

MEAM 520

Lecture 13: POMDPs

Cynthia Sung, Ph.D.

Mechanical Engineering & Applied Mechanics

University of Pennsylvania

Lab 3: Trajectory Planning

MEAM 520, University of Pennsylvania

October 3, 2018

This lab consists of two portions, with a pre-lab due on **Wednesday, October 10, by midnight (11:59 p.m.)** and a lab report due on **Wednesday, October 17, by midnight (11:59 p.m.)**. Late submissions will be accepted until midnight on Saturday following the deadline, but they will be penalized by 25% for each partial or full day late. After the late deadline, no further assignments may be submitted; post a private message on Piazza to request an extension if you need one due to a special situation.

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. To help you actually learn the material, what you submit must be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. When you get stuck, post a question on Piazza or go to office hours!

Individual vs. Pair Programming

If you choose to work on the lab in a pair, work closely with your partner throughout the lab, following these guidelines, which were adapted from "All I really needed to know about pair programming I learned in kindergarten," by Williams and Kessler, *Communications of the ACM*, May 2000. This article is available on Canvas under Files / Resources.

- Start with a good attitude, setting aside any skepticism, and expect to jell with your partner.
- Don't start alone. Arrange a meeting with your partner as soon as you can.
- Use just one setup, and sit side by side. For a programming component, a desktop computer with a large monitor is better than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (writing, using the mouse/keyboard, moving the robot) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every 30 minutes, *even if one partner is much more experienced than the other*. You may want to set a timer to help you remember to switch.
- If you notice an error in the equation or code that your partner is writing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.
- Recognize that working in pairs usually takes more time than working alone, but it produces better work, deeper learning, and a more positive experience for the participants.

Lab 3 due **10/17, 11:59 p.m.**

Grasping POMDPs

Kaijen Hsiao and Leslie Pack Kaelbling and Tomás Lozano-Pérez

Abstract—We provide a method for planning under uncertainty for robotic manipulation by partitioning the configuration space into a set of regions that are closed under compliant motions. These regions can be treated as states in a partially observable Markov decision process (POMDP), which can be solved to yield optimal control policies under uncertainty. We demonstrate the approach on simple grasping problems, showing that it can construct highly robust, efficiently executable solutions.

I. INTRODUCTION

A great deal of progress has been made on the problem of planning motions for robots with many degrees of freedom through free space [10], [9], [13]. These methods enable robots to move through complex environments, as long as they are not in contact with the objects in the world. However, as soon as the robot needs to contact the world, in order to manipulate objects, for example, these strategies do not apply. The fundamental problem with planning for motion in contact is that the configuration of the robot and the objects in the world is not exactly known at the outset of execution, and, given the resolution of sensors, it cannot be exactly known. In such cases, traditional open-loop plans (even extended with simple feedback) are not reliable.

It is useful to distinguish between modes of uncertainty that can be effectively modeled, and those that cannot. In situations with unmodelable uncertainty, such as insertion of keys into locks, very fine-grained details of the surfaces can have large effects on the necessary directions of applied forces, and the available sensors can gain little or no information about those surfaces. When the uncertainty is unmodelable, we must fall back to strategies such as “wiggling” the key, which are highly successful without ever building a model of the underlying situation.

Modelable uncertainty, on the other hand, typically occurs at a coarser scale. In attempting to pick up a mug, for example, a robot with vision or range sensing might have a good high-level model of the situation, but significant remaining uncertainty about the pose or shape of the mug. Based on sensor feedback, it can reason about whether the hand is currently in contact with the handle or the cup body, and choose actions that will both gather more information about the situation and make progress toward a desired grasp with a multi-fingered hand.

This research was supported in part by DARPA IPTO Contract FA8750-05-2-0249, “Effective Bayesian Transfer Learning”, and in part by the Singapore-MIT Alliance agreement dated 11/6/98.

Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar Street, Cambridge, MA 02139 {kjhiao, lpk, tip}@csail.mit.edu

An early approach to planning in the presence of modelable uncertainty was developed in [15]. They used a worst-case model of sensor and motion error, and developed a framework for computing conservative plans under these assumptions. This method was computationally complex, and prone to failure due to overconservatism: if there was no plan that would work for all possible configurations consistent with the initial knowledge state, then the entire system would fail.

In this paper, we build on those ideas, addressing the weaknesses in the approach via abstraction and probabilistic representation. By modeling the initial uncertainty using a probability distribution, rather than a set, and doing the same for uncertainties in dynamics and sensing, we are in a position to make trade-offs when it is not possible to succeed in every possible situation. We can choose plans that optimize a variety of different objective functions involving those probabilities, including, most simply, the plan most likely to achieve the goal. The probabilistic representation also affords an opportunity for enormous computational savings through a focus on the parts of the space that are most likely to be encountered.

By building an abstraction of the underlying continuous configuration and action spaces, we lose the possibility of acting optimally, but gain an enormous amount in computational simplification, making it feasible to compute solutions to real problems. Concretely, we will use methods of model minimization to create an abstract model of the underlying configuration space, and then model the problem of choosing actions under uncertainty as a *partially observable Markov decision process* [25].

II. BACKGROUND AND APPROACH

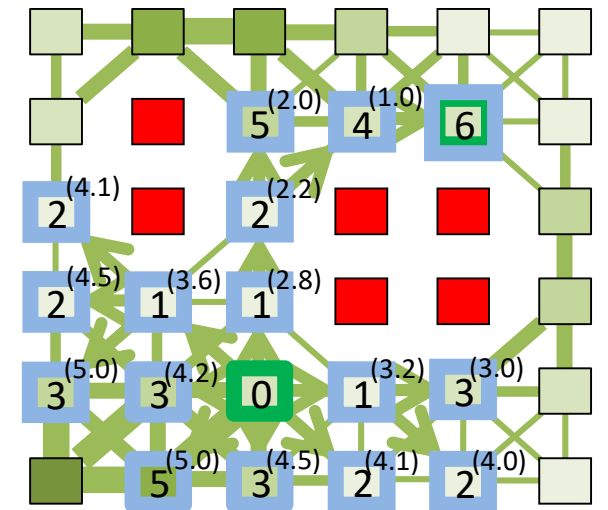
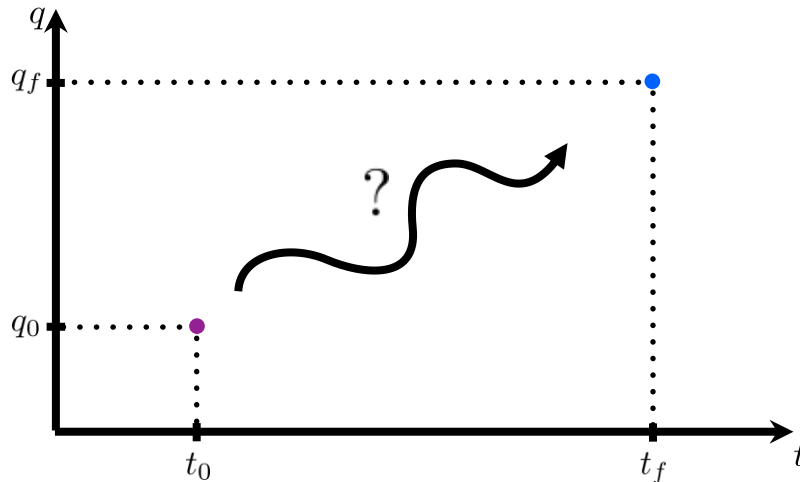
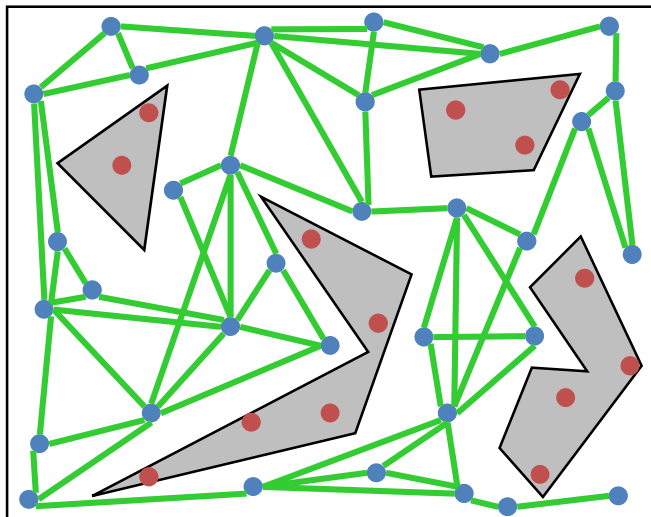
The approach we outline here applies to any domain in which a robot is moving or interacting with other objects and there is non-trivial uncertainty in the configuration. In this paper, we concentrate on the illustrative problem of a robot arm and hand performing pick-and-place operations. We assume that the robot’s position in the global frame is reasonably well known, but that there is some uncertainty about the relative pose and/or shape of the object to be manipulated. Additionally, we assume that there are tactile and/or force sensors on the robot that will enable it to perform compliant motions and to reasonably reliably detect when it makes or loses contacts. We frame this problem primarily as a planning problem. That is, we assume that a reasonably accurate model of the task dynamics and sensors is known, and that the principal uncertainty is in the configuration of the robot and the state of the objects in

Hsiao, K and Kaelbling, LP, and Lozano-Pérez, T. “Grasping POMDPs.” ICRA 2007.

Recap of planning so far

Planning strategy:

1. Convert your free C-space into a graph/roadmap
2. Find a path from q_{start} to a node q_a that is in the roadmap
3. Find a path from q_{goal} to a node q_b that is in the roadmap
4. Search the roadmap for a path from q_a to q_b



Noise and Uncertainty

? question ☆

42 views

Lyric position tracking issue

The position tracking on Lyric doesn't seem to be working. My position tracking works perfectly fine in simulation, but is very off when I use Lyric, which leads me to believe there might be a problem with Lyric

lab5

? question ☆

39 views

Lyric bot's sensors awry

Looks like Lyric's joint 1 sensor is non-functional and the other two are misreporting values. What do we do?

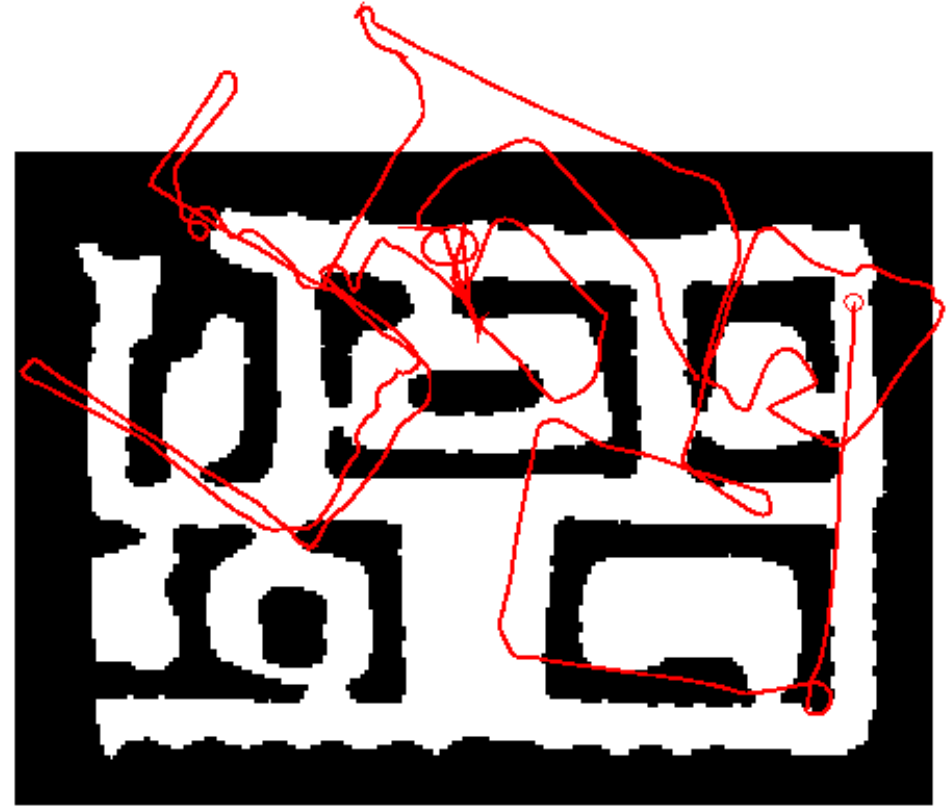
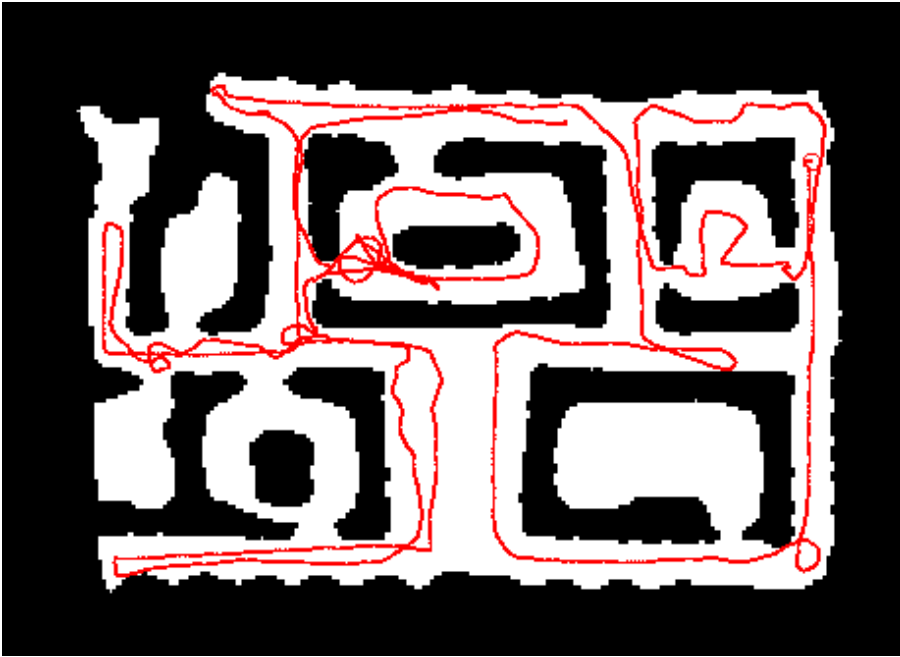
lab5

It can be dangerous to rely on just your planner.

Robots can malfunction, be uncalibrated, have noisy sensors, ...

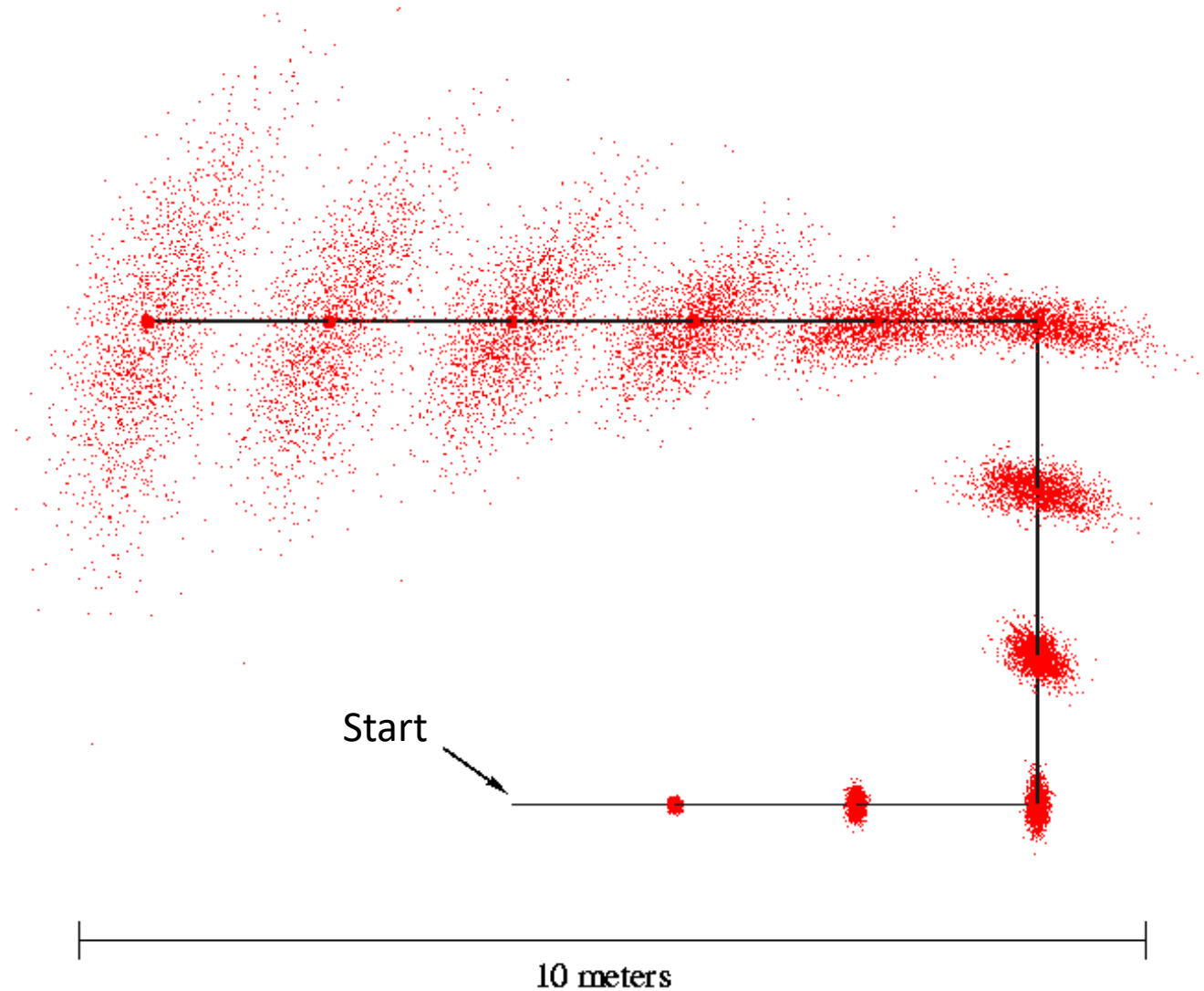


What is uncertainty?



Dead reckoning is not very good!

In the absence of new info, uncertainty increases over time

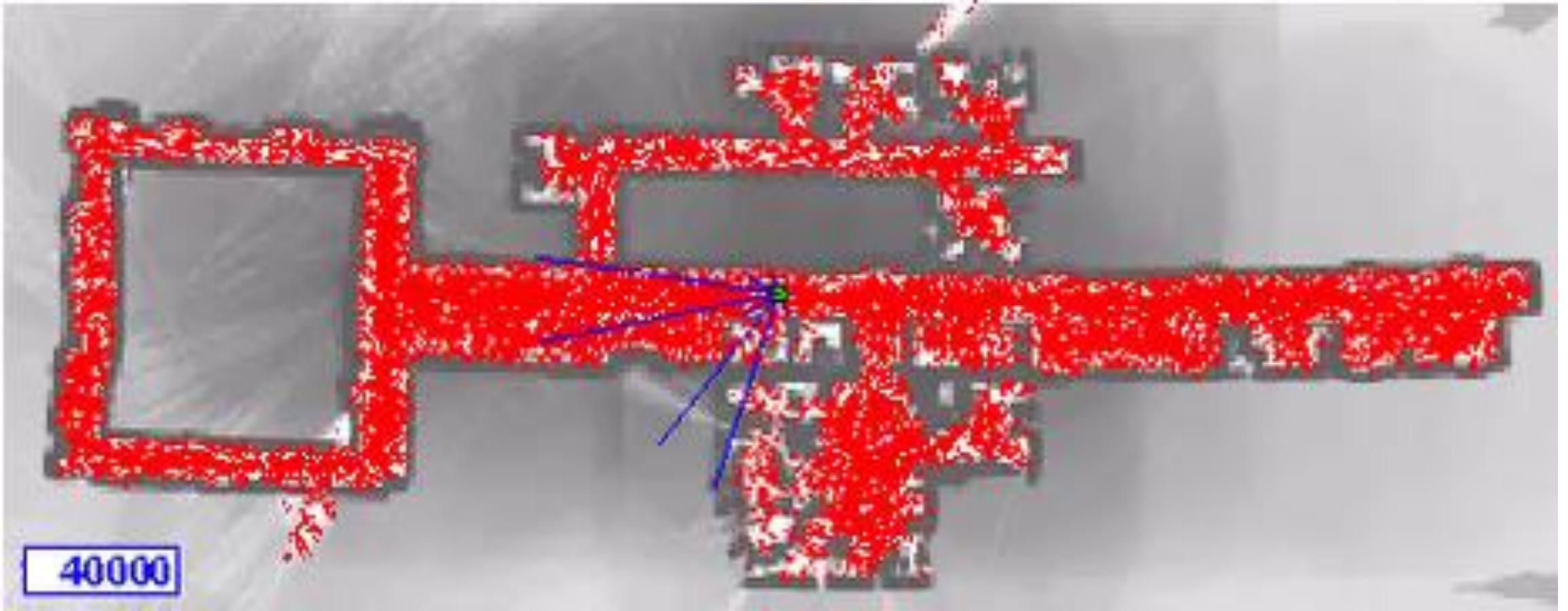


Can we plan to reduce uncertainty?

“There are two ways to reduce uncertainty through action:

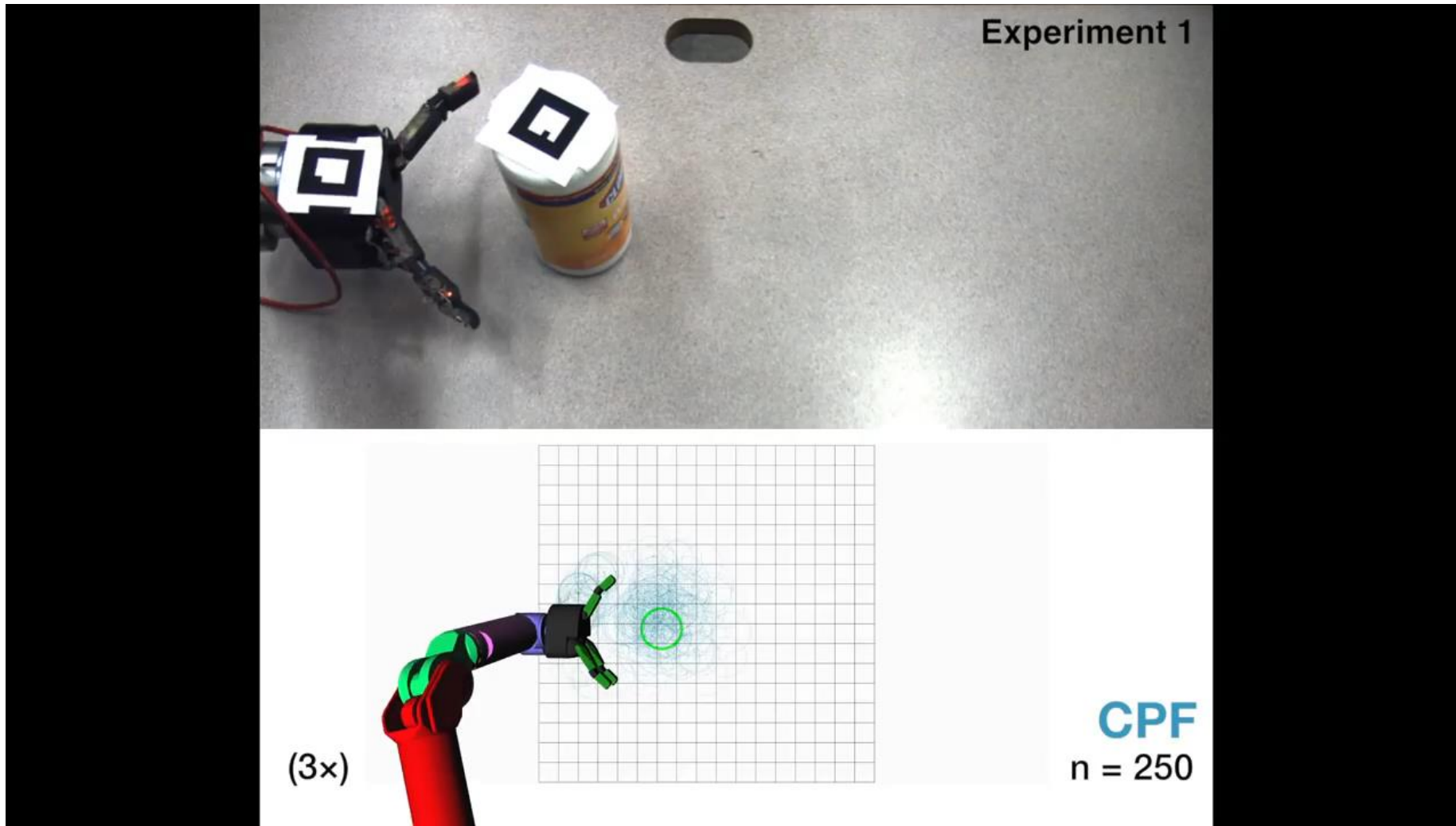
- one is to act to obtain observations that contain information about the underlying state;
- the other is to take actions that are ‘funnels’...

Obtain observations



D. Fox. Adapting the sample size in particle filters through KLD-sampling. *International Journal of Robotics Research*, 2003.

Use funnels

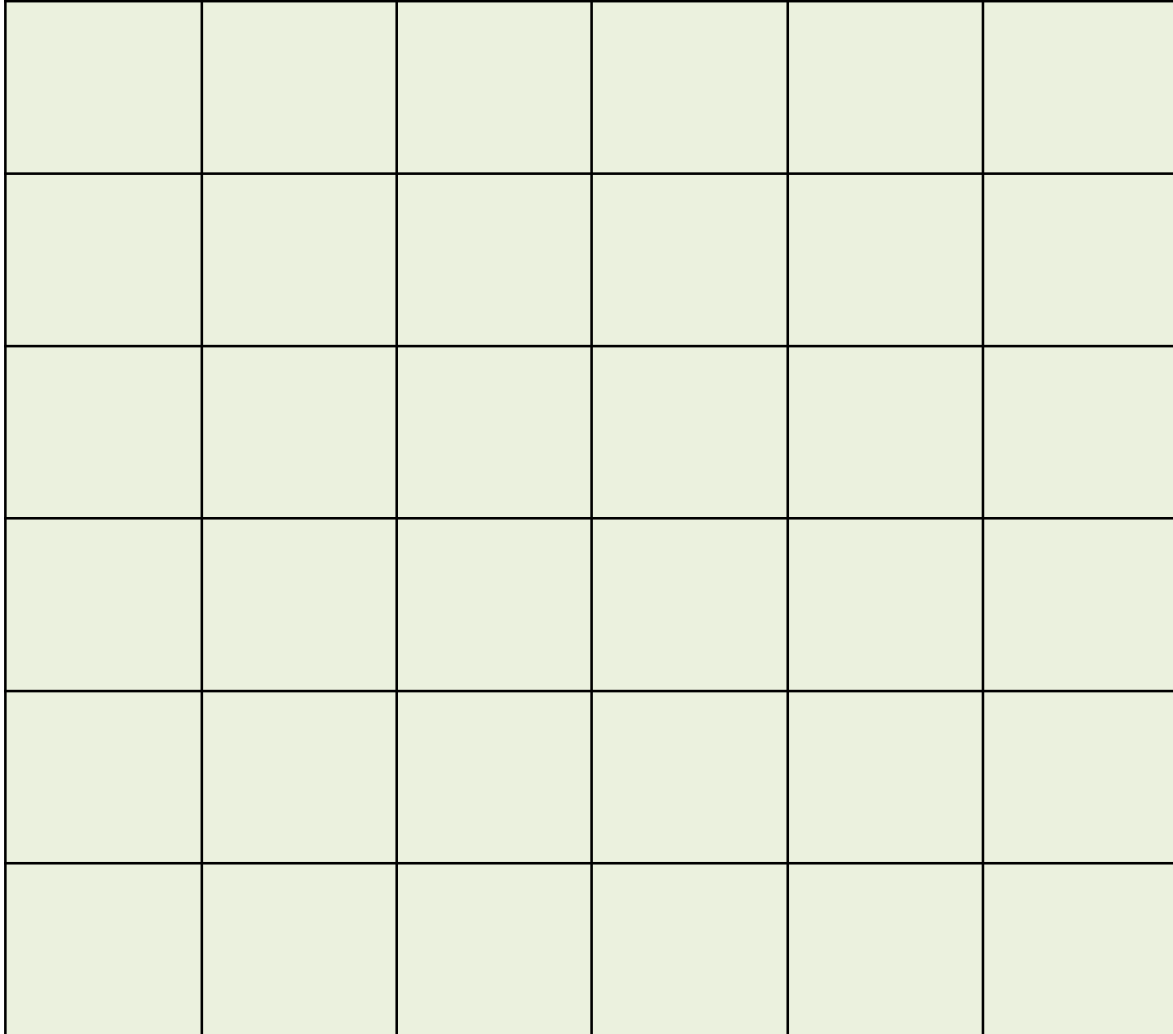


Pose Estimation for Contact Manipulation with Manifold Particle Filters. Michael C. Koval, Mehmet R. Dogar, Nancy S. Pollard, Siddhartha S. Srinivasa. IROS 2013

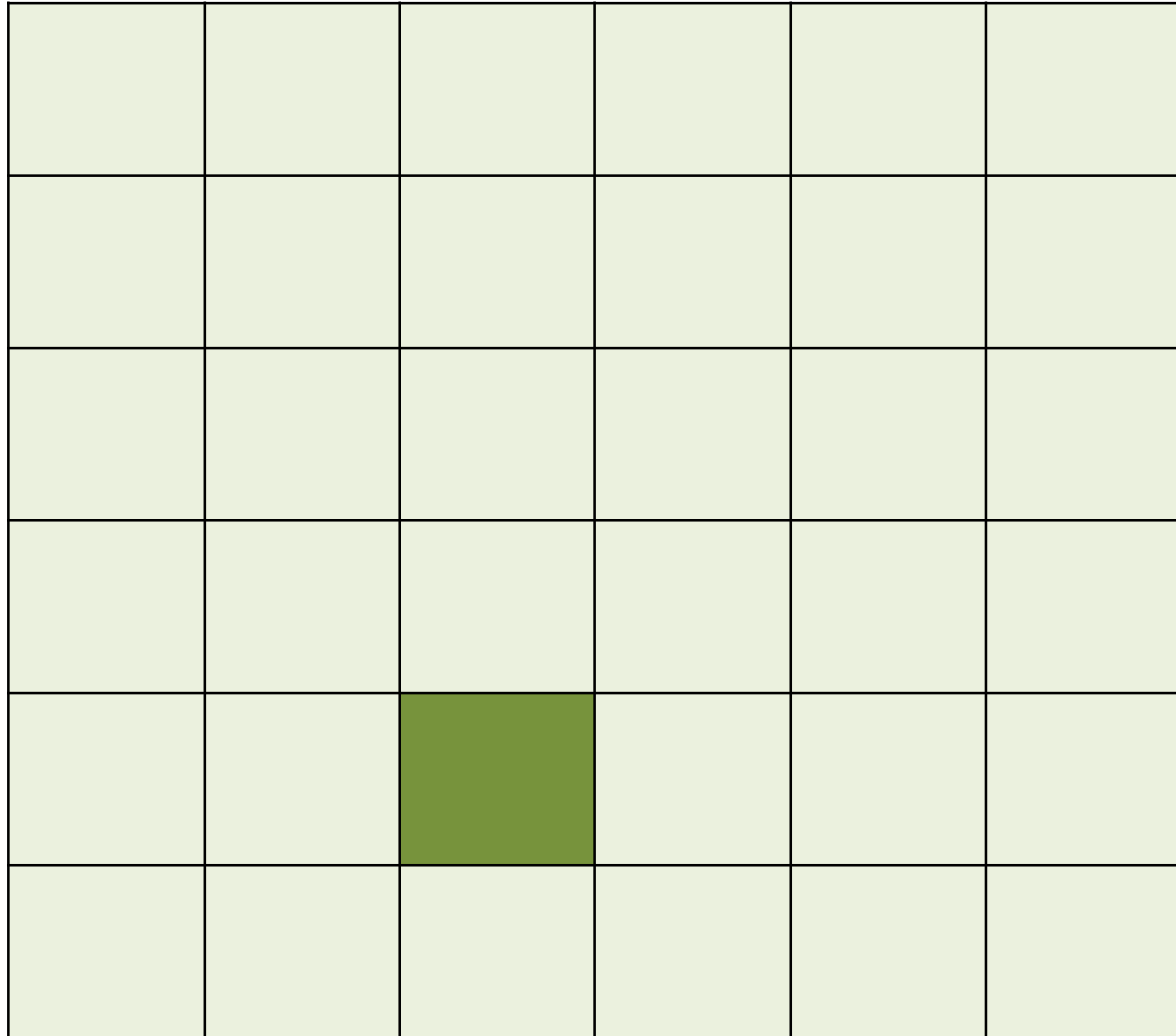
This paper: Planning with POMDPs

A POMDP is a **partially observable Markov decision process**

Let's start with a Markov Decision Process (MDP)

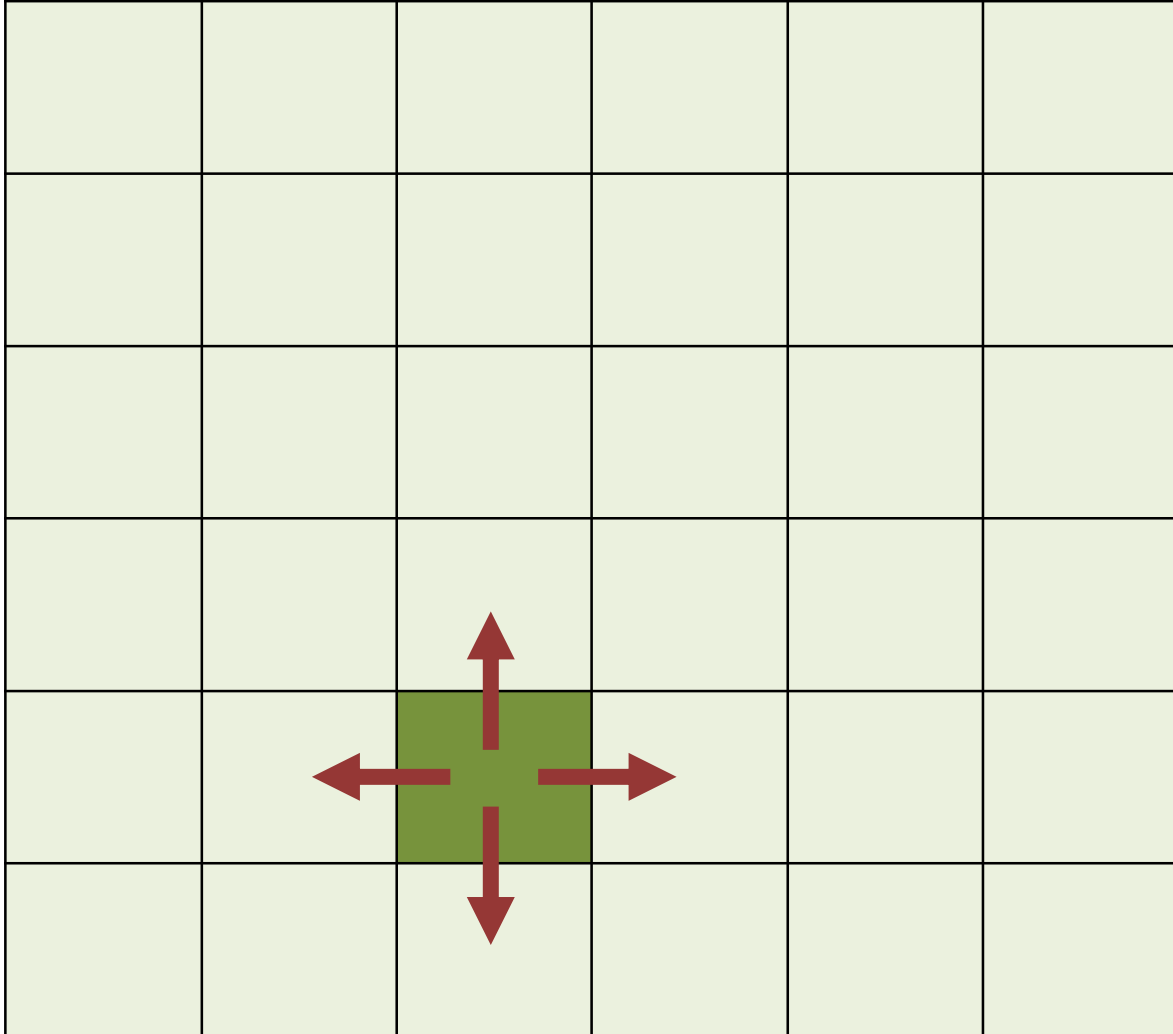


Let's start with a Markov Decision Process (MDP)



Robot has a **state**

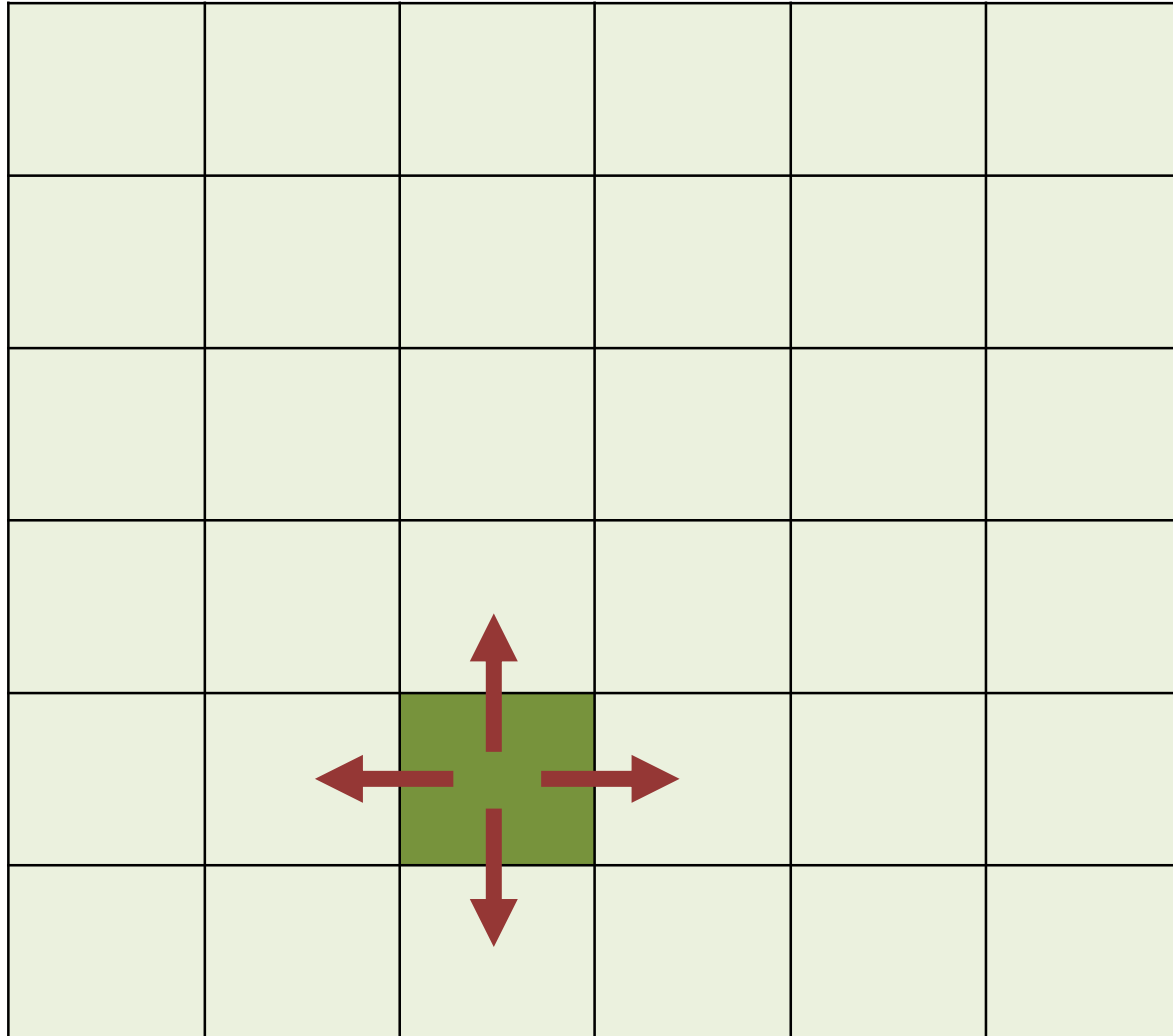
Let's start with a Markov Decision Process (MDP)



Robot has a **state**

It can take 4 **actions**

Let's start with a Markov Decision Process (MDP)



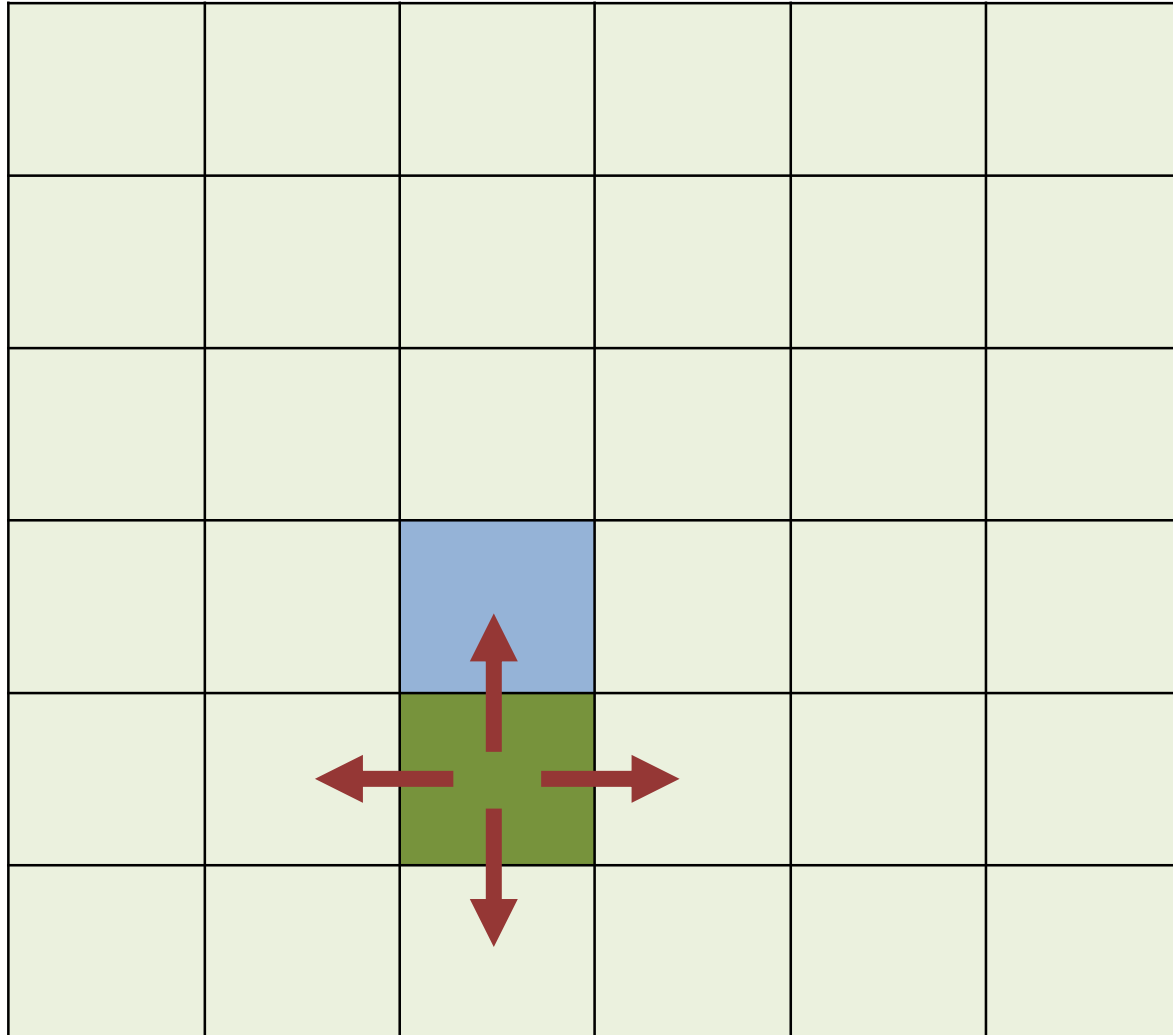
Robot has a **state**

It can take 4 **actions**

Regular path planning:

Do a search to find the sequence of actions to get to the **goal**

Let's start with a Markov Decision Process (MDP)



Robot has a **state**

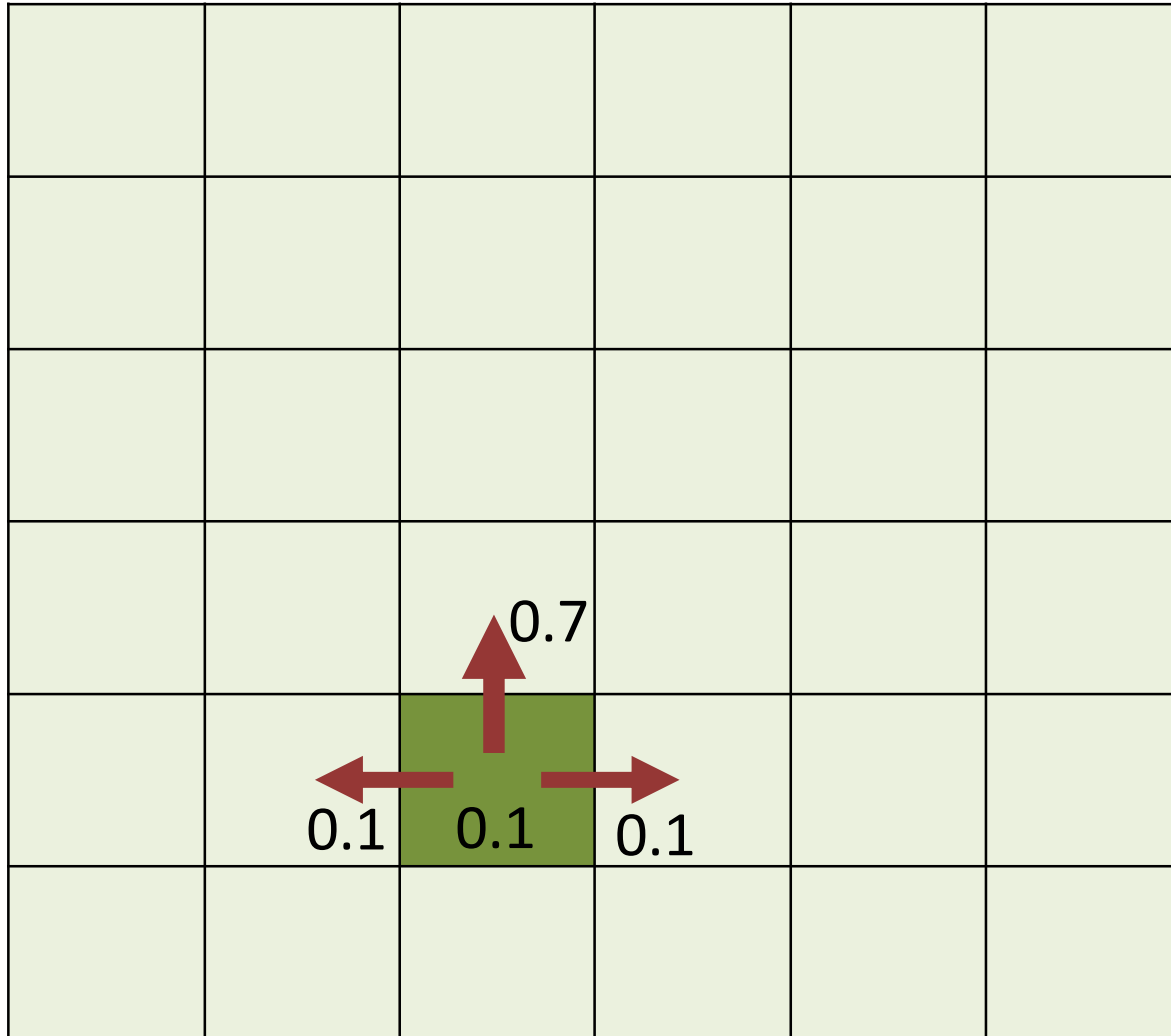
It can take 4 **actions**

Regular path planning:

Do a search to find the sequence of actions to get to the **goal**

What if the robot doesn't always follow your instructions?

Let's start with a Markov Decision Process (MDP)



Robot has a **state**

It can take 4 **actions**

Regular path planning:

Do a search to find the sequence of actions to get to the **goal**

What if the robot doesn't always follow your instructions?

Let's start with a Markov Decision Process (MDP)

		1.0			

Robot has a **state**

It can take 4 **actions**

Regular path planning:

Do a search to find the
sequence of actions to get to
the **goal**

What if the robot doesn't
always follow your instructions?

Let's start with a Markov Decision Process (MDP)

		0.7			
	0.1	0.1	0.1		

Robot has a **state**

It can take 4 **actions**

Regular path planning:

Do a search to find the sequence of actions to get to the **goal**

What if the robot doesn't always follow your instructions?



Let's start with a Markov Decision Process (MDP)

		0.49			
	0.14	0.14	0.14		
0.01	0.02	0.03	0.02	0.01	

Robot has a **state**

It can take 4 **actions**

Regular path planning:

Do a search to find the sequence of actions to get to the **goal**

What if the robot doesn't always follow your instructions?



Let's start with a Markov Decision Process (MDP)

		0.343			
	0.147	0.147	0.147		
0.021	0.042	0.063	0.042	0.021	
0.004	0.006	0.007	0.006	0.003	0.001

Robot has a **state**

It can take 4 **actions**

Regular path planning:

Do a search to find the sequence of actions to get to the **goal**

What if the robot doesn't always follow your instructions?



Let's start with a Markov Decision Process (MDP)

		0.0343			
	0.0147	0.049	0.2548		
0.0021	0.0189	0.1582	0.1218	0.1050	
0.0025	0.0342	0.0511	0.0636	0.0318	0.0148
0.0025	0.0076	0.0112	0.0097	0.0066	0.0029
0.0004	0.0006	0.0007	0.0006	0.0003	0.0001

Robot has a **state**

It can take 4 **actions**

Regular path planning:

Do a search to find the sequence of actions to get to the **goal**

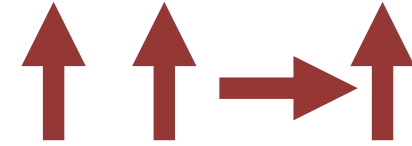
What if the robot doesn't always follow your instructions?



Does the sequence of steps matter?

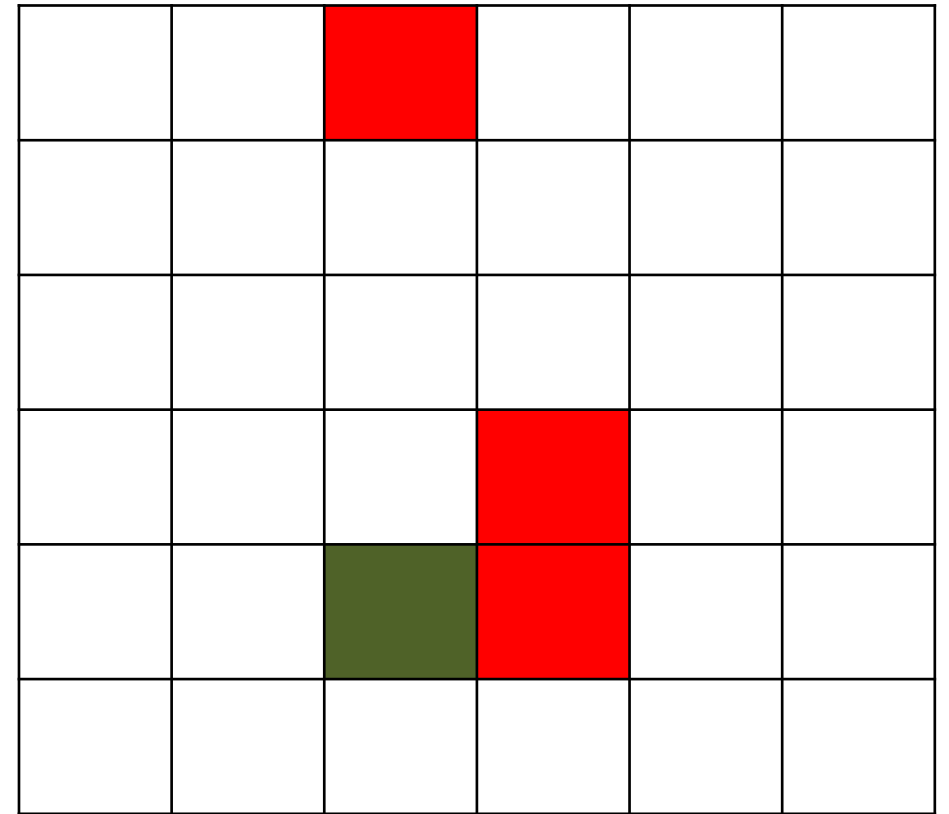
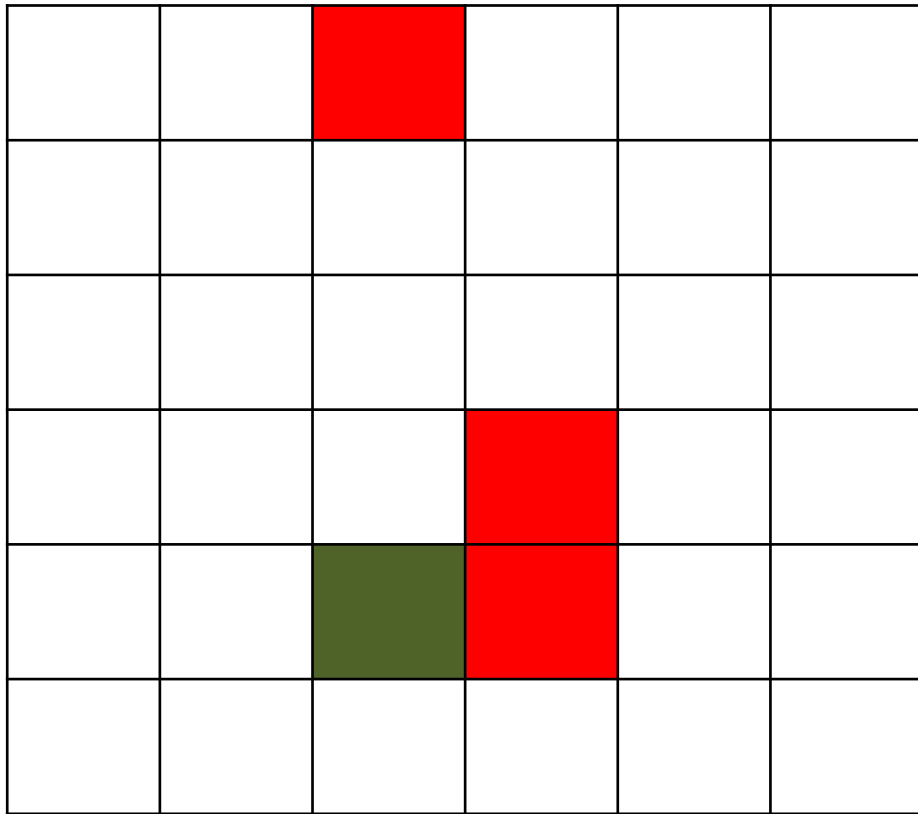


		0.0343			
	0.0147	0.049	0.2548		
0.0021	0.0189	0.1582	0.1218	0.1050	
0.0025	0.0342	0.0511	0.0636	0.0318	0.0148
0.0025	0.0076	0.0112	0.0097	0.0066	0.0029
0.0004	0.0006	0.0007	0.0006	0.0003	0.0001

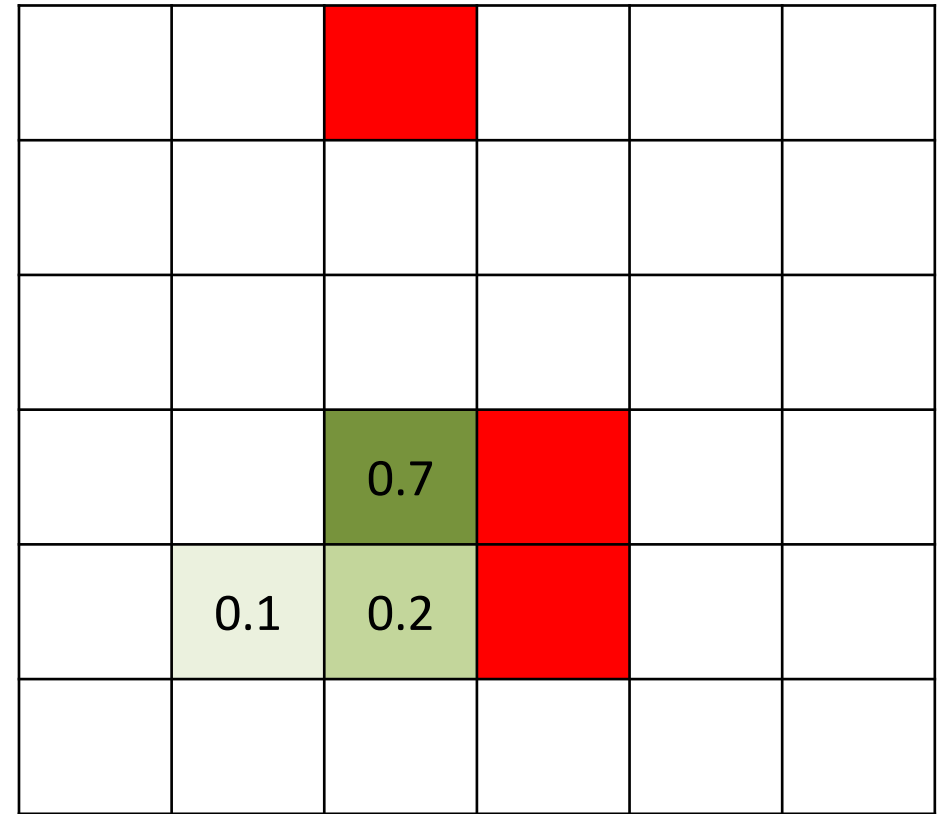
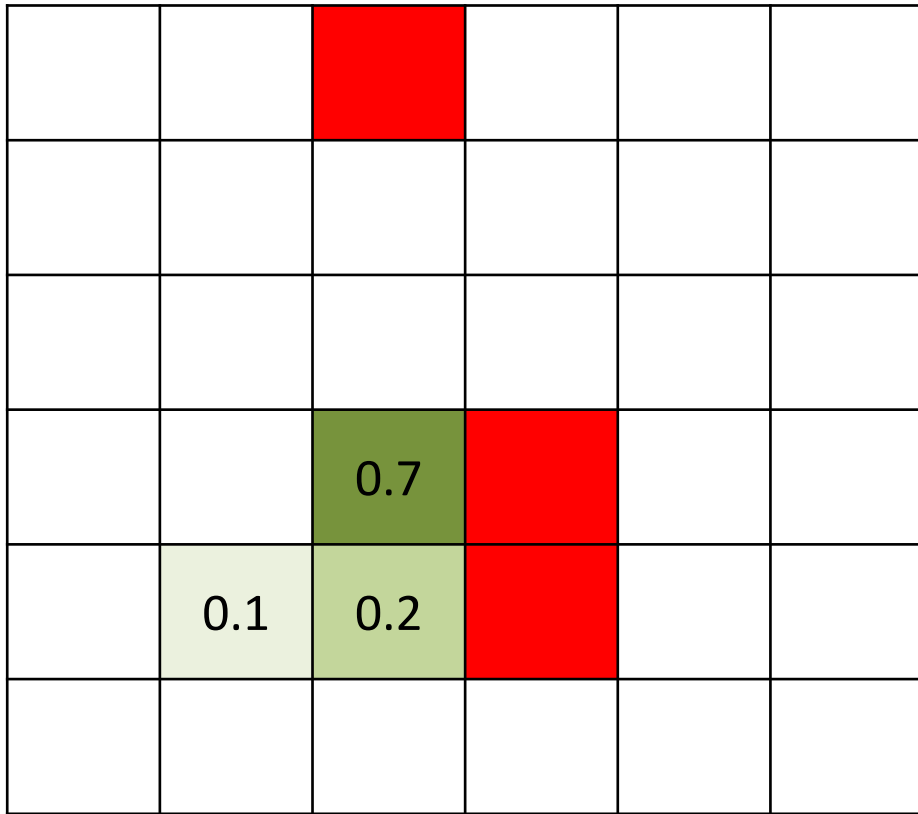
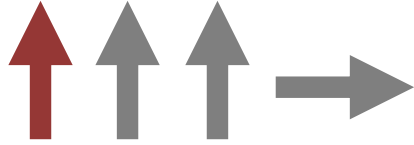


		0.0343			
	0.0147	0.049	0.2548		
0.0021	0.0189	0.1582	0.1218	0.1050	
0.0025	0.0342	0.0511	0.0636	0.0318	0.0148
0.0025	0.0076	0.0112	0.0097	0.0066	0.0029
0.0004	0.0006	0.0007	0.0006	0.0003	0.0001

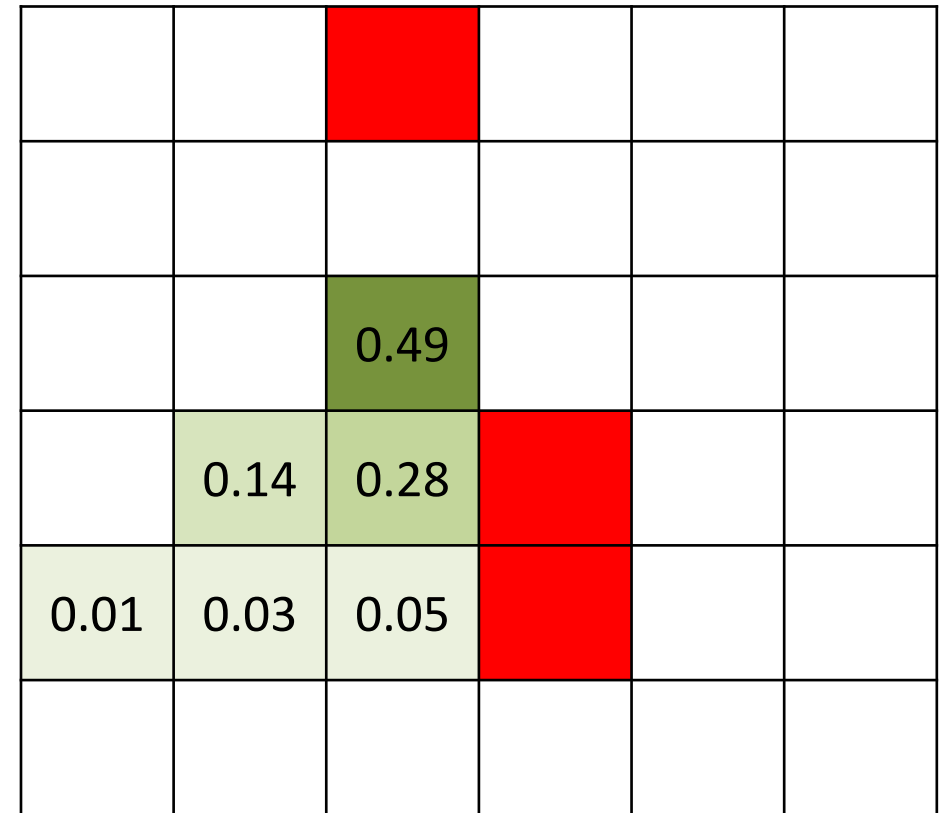
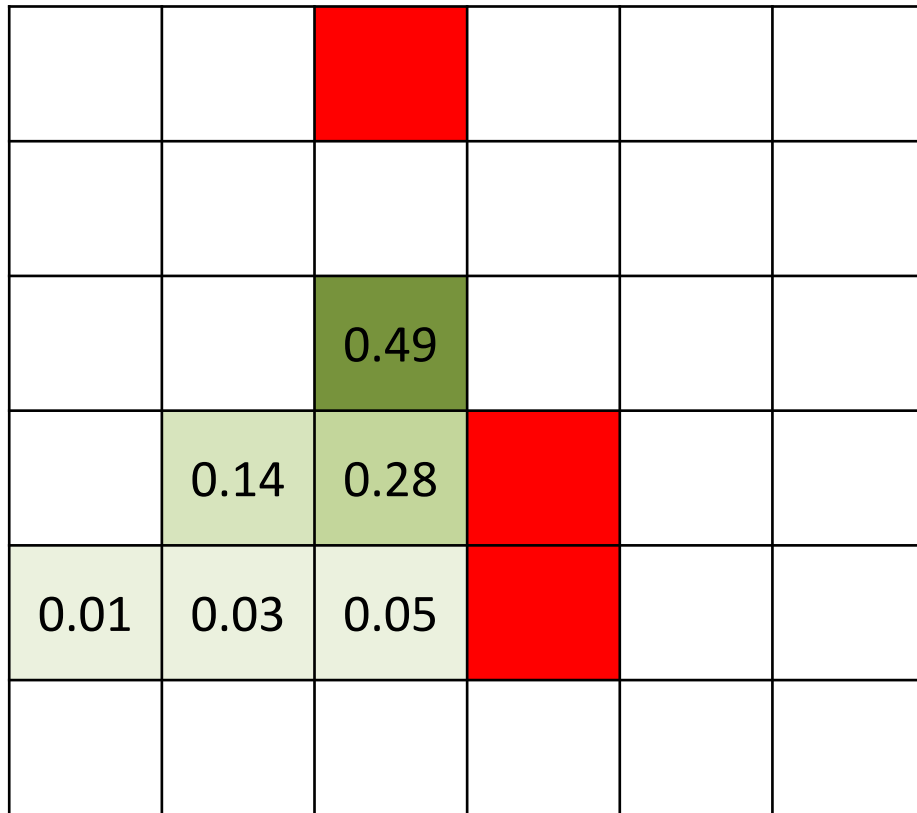
Let's put some obstacles in this space



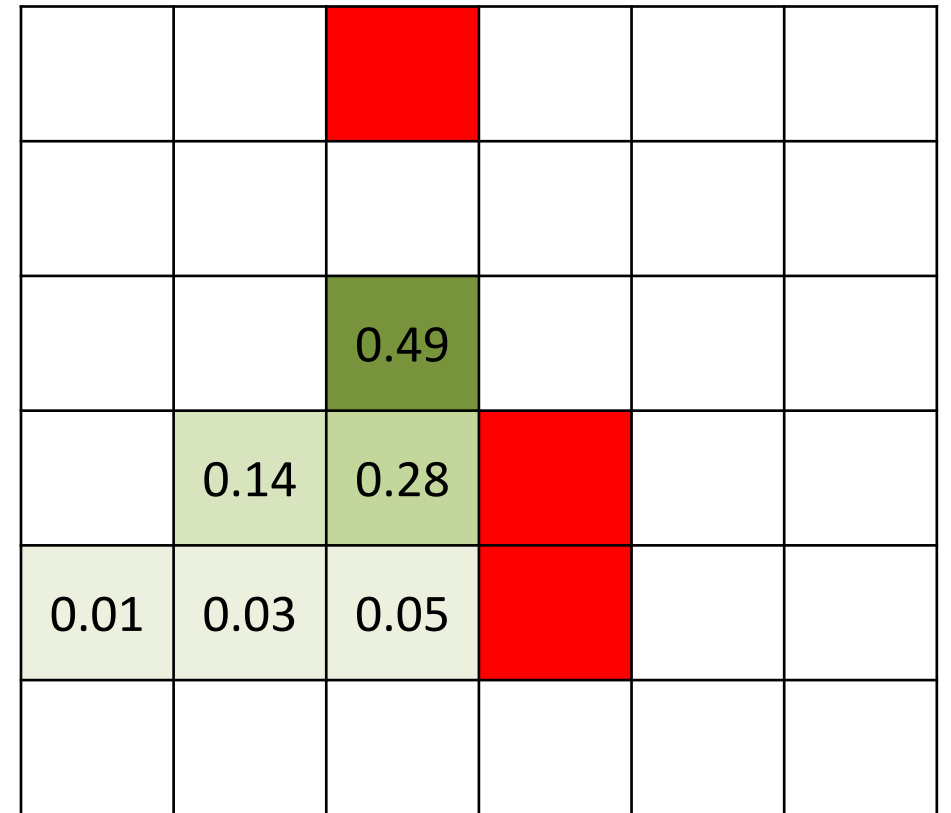
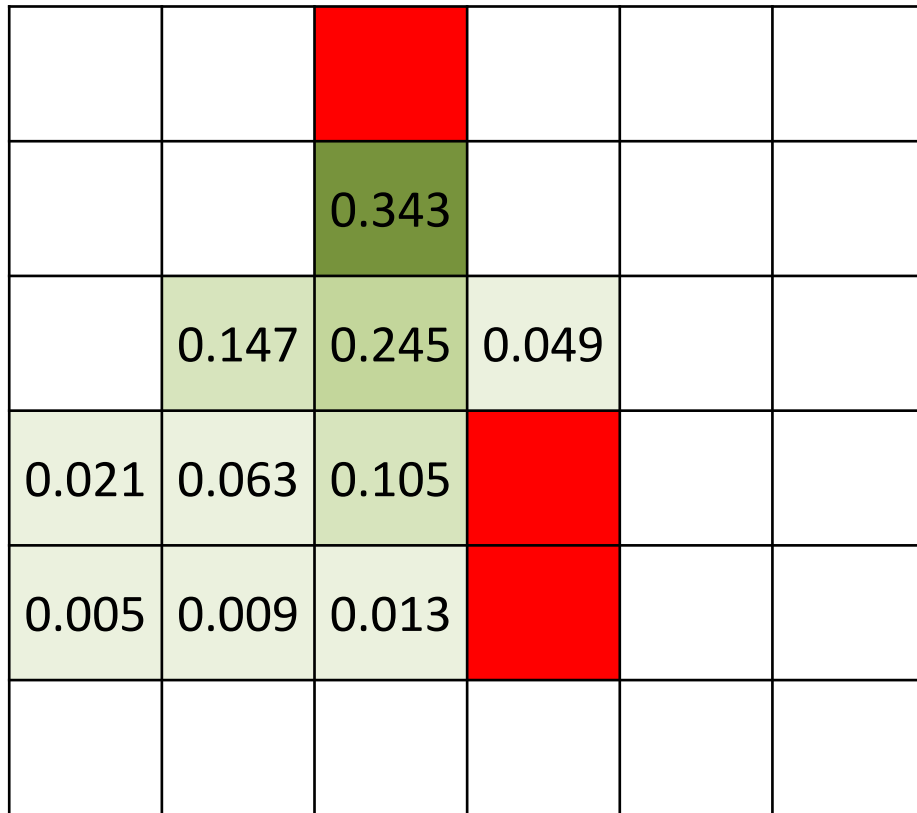
Let's put some obstacles in this space



Let's put some obstacles in this space



Let's put some obstacles in this space



Let's put some obstacles in this space



		0.343			
	0.147	0.245	0.049		
0.021	0.063	0.105			
0.005	0.009	0.013			



		0.049			
	0.014	0.077	0.343		
0.001	0.017	0.376			
0.001	0.024	0.089			
0.001	0.003	0.005			

Let's put some obstacles in this space



	0.0147	0.0931	0.245		
0.0021	0.021	0.1722	0.1813	0.0343	
0.0026	0.0366	0.1539			
0.0026	0.0107	0.0272			
0.0005	0.0009	0.0013			



	0.0147	0.0931	0.245		
0.0021	0.021	0.3066	0.042	0.0343	
0.0026	0.0562	0.1392			
0.0033	0.0135	0.0237			
0.0005	0.0009	0.0008	0.0005		

The distribution of robot states has changed!!

MDPs

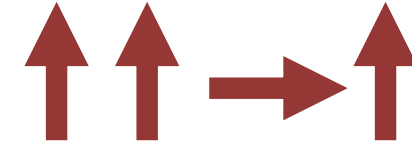
Markov Decision Processes are based on the idea that different sequences of actions change the probability distribution for your robot's state

Goal: Find a sequence of actions that maximizes the probability that you end up where you want to go

Let's put some obstacles in this space



	0.0147	0.0931	0.245		
0.0021	0.021	0.1722	0.1813	0.0343	
0.0026	0.0366	0.1539			
0.0026	0.0107	0.0272			
0.0005	0.0009	0.0013			

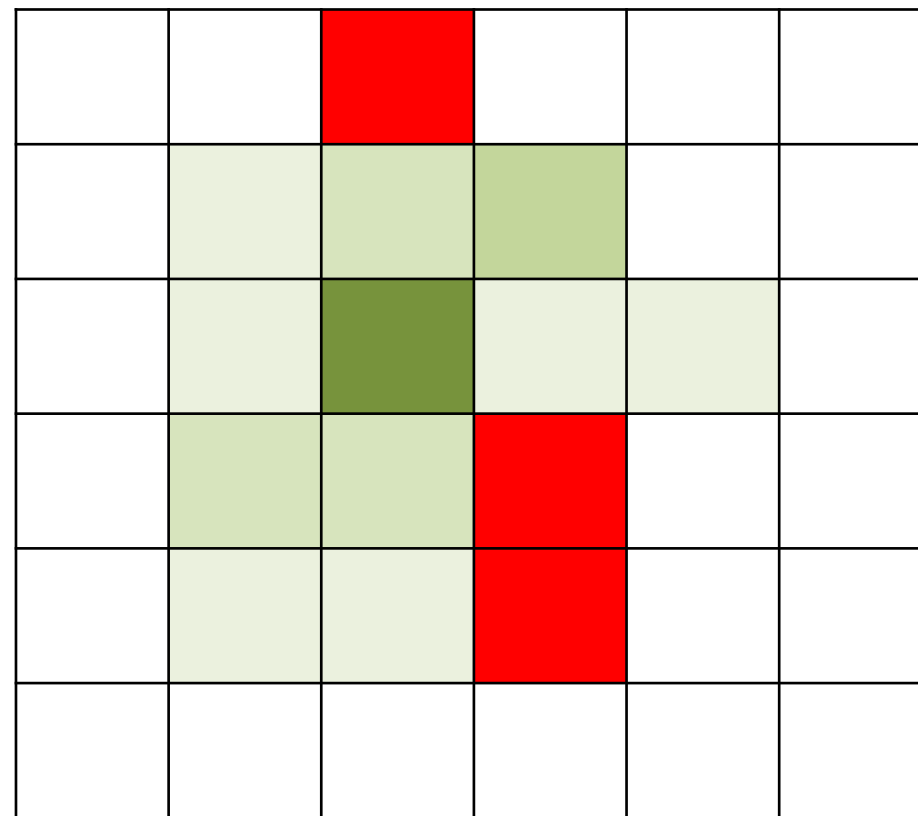
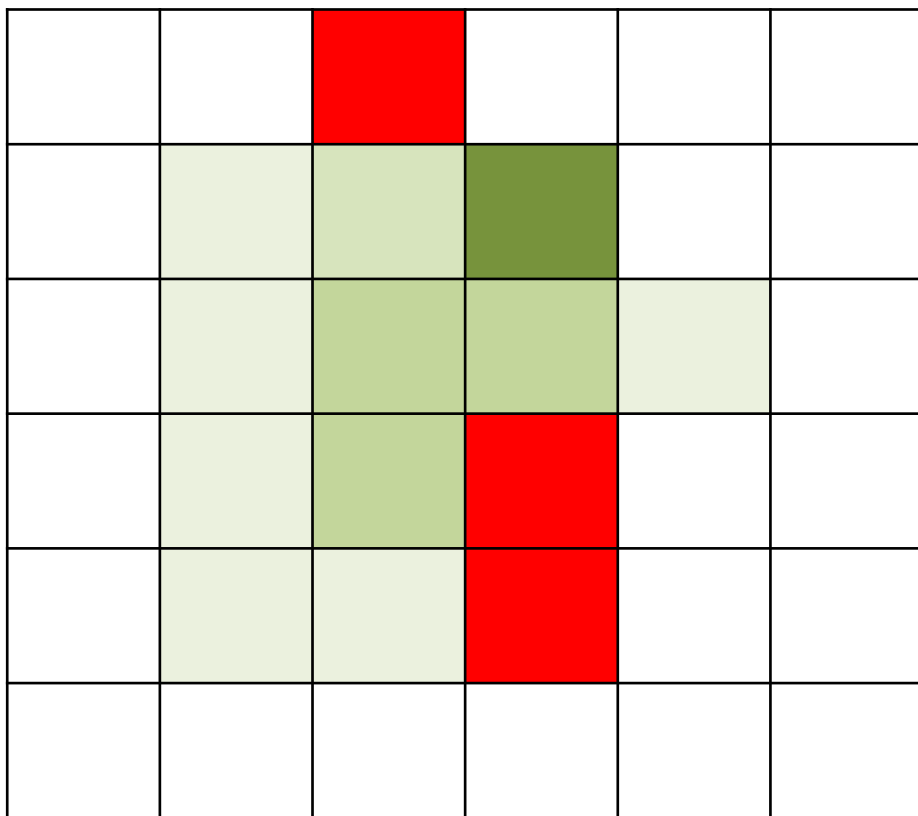


What if we want to end up here?

	0.0147	0.0931	0.245		
0.0021	0.021	0.3066	0.042	0.0343	
0.0026	0.0562	0.1392			
0.0033	0.0135	0.0237			
0.0005	0.0009	0.0008	0.0005		

For this sequence, most likely to end up here

Let's assign a reward



Let's assign a reward

$$\text{Total reward} = \sum_{\text{states } s_i} P(s_i) R(s_i)$$



1	1		1	1	1
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1		1	1
1	1	1		1	1
1	1	1	1	1	1

Total reward = 1.0



1	1		1	1	1
1	1	1	1	1	1
1	1	1	1	1	1
1	1	1		1	1
1	1	1		1	1
1	1	1	1	1	1

Total reward = 1.0

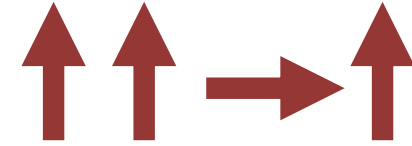
Let's assign a reward

$$\text{Total reward} = \sum_{\text{states } s_i} P(s_i) R(s_i)$$



1	0		0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	0		0	1
1	1	0		0	1
1	1	1	0	1	1

Total reward = 0.5445



1	0		0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	0		0	1
1	1	0		0	1
1	1	1	0	1	1

Total reward = 0.7015

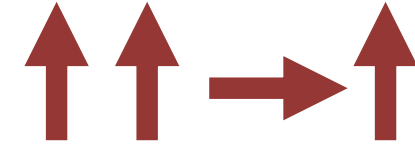
Let's assign a reward

$$\text{Total reward} = \sum_{\text{states } s_i} P(s_i) R(s_i)$$



..	0.13		0.5	0.25	0.13
0.13	0.25	0.5	1	0.5	0.25
..	0.13	0.25	0.5	0.25	0.13
	..	0.13		0.13	..
		

Total reward = 0.4640



..	0.13		0.5	0.25	0.13
0.13	0.25	0.5	1	0.5	0.25
..	0.13	0.25	0.5	0.25	0.13
	..	0.13		0.13	..
		

Total reward = 0.4272

Rewards

We don't usually assign rewards just for an end state.

Instead, accumulate rewards through an entire trajectory / action sequence

You can also assign rewards to actions instead of just states

Goal: Find an action sequence that maximizes accumulated reward

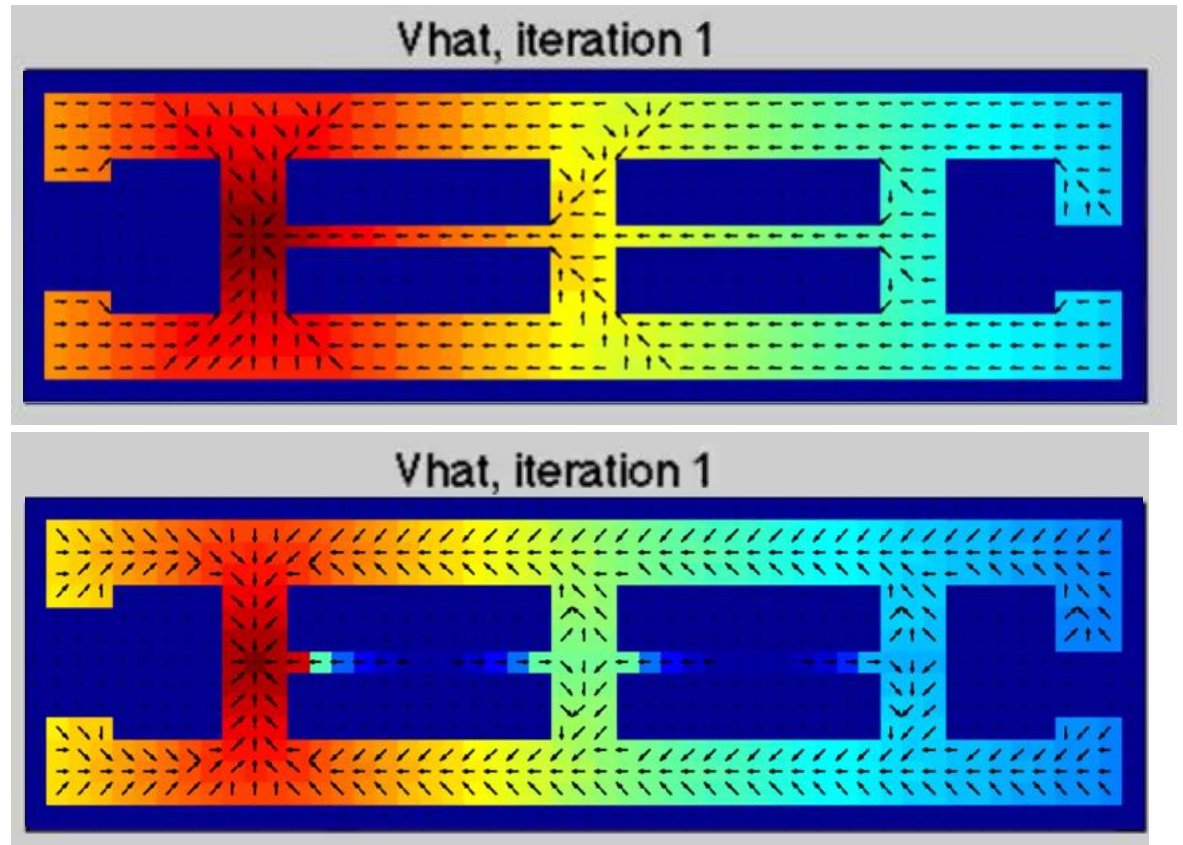
Policies

Usually, we use MDPs to find “policies”

For every state, find the action that maximizes eventual reward



For more about how to solve for policies, take machine learning or do a final project!



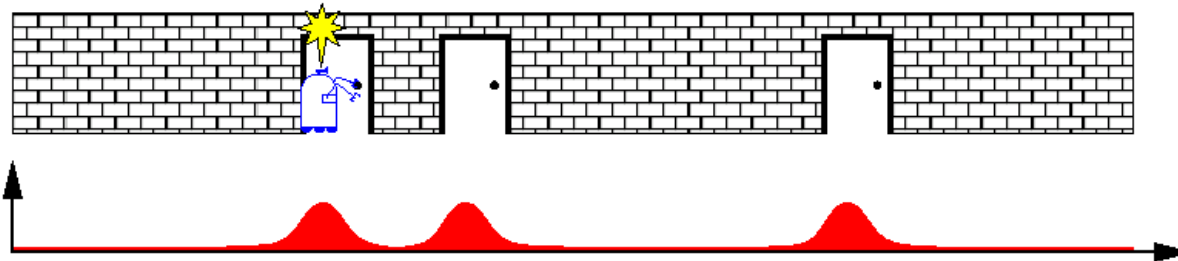
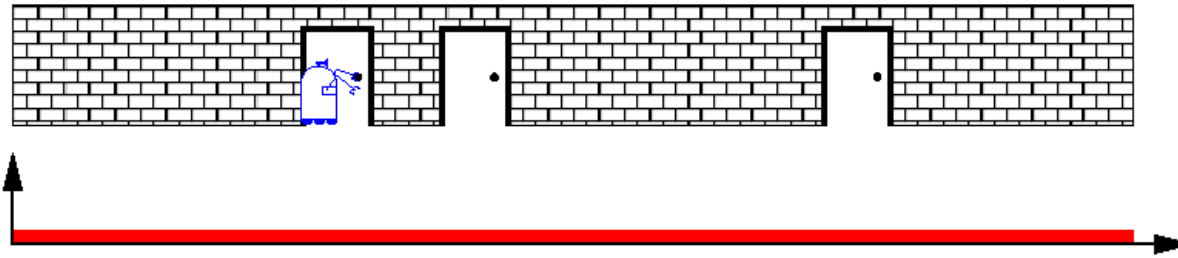
This paper: Planning with POMDPs

A POMDP is a **partially observable** Markov decision process

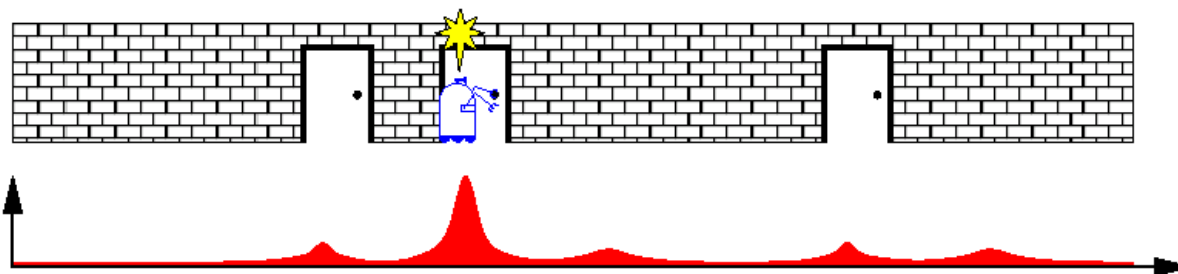
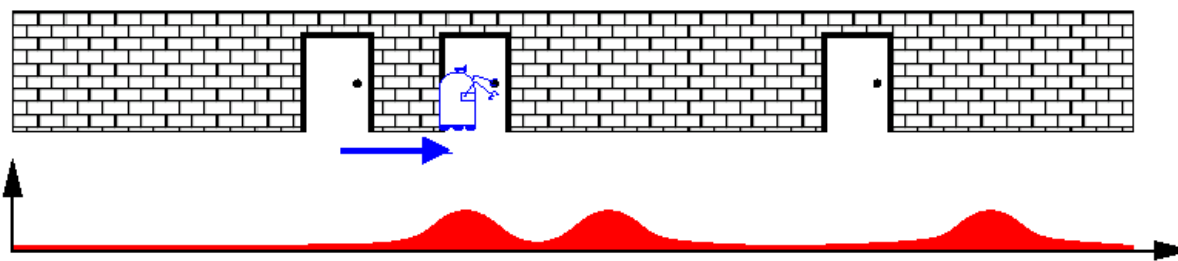
Ok, so what about this part?

An MDP assumes that the state of the robot is known.
Otherwise, it will not be able to choose the correct action.

Robots use sensors, which have noise.
They can only **partially observe** their state.

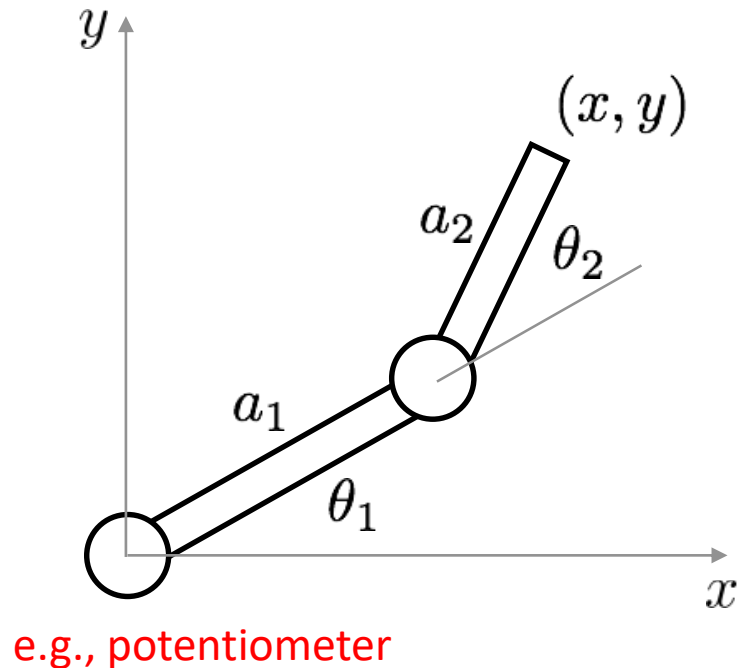


Let's say the robot's door sensor is really really good. How does this figure change?

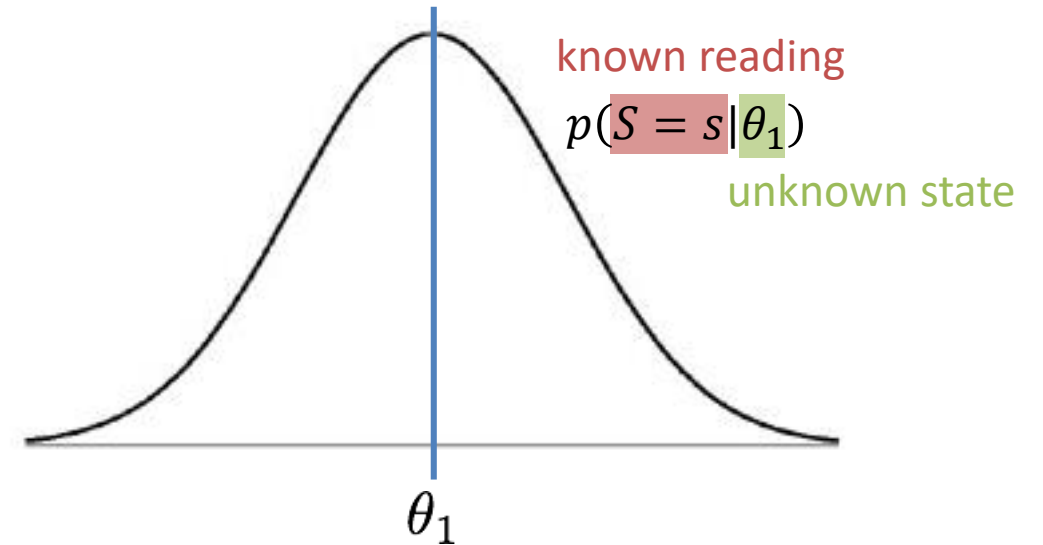


State Estimation

State estimation is the problem of estimating quantities from sensor data



The sensor has some noise distribution

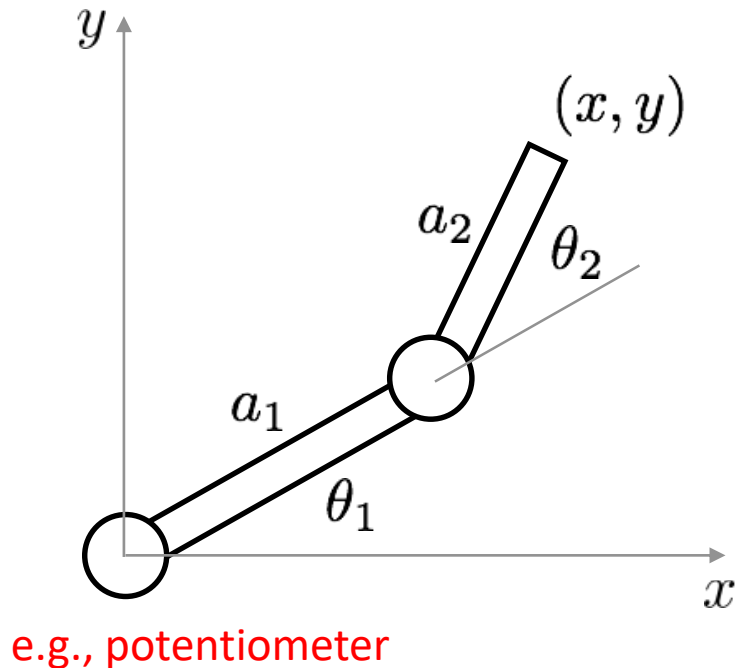


We want to find $p(\theta_1)$

This is called the robot's "belief"

The sensing info s is the robot's "observation"

Bayes Rule



$$p(\theta_1 | S = s) = \frac{p(S = s | \theta_1) p(\theta_1)}{p(S = s)}$$

$p(S = s \text{ AND } \theta_1)$

independent of θ_1 ,
depends only on
environment/robot

Probabilities must sum to **1**

$$p(\theta_1 | S = s) = \eta p(S = s | \theta_1) p(\theta_1)$$

where the **normalizer** $\eta = (\sum_{\theta_1} p(\theta_1 | S = s))^{-1}$

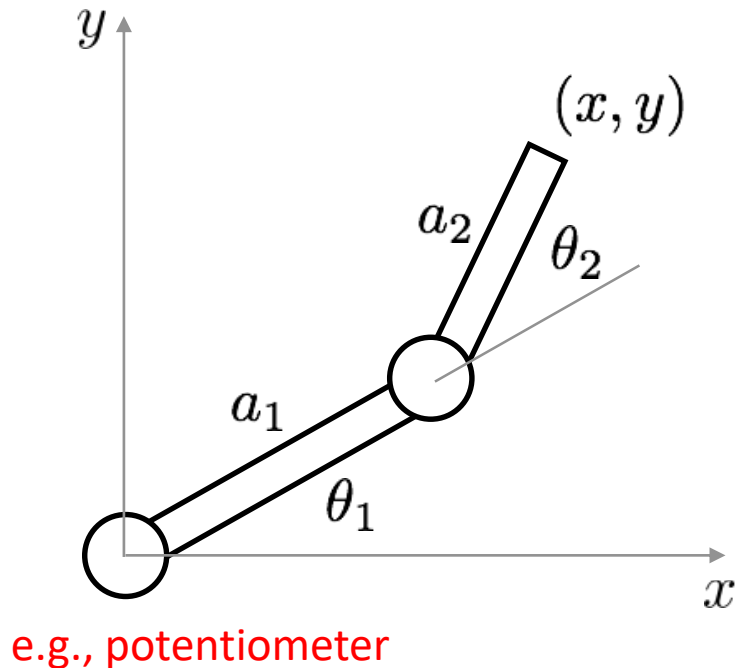
Repeated Measurements

Let's say we have no idea of the starting config:

$$p(\theta_1) \sim U[\theta_{min}, \theta_{max}]$$

Take one sensor reading:

$$p(\theta_1) = \eta p(S = s_1 | \theta_1) p(\theta_1)$$



Take another sensor reading:

$$\eta = \left(\sum_{\theta_1} p(S = s_2 | \theta_1) p(\theta_1) \right)^{-1}$$

$$p(\theta_1) = \eta p(S = s_2 | \theta_1) p(\theta_1)$$

Often, the robot is moving at the same time

Let's say we have no idea of the starting config:

$$p(\theta_1^0) \sim U[\theta_{min}, \theta_{max}]$$

Take one sensor reading:

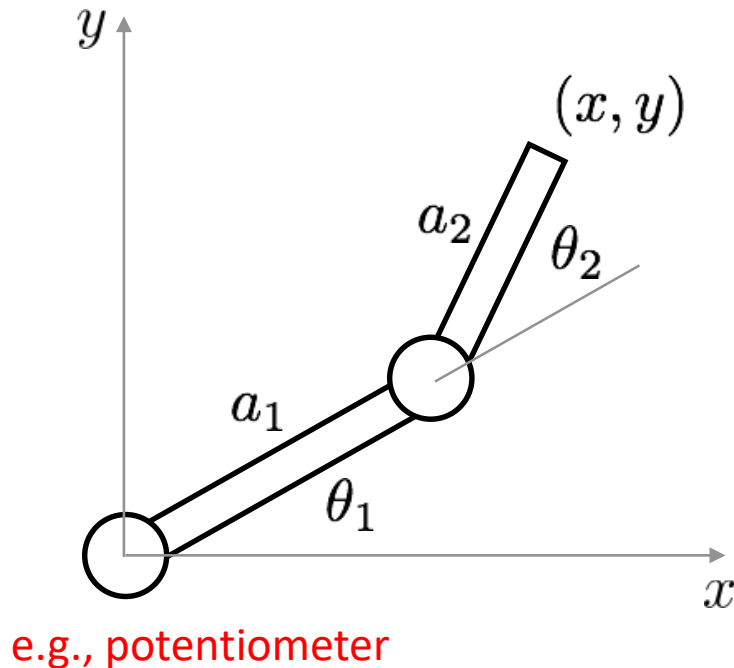
$$p(\theta_1^1) = \eta p(S = s_1 | \theta_1^0) p(\theta_1^0)$$

Move the robot:

$$p(\theta_1^2) = \sum_{\theta_1^1} p(\theta_1^2 | u_1, \theta_1^1) p(\theta_1^1)$$

Take another sensor reading:

$$\eta = \left(\sum_{\theta_1} p(S = s_2 | \theta_1^2) p(\theta_1^2) \right)^{-1}$$
$$p(\theta_1^2) = \eta p(S = s_2 | \theta_1^2) p(\theta_1^2)$$



Bayes Filter

Pseudocode:

For all q_t configuration at time t

new belief w/o sensing control input belief at time t-1

$$\overline{bel}(q_t) = \int p(q_t | u_t, q_{t-1}) bel(q_{t-1}) dt$$

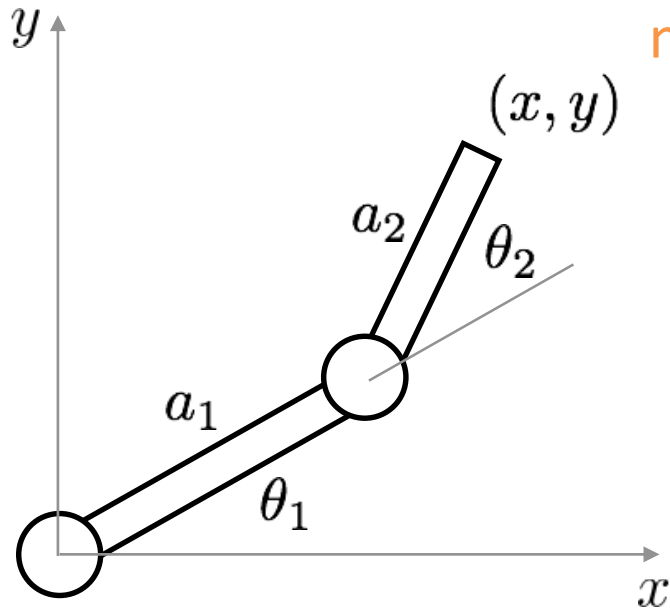
configuration at time t-1

$$bel(q_t) = \eta p(s_t | q_t) \overline{bel}(q_t)$$

normalize sensing info

end

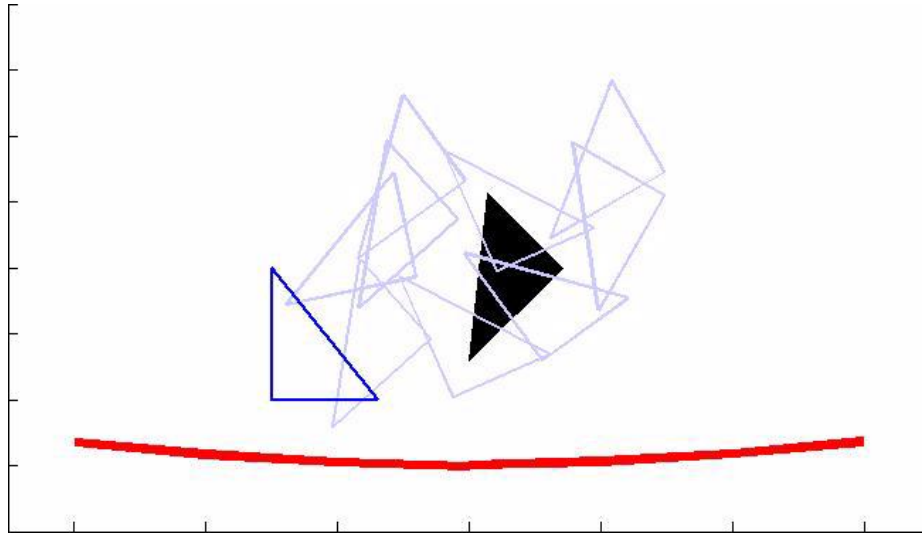
return $bel(q_t)$
"belief"



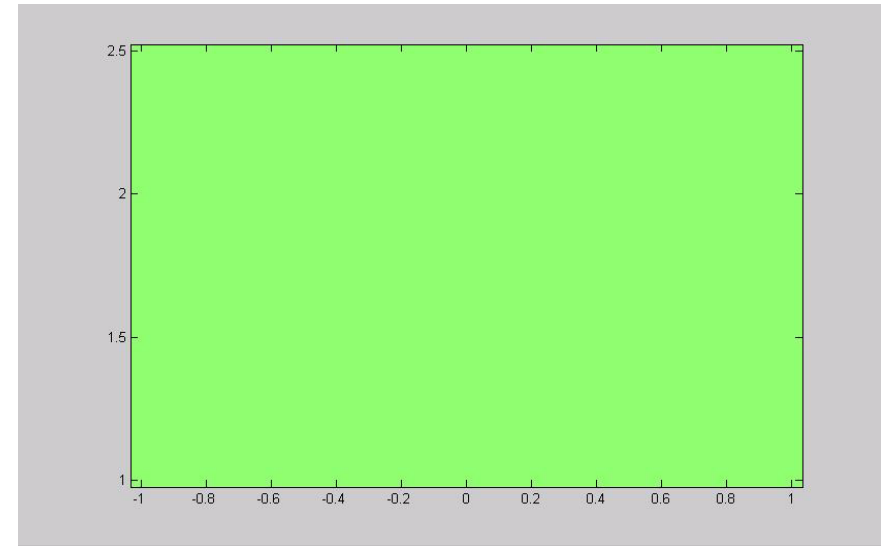
e.g., potentiometer

$$\Pr(s_{t+1} = i | o_1 \dots o_{t+1}, a_1 \dots a_t) = \sum_j b_j \Pr(s_{t+1} = i | s_t = j, a_t = a) \cdot \Pr(o_{t+1} = o | s_{t+1} = i)$$

Object localization



$$\theta = \frac{\pi}{4}$$



What is the result of the policy search?

Goal: end up in 1A

Initial conditions: Start in 2A, 2C, or 2E

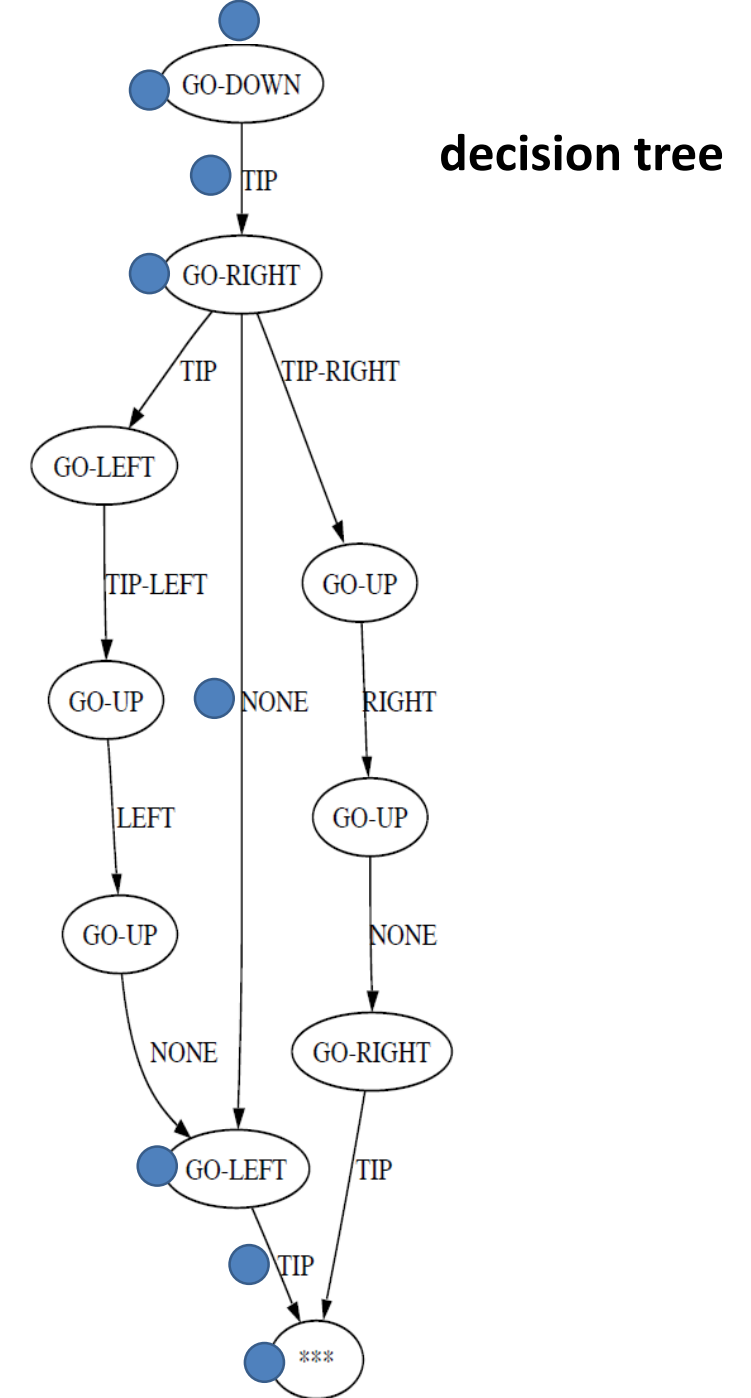
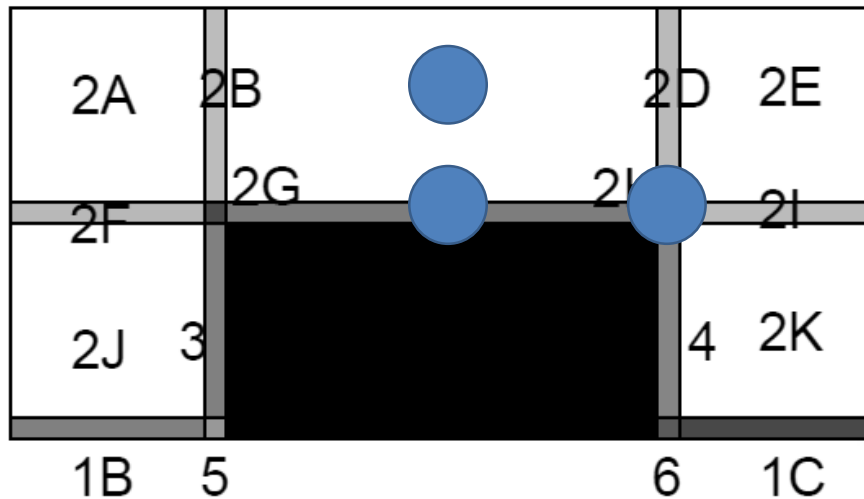


Figure 1

What is the result of the policy search?

Goal: end up in 1A

Initial conditions: Start in 2A, 2C, or 2E

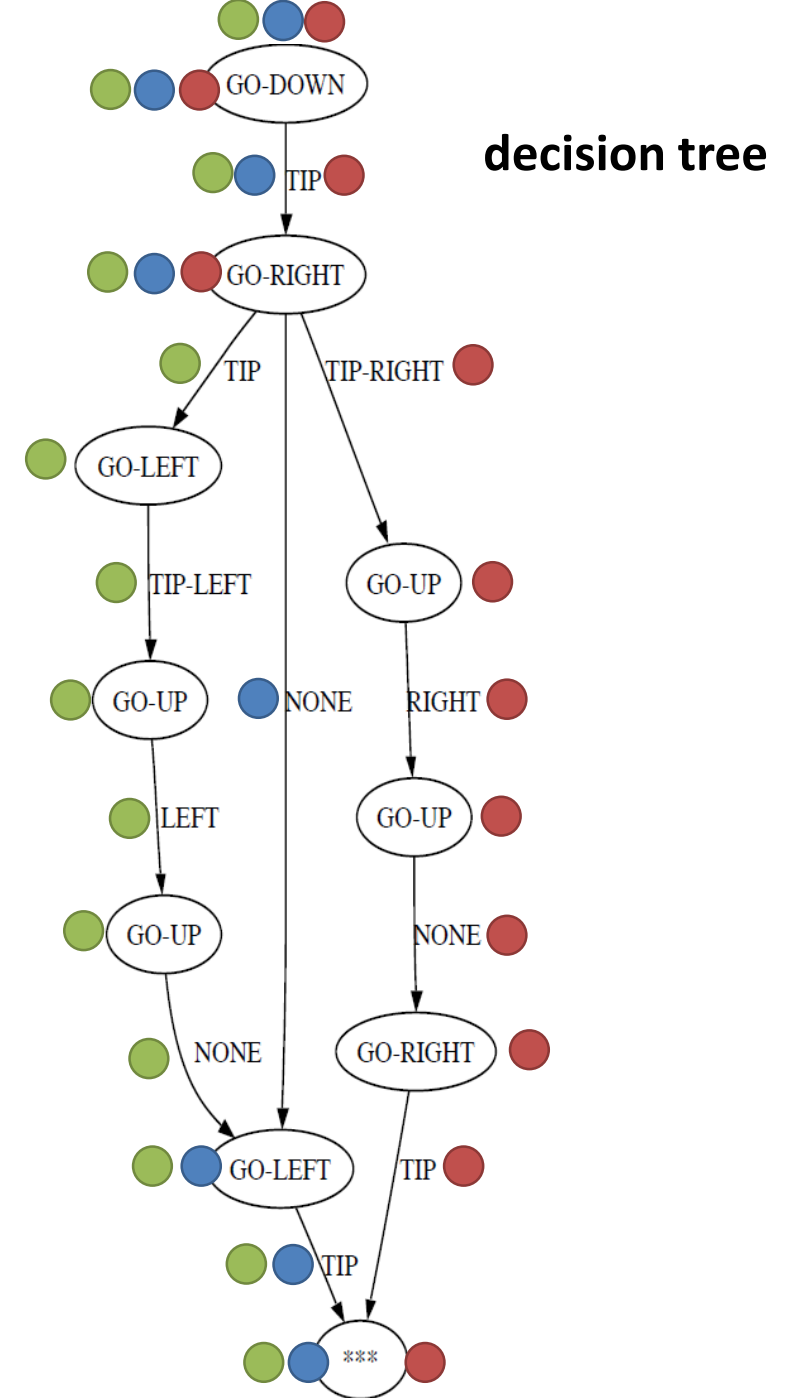
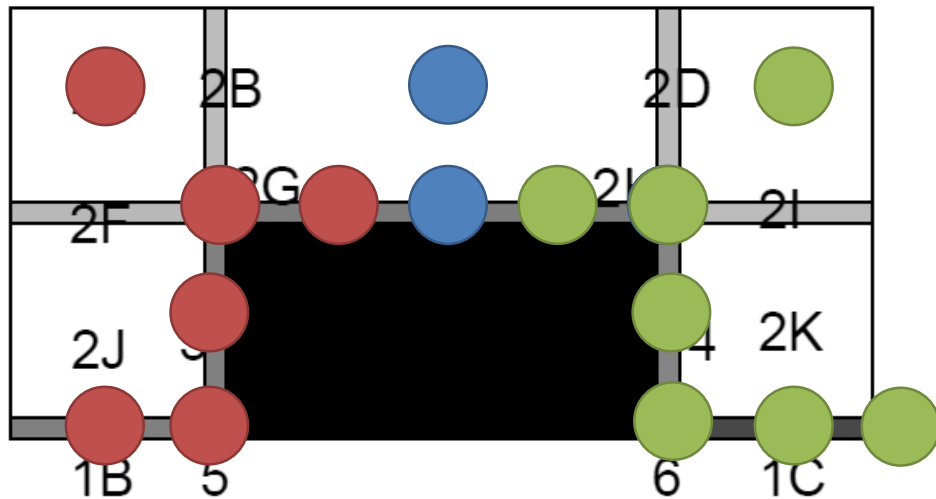


Figure 1

POMDPs in Grasping



POMDP-lite for robust robot planning under uncertainty. Min Chen, Emilio Frazzoli, Davis Hsu, Wee Sun Lee. ICRA 2016

POMDPs in Grasping

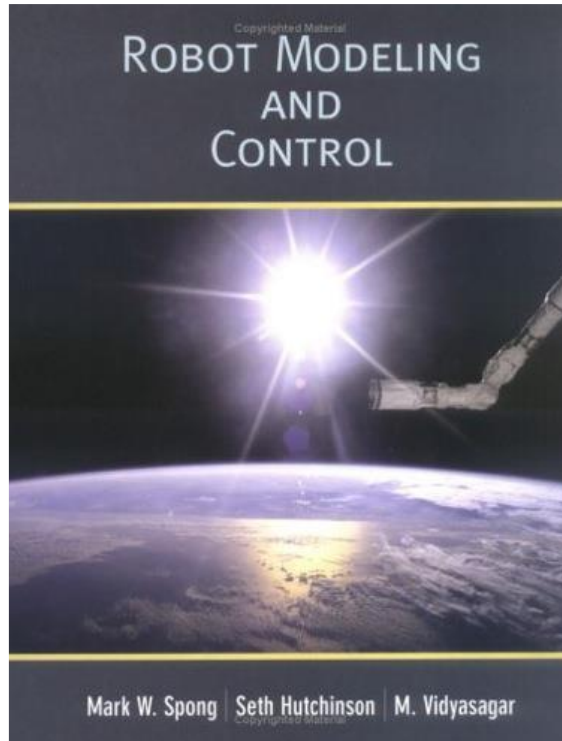


POMDP-lite for robust robot planning under uncertainty. Min Chen, Emilio Frazzoli, Davis Hsu, Wee Sun Lee. ICRA 2016

<https://youtu.be/EAJRH5Qp1w>

That brings us to the end of position/orientation analysis

Next time: Velocity



Chapter 4: Velocity Kinematics

- Read 4.intro – 4.4

Lab 3: Trajectory Planning

MEAM 520, University of Pennsylvania
October 3, 2018

This lab consists of two portions, with a pre-lab due on Wednesday, October 10, by midnight (11:59 p.m.) and a lab report due on Wednesday, October 17, by midnight (11:59 p.m.). Late submissions will be accepted until midnight on Saturday following the deadline, but they will be penalized by 25% for each partial or full day late. After the late deadline, no further assignments may be submitted; post a private message on Piazza to request an extension if you need one due to a special situation.

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. To help you actually learn the material, what you submit must be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. When you get stuck, post a question on Piazza or go to office hours!

Individual vs. Pair Programming

If you choose to work on the lab in a pair, work closely with your partner throughout the lab, following these guidelines, which were adapted from "All I really needed to know about pair programming I learned in kindergarten," by Williams and Kessler, *Communications of the ACM*, May 2000. This article is available on Canvas under Files / Resources.

- Start with a good attitude, setting aside any skepticism, and expect to jell with your partner.
- Don't start alone. Arrange a meeting with your partner as soon as you can.
- Use just one setup, and sit side by side. For a programming component, a desktop computer with a large monitor is better than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (writing, using the mouse/keyboard, moving the robot) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every 30 minutes, even if one partner is much more experienced than the other. You may want to set a timer to help you remember to switch.
- If you notice an error in the equation or code that your partner is writing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.
- Recognize that working in pairs usually takes more time than working alone, but it produces better work, deeper learning, and a more positive experience for the participants.

1

Lab 3: Trajectory Planner due 10/17