a) + \gamma(\sum_{s' \in S} T(s,a,s')V^*(s'))]$

Value Iteration

- Dynamic programming approach to estimate the optimal value function
- Recursively calculate the value of each state
- Use estimate of value function at iteration *n* to calculate new estimate at iteration *n+1*
- Infinite discounted horizon
- *Q(s,a)* – value of taking action *a* in state *s*
  - Take *a* now, then follow π after that

Value Iteration

Initialize *V(s)* # Usual random or uniform

Repeat until $|V_n - V_{n-1}| < \epsilon$

Bellman Residual

Foreach $s \in S$

Foreach $a \in A$

$$Q_n(s,a) = R(s,a) + \gamma(\sum_{s' \in S} T(s,a,s') V_{n-1}(s'))$$

$$V_n(s) = \max_a Q_n(s,a)$$

done

Bellman backup

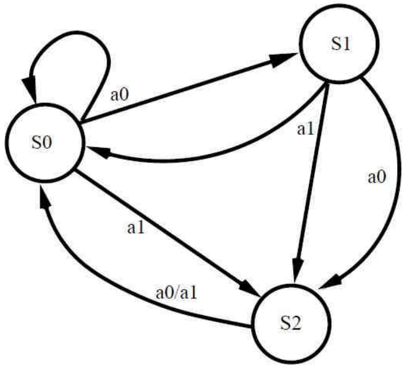
done

done

Value Iteration

- $\pi(s) = argmax_a[\ R(s,a) + \gamma(\sum_{s' \in S} T(s,a,s')V(s'))]$
- Policy is **stationary** and **time independent**
- Can use most recent estimates
  - If a state has already been updated, use most recent estimate
- Each iteration: quadratic in number of states, linear in number of actions
- # of iterations: Polynomial in $\frac{1}{1-\gamma}$
- CAN BE SLOW

Markov Decision Processes



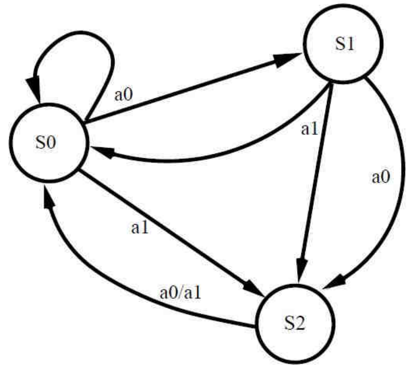
$T(s, a_0, s')$		s'		
		s_0	s_1	s_2
s	s_0	0.2	0.8	0.0
	s_1	0.0	0.0	1.0
	s_2	1.0	0.0	0.0

$T(s, a_1, s')$		s'		
		s_0	s_1	s_2
s	s_0	0.0	0.0	1.0
	s_1	0.4	0.0	0.6
	s_2	1.0	0.0	0.0

	$R(s, a_0)$	$R(s, a_1)$
s_0	4	2
s_1	5	7
s_2	10	10

	a0	a1	V(s)
S0	0	0	0
S1	0	0	0
S2	0	0	0

Markov Decision Processes



$T(s, a_0, s')$		s'		
		s_0	s_1	s_2
s	s_0	0.2	0.8	0.0
	s_1	0.0	0.0	1.0
	s_2	1.0	0.0	0.0

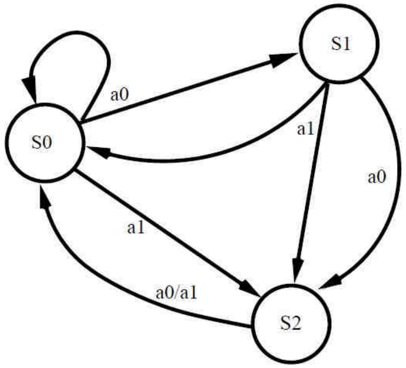
$T(s, a_1, s')$		s'		
		s_0	s_1	s_2
s	s_0	0.0	0.0	1.0
	s_1	0.4	0.0	0.6
	s_2	1.0	0.0	0.0

	$R(s, a_0)$	$R(s, a_1)$
s_0	4	2
s_1	5	7
s_2	10	10

	a0	a1	V(s)
S0	4	2	4
S1	5	7	7
S2	10	10	10

$$Q(s_0, a_0) = 4 + 0.9(0.2 * 0 + 0.8 * 0 + 0 * 0) = 4$$
$$Q(s_0, a_1) = 2 + 0.9(0.0 * 0 + 0.0 * 0 + 1.0 * 0) = 2$$
$$Q(s_1, a_0) = 5 + 0.9(0.0 * 0 + 0.0 * 0 + 1.0 * 0) = 5$$
$$Q(s_1, a_1) = 7 + 0.9(0.4 * 0 + 0.0 * 0 + 0.6 * 0) = 7$$
$$Q(s_2, a_0) = 10 + 0.9(1.0 * 0 + 0.0 * 0 + 0 * 0) = 10$$
$$Q(s_2, a_1) = 10 + 0.9(1.0 * 0 + 0.0 * 0 + 0 * 0) = 10$$

Markov Decision Processes



$T(s, a_0, s')$		s'		
		s_0	s_1	s_2
s	s_0	0.2	0.8	0.0
	s_1	0.0	0.0	1.0
	s_2	1.0	0.0	0.0

$T(s, a_1, s')$		s'		
		s_0	s_1	s_2
s	s_0	0.0	0.0	1.0
	s_1	0.4	0.0	0.6
	s_2	1.0	0.0	0.0

	$R(s, a_0)$	$R(s, a_1)$
s_0	4	2
s_1	5	7
s_2	10	10

	a0	a1	V(s)
S0	5.22	11	11
S1	14	17.9	17.9
S2	13.6	13.6	13.6

$$Q(s_0, a_0) = 4 + 0.9(0.2 * 4 + 0.8 * 7 + 0 * 10) = 5.22$$
$$Q(s_0, a_1) = 2 + 0.9(0.0 * 4 + 0.0 * 7 + 1.0 * 10) = 11$$
$$Q(s_1, a_0) = 5 + 0.9(0.0 * 4 + 0.0 * 7 + 1.0 * 10) = 14$$
$$Q(s_1, a_1) = 7 + 0.9(0.4 * 4 + 0.0 * 7 + 0.6 * 10) = 17.98$$
$$Q(s_2, a_0) = 10 + 0.9(1.0 * 4 + 0.0 * 7 + 0 * 10) = 13.6$$
$$Q(s_2, a_1) = 10 + 0.9(1.0 * 4 + 0.0 * 7 + 0 * 10) = 13.6$$

Policy Iteration

- Choose arbitrary policy, π'
- Repeat

  $\pi = \pi'$

  Compute value function of π

  $V_\pi(s) = R\big(s, \pi(s)\big) + \gamma[\sum_{s' \in S} T(s, \pi(s), s') V_\pi(s')]$

  Improve policy at each state

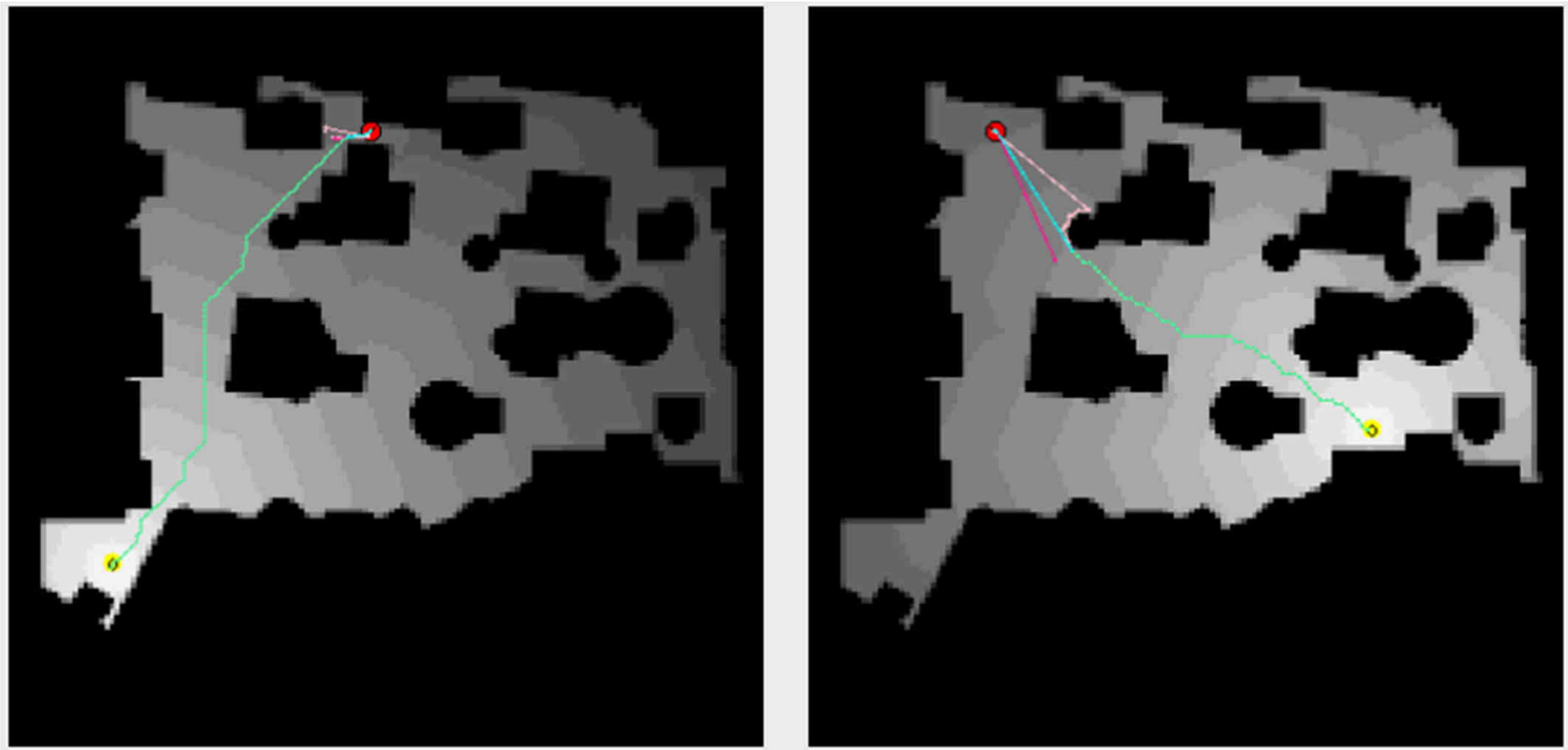
  $\pi'(s) = argmax_a (R(s,a) + \gamma \sum_{s' \in S} T(s, a, s') V_\pi(s'))$

  Until $\pi == \pi'$

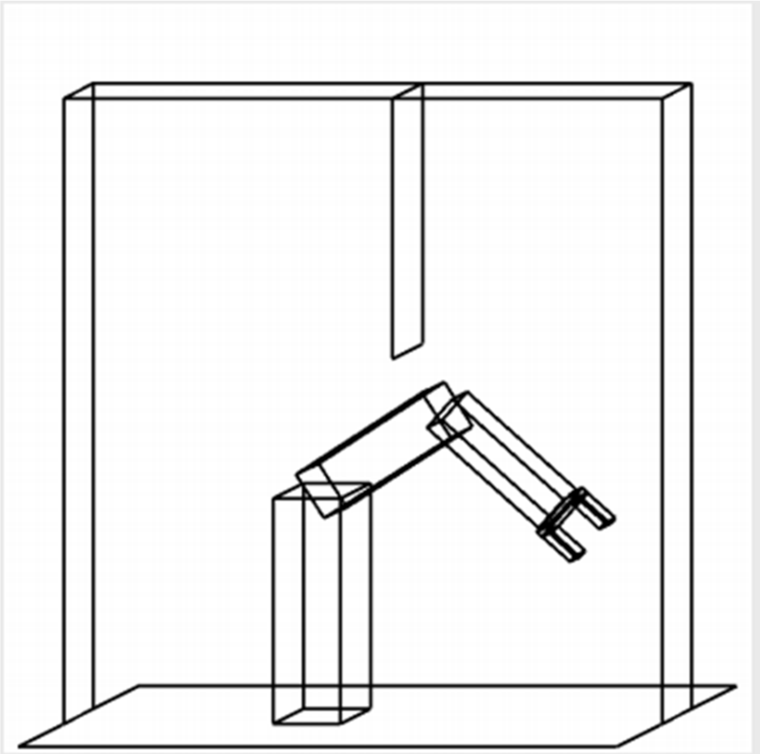
Policy Evaluation

Policy Update

MDP Navigation Policy

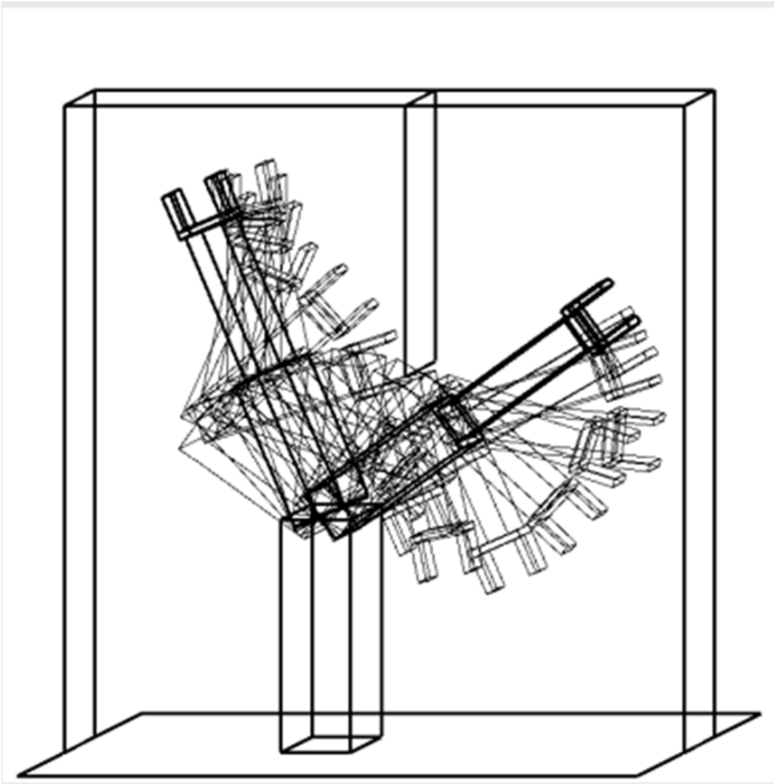


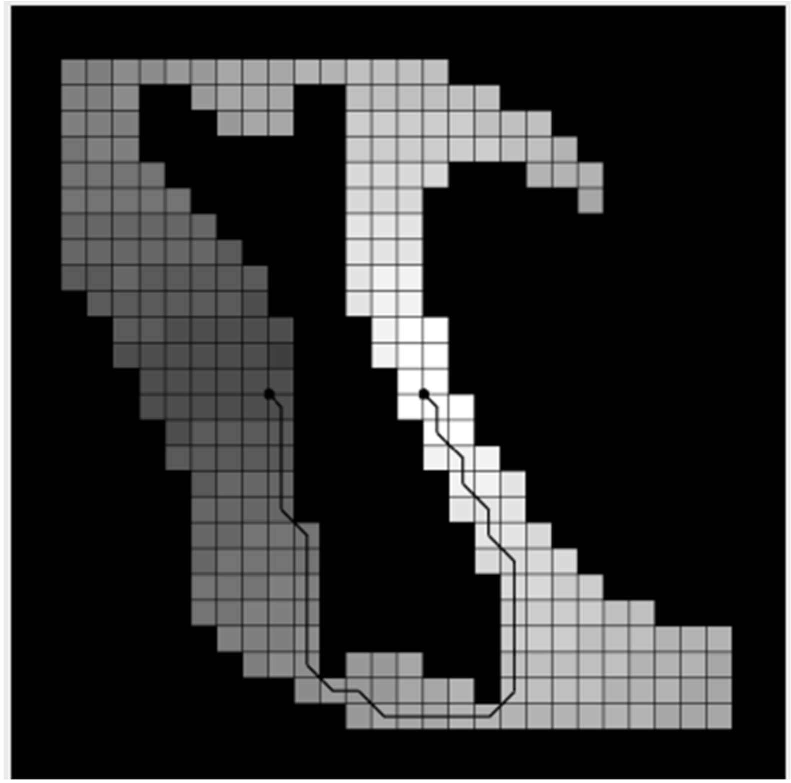
Arm



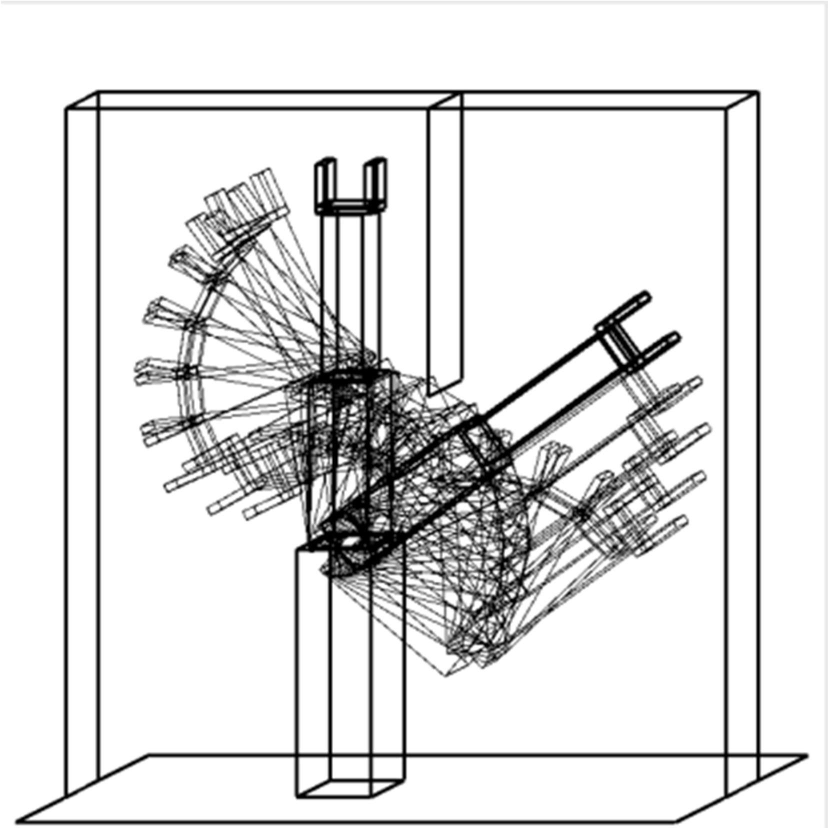
- Two-link robot arm with noisy movement

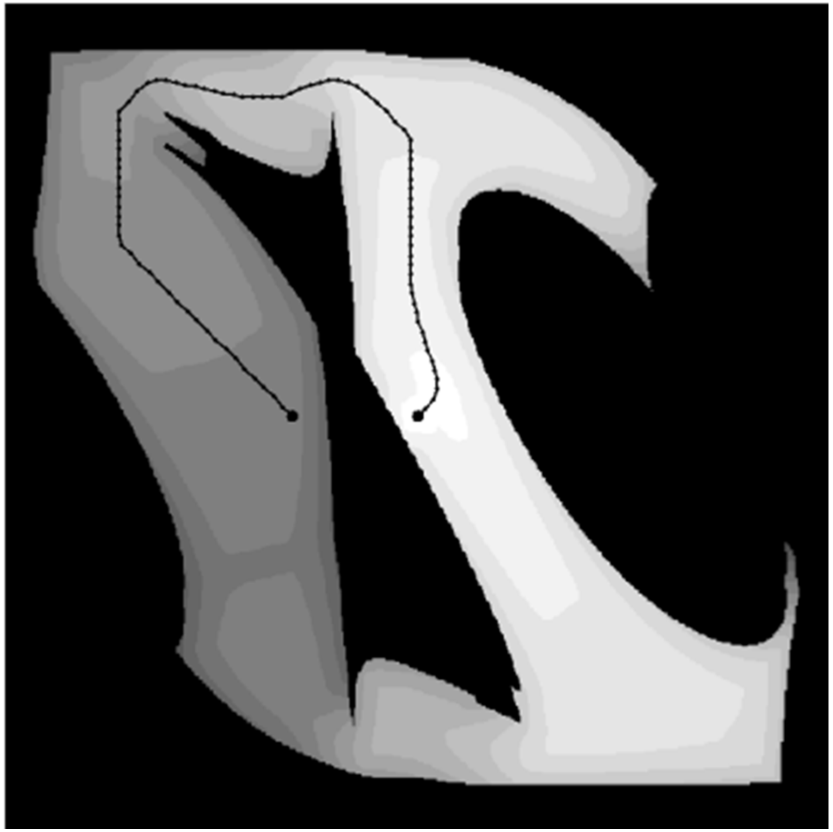
Arm – Discrete Representation





Arm – Discrete Representation





Applications in Robotics

- Good for navigation and control
- Only for low dimensional
  - Higher dimensions require learning, clustering,
- Compare against trajectory planning
- Can represent higher-level reasoning