Statistical Filtering and Control for AI and Robotics

Planning and Control: Markov Decision Processes

Alessandro Farinelli

UNIVERSITÀ di VERONA
Dipartimento di INFORMATICA

Outline

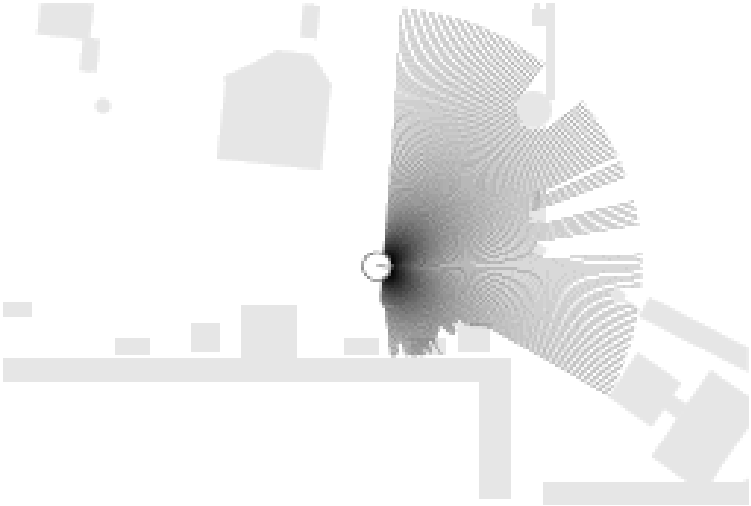
- Uncertainty: localization for mobile robots
  - State estimation based on Bayesian filters [recall]
- Acting Under Uncertainty
  - Markov Decision Problem
  - Solution approaches
- Motion planning
  - Markov Decision Processes for path planning

- Acknowledgment: material based on
  - Russel and Norvig; Artificial Intelligence: a Modern Approach
  - Thrun, Burgard, Fox; Probabilistic Robotics

Mobile robots

Sensors

Range finders: sonar (land, underwater), laser range finder, radar (aircraft), tactile sensors, GPS



Imaging sensors: cameras (visual, infrared)

Proprioceptive sensors: shaft decoders (joints, wheels), inertial sensors, force sensors, torque sensors

Uncertainty

open = open a door

Will *open* actually open the door ?

<u>Problems:</u>

- 1) partial observability and noisy sensors
- 2) uncertainty in action outcomes
- 3) immense complexity of modelling and predicting environment

Probability

Probabilistic assertions summarize effects of

- laziness (enumeration of all relevant facts),
- ignorance (lack of relevant facts)

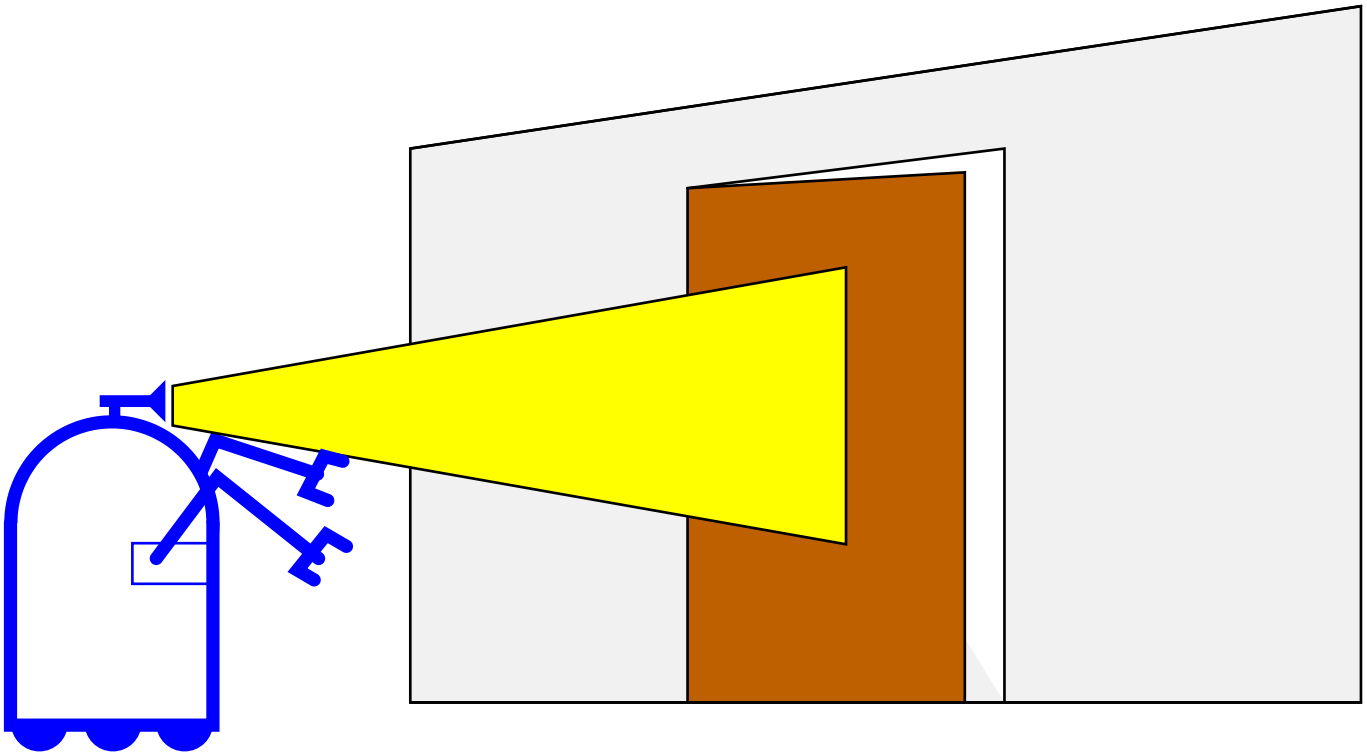
Subjective or Bayesian probability:

- Probabilities relate propositions to one's own state of knowledge
  - P(open|I am in front of the door) = 0.6
  - P(open|I am in front of the door; <u>door is not locked</u>) = 0.8

Simple Example of State Estimation

Suppose a robot obtains measurement z

What is $P(open|z)$?



Causal vs. Diagnostic Reasoning

P(open|z) is diagnostic

P(z|open) is causal

Often causal knowledge is easier to obtain

Bayes rule allows us to use causal knowledge:

$$P(open \mid z) = \frac{P(z \mid open)P(open)}{P(z)}$$

count frequencies!

Example

P(z|open) = 0.6 P(z|¬open) = 0.3

P(open) = P(¬open) = 0.5

$$P(open \mid z) = \frac{P(z \mid open)P(open)}{P(z \mid open)\,p(open) + P(z \mid \neg open)\,p(\neg open)}$$

$$P(open \mid z) = \frac{0.6 \cdot 0.5}{0.6 \cdot 0.5 + 0.3 \cdot 0.5} = \frac{2}{3} = 0.67$$

z raises the probability that the door is open.

Combining Evidence

Suppose our robot obtains another observation z2.

How can we integrate this new information?

More generally, how can we estimate P(x| z1...zn)?

Recursive Bayesian Updating

$$P(x \mid z_1, \ldots, z_n) = \frac{P(z_n \mid x, z_1, \ldots, z_{n-1})\, P(x \mid z_1, \ldots, z_{n-1})}{P(z_n \mid z_1, \ldots, z_{n-1})}$$

Markov assumption: z_n independent of $z_1, \ldots, z_{n-1}$ if we know x

$$P(x \mid z_1, \ldots, z_n) = \frac{P(z_n \mid x)\, P(x \mid z_1, \ldots, z_{n-1})}{P(z_n \mid z_1, \ldots, z_{n-1})}$$

$$= \eta\, P(z_n \mid x)\, P(x \mid z_1, \ldots, z_{n-1})$$

$$= \eta_{1 \ldots n} \prod_{i=1 \ldots n} P(z_i \mid x)\, P(x)$$

Example: Second Measurement

P(z2|open) = 0.5 P(z2|¬open) = 0.6

P(open|z1)=2/3

$$P(open \mid z_2, z_1) = \frac{P(z_2 \mid open)\, P(open \mid z_1)}{P(z_2 \mid open)\, P(open \mid z_1) + P(z_2 \mid \neg open)\, P(\neg open \mid z_1)}$$

$$= \frac{\frac{1}{2} \cdot \frac{2}{3}}{\frac{1}{2} \cdot \frac{2}{3} + \frac{3}{5} \cdot \frac{1}{3}} = \frac{5}{8} = 0.625$$

z_2 lowers the probability that the door is open.

Actions

Often the world is dynamic

- actions carried out by the robot,
- actions carried out by other agents,
- time passing by

How can we incorporate such actions?

Typical Actions

The robot moves

The robot moves objects

People move around the robot

Actions are never carried out with absolute certainty.

In contrast to measurements, actions generally increase the uncertainty.

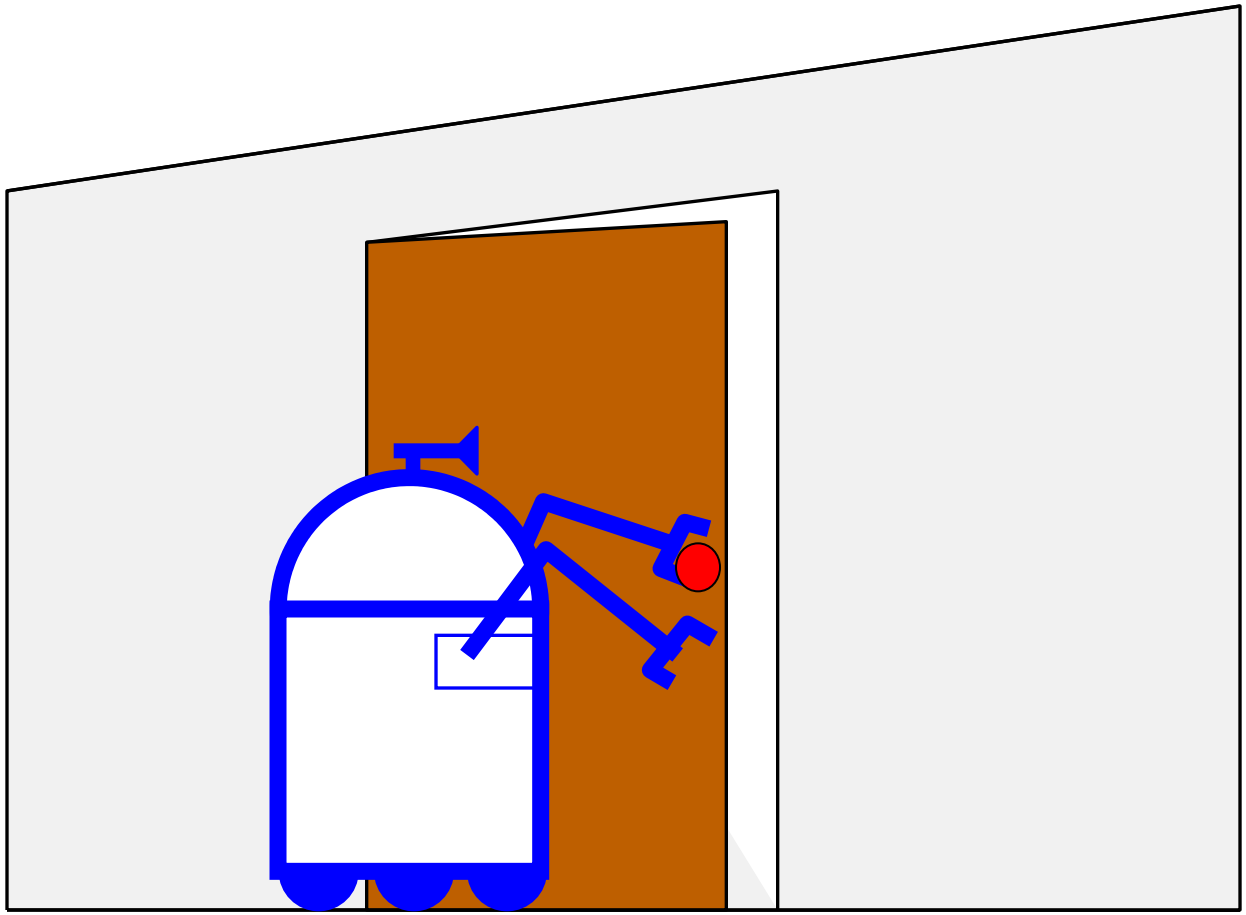
Modeling Actions

To incorporate the outcome of an action u into the current "belief", we use conditional pdf

$$P(x' \mid u,x)$$

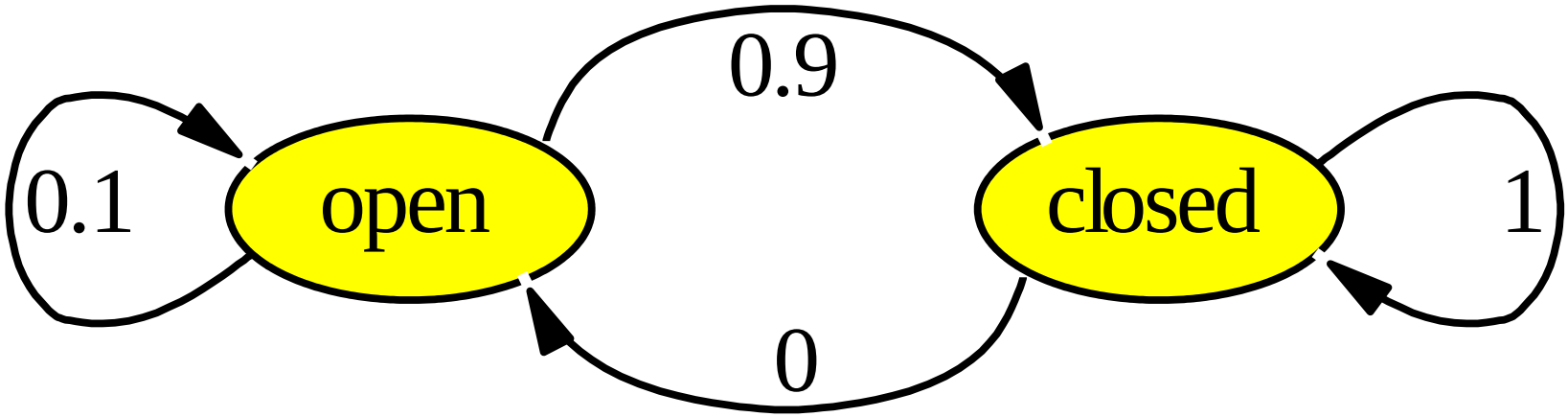
This term specifies the pdf that <u>executing u changes the state from x to x'</u>.

Example: Closing the door



State Transitions

- P(x'|u,x) for u = "close door":



- If the door is open, the action "close door" succeeds in 90% of all cases.

Integrating the Outcome of Actions

Continuous case:

$$P(x'|u) = \int P(x'|u,x)P(x)dx$$

Discrete case:

$$P(x'|u) = \sum P(x'|u,x)P(x)$$

Example: The Resulting Belief

$$
\begin{aligned}
P(closed \mid u) &= \sum P(closed \mid u, x) P(x) \\
&= P(closed \mid u, open) P(open) \\
&\quad + P(closed \mid u, closed) P(closed) \\
&= \frac{9}{10} * \frac{5}{8} + \frac{1}{1} * \frac{3}{8} = \frac{15}{16} \\
P(open \mid u) &= \sum P(open \mid u, x) P(x) \\
&= P(open \mid u, open) P(open) \\
&\quad + P(open \mid u, closed) P(closed) \\
&= \frac{1}{10} * \frac{5}{8} + \frac{0}{1} * \frac{3}{8} = \frac{1}{16} \\
&= 1 - P(closed \mid u)
\end{aligned}
$$

Bayes Filters: Framework

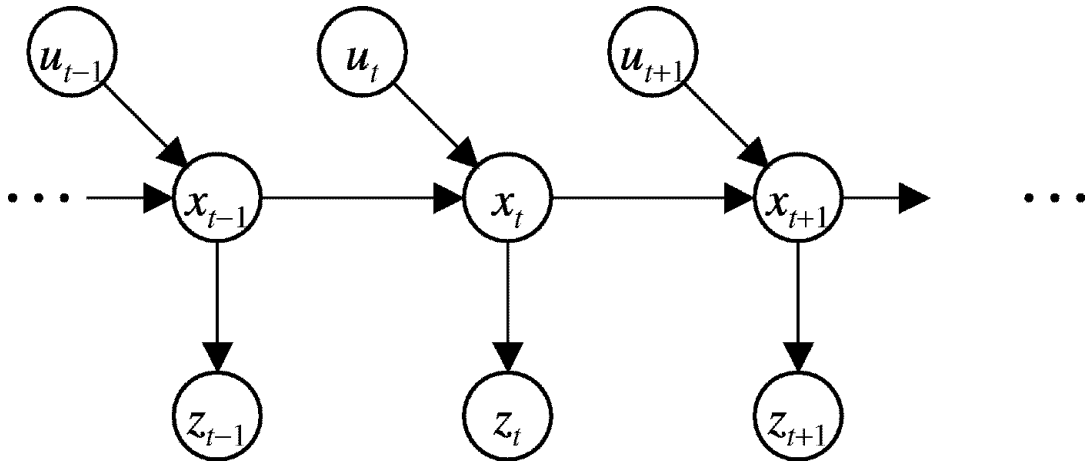
- Given:
  - Stream of observations z and action data u:

$$d_t = \{u_1, z_1 \ldots, u_t, z_t\}$$

  - Sensor model P(z|x)
  - Action model P(x'|u,x)
  - Prior probability of the system state P(x)
- Compute:
  - Estimate of the state X of a dynamical system
  - The posterior of the state is also called Belief:

$$Bel(x_t) = P(x_t \mid u_1, z_1 \ldots, u_t, z_t)$$

Markov Assumption



$$p(z_t \mid x_{0:t}, z_{1:t}, u_{1:t}) = p(z_t \mid x_t)$$

$$p(x_t \mid x_{1:t-1}, z_{1:t}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t)$$

Underlying Assumptions

- Static world (no one else changes the world)
- Independent noise (over time)
- Perfect model, no approximation errors

Bayes Filters

z = observation
u = action
x = state

$$\boxed{Bel(x_t)} = P(x_t \mid u_1, z_1 \ldots, u_t, z_t)$$

Bayes
$$= \eta \; P(z_t \mid x_t, u_1, z_1, \ldots, u_t) \, P(x_t \mid u_1, z_1, \ldots, u_t)$$

Markov
$$= \eta \; P(z_t \mid x_t) \, P(x_t \mid u_1, z_1, \ldots, u_t)$$

Total prob.
$$= \eta \; P(z_t \mid x_t) \int P(x_t \mid u_1, z_1, \ldots, u_t, x_{t-1}) \, P(x_{t-1} \mid u_1, z_1, \ldots, u_t) \, dx_{t-1}$$

Markov
$$= \eta \; P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \, P(x_{t-1} \mid u_1, z_1, \ldots, u_t) \, dx_{t-1}$$

Markov
$$= \eta P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \, P(x_{t-1} \mid u_1, z_1, \ldots, z_{t-1}) \, dx_{t-1}$$

$$\boxed{= \eta \; P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \, Bel(x_{t-1}) \, dx_{t-1}}$$

Bayes Filter Algorithm

1. Algorithm **Bayes_filter**(*Bel(x),d*):
2. η=0
3. If *d* is a *perceptual* data item *z* then
4. For all *x* do
5. $$Bel'(x) = P(z \mid x) Bel(x)$$
6. $$\eta = \eta + Bel'(x)$$
7. For all *x* do
8. $$Bel'(x) = \eta^{-1} Bel'(x)$$
9. Else if *d* is an *action* data item *u* then
10. For all *x'* do
11. $$Bel'(x') = \int P(x' \mid u, x)\, Bel(x)\, dx$$
12. Return *Bel'(x)*

$$Bel(x_t) = \eta\ P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1})\, Bel(x_{t-1})\, dx_{t-1}$$

Bayes Filters are Familiar!

$$Bel(x_t) = \eta \; P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \, Bel(x_{t-1}) \, dx_{t-1}$$

Kalman filters

Particle filters

Hidden Markov models

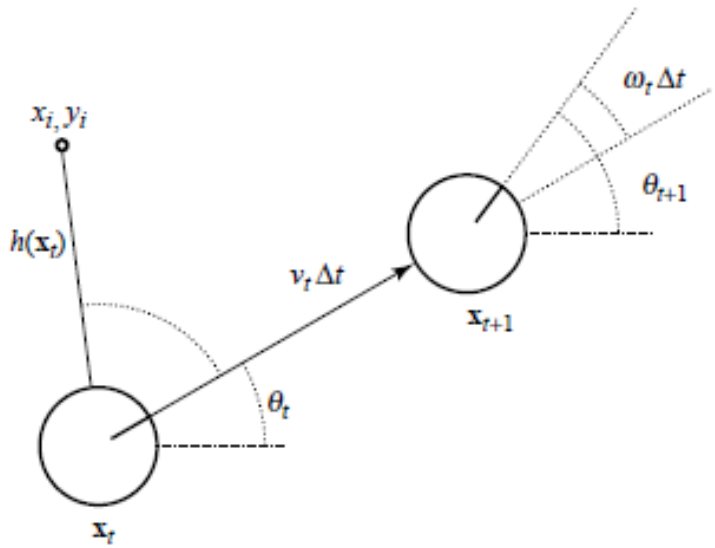
Dynamic Bayesian networks

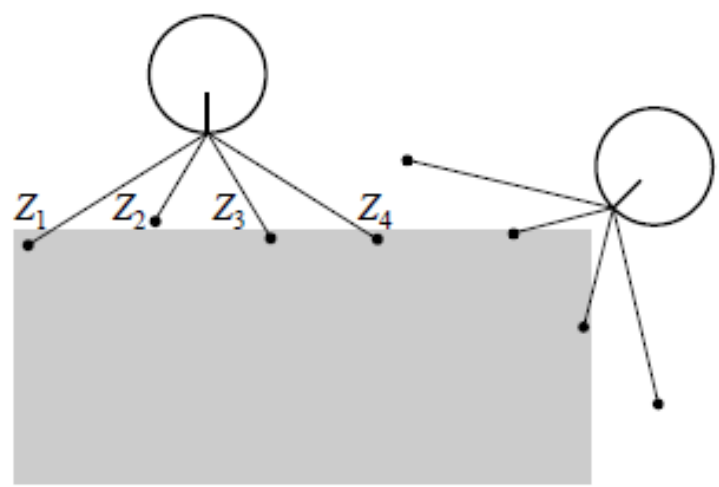
Partially Observable Markov Decision Processes (POMDPs)

Bayesian filters for localization

How do I know whether I am in front of the door ?

Localization as a <u>state estimation process (filtering)</u>

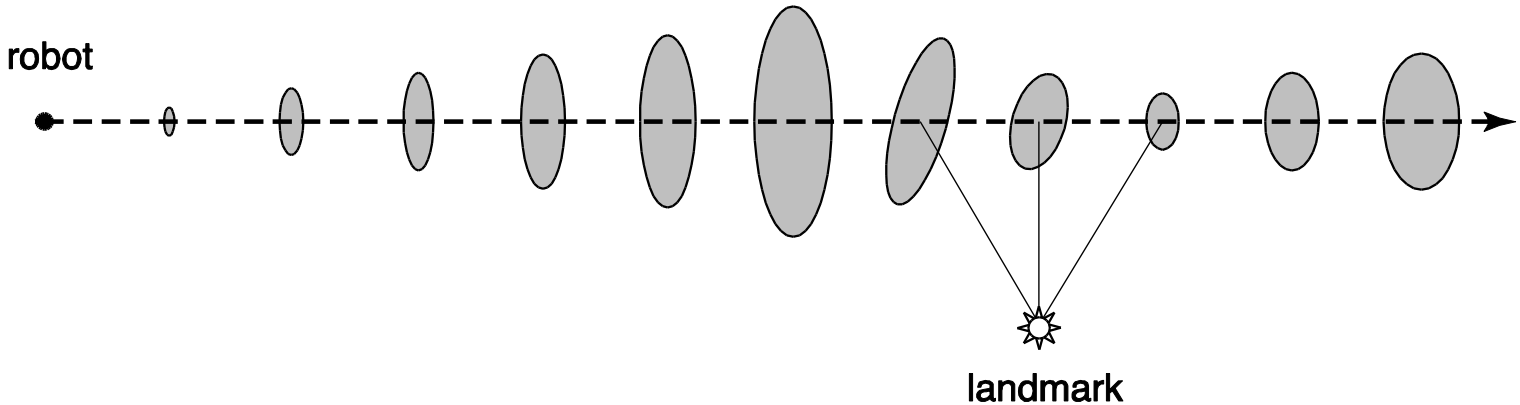




State update

Sensor Reading

Kalman Filter for Localization



Gaussian pdf for belief

- Pros: closed form representation, very fast update
- Cons:

  Works only for linear action and sensor models (can use EKF to overcome this)

  Works well only for unimodal beliefs

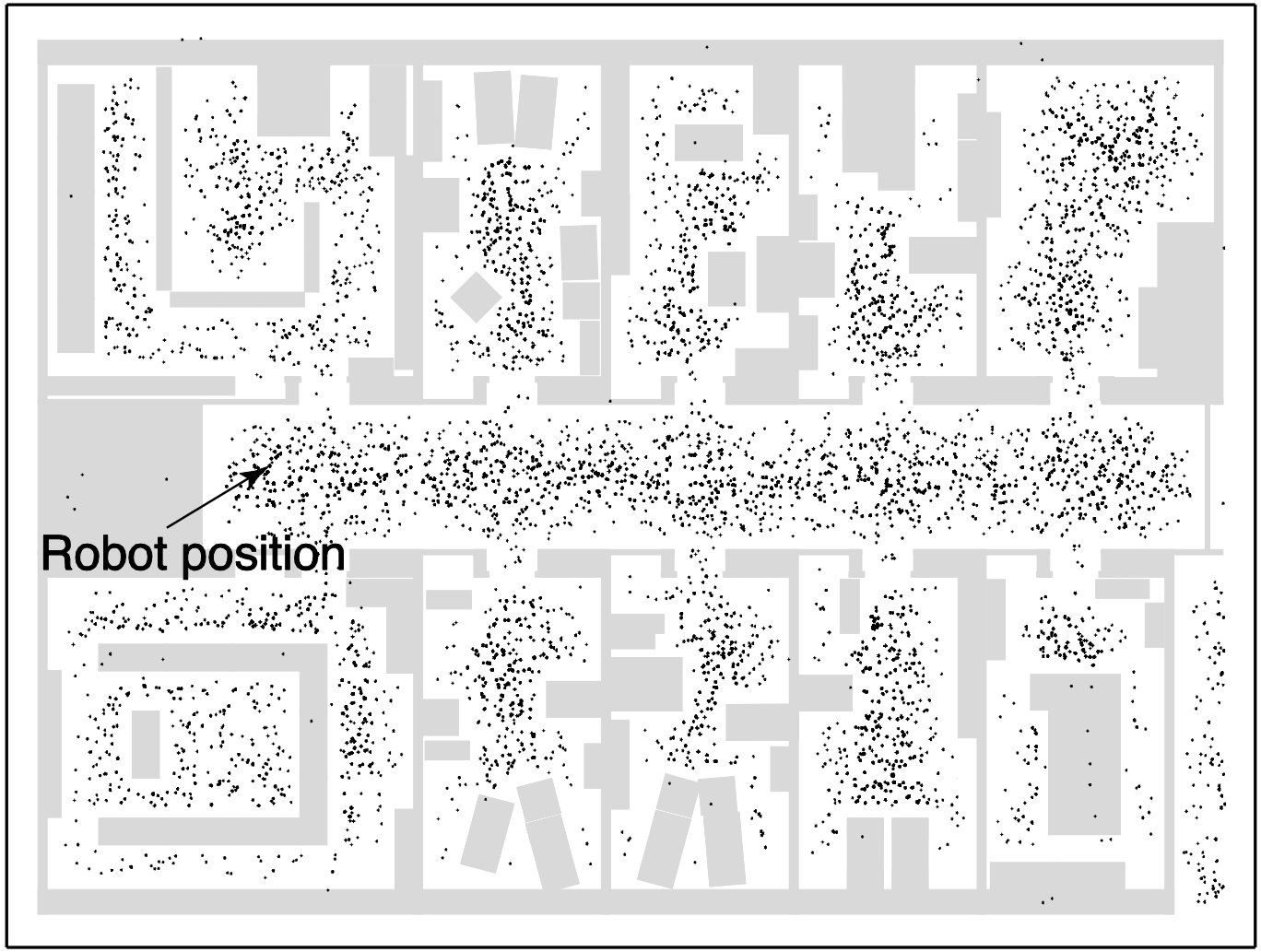
Particle filters

Particles to represent the belief

<u>Pros</u>: no assumption on belief, action and sensor models

<u>Cons</u>: update can be computationally demanding

Particle Filters: prior

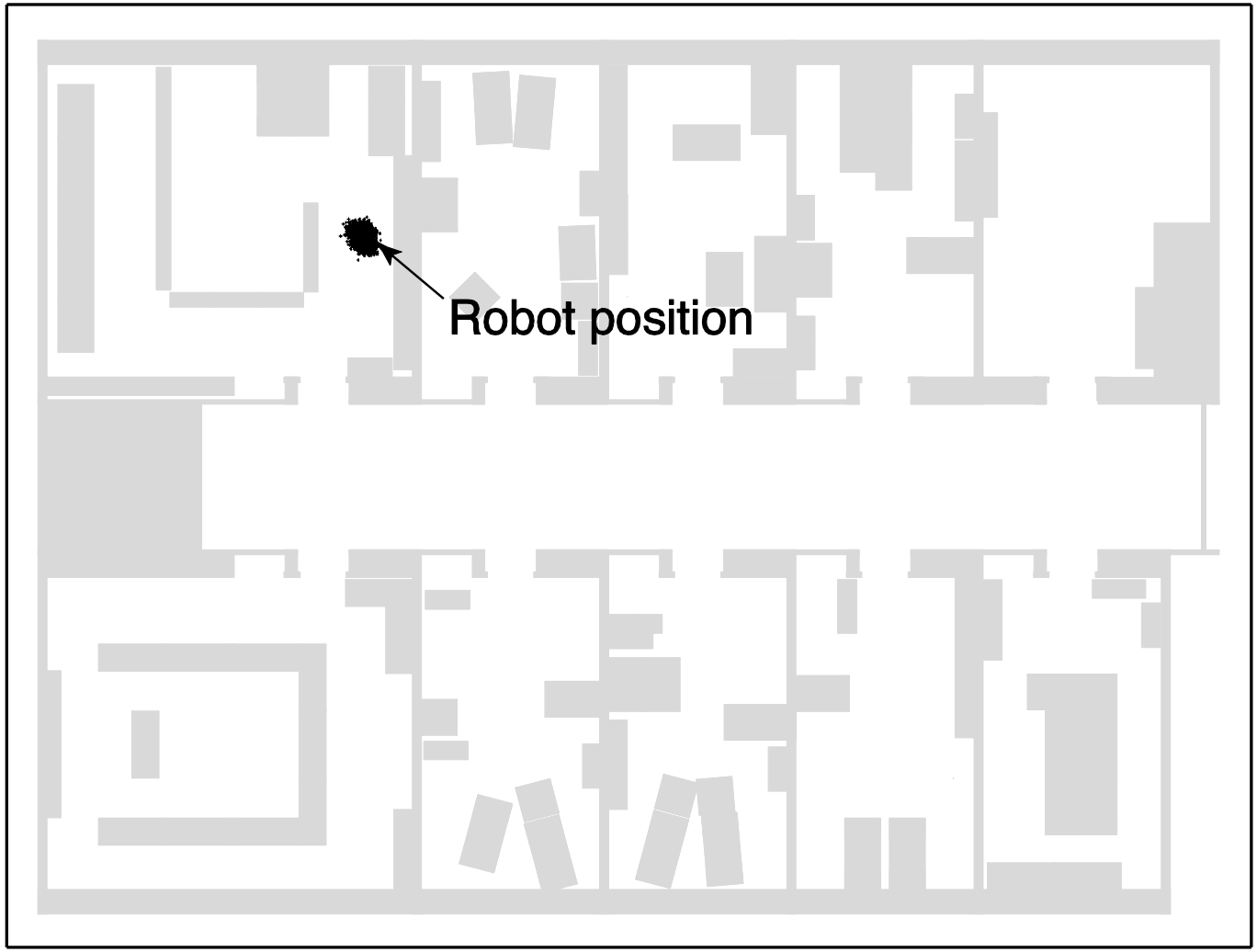


(a)

Particle Filters: bimodal belief

(b)

Particle Filters: Unimodal beliefs



(c)

Mapping and SLAM

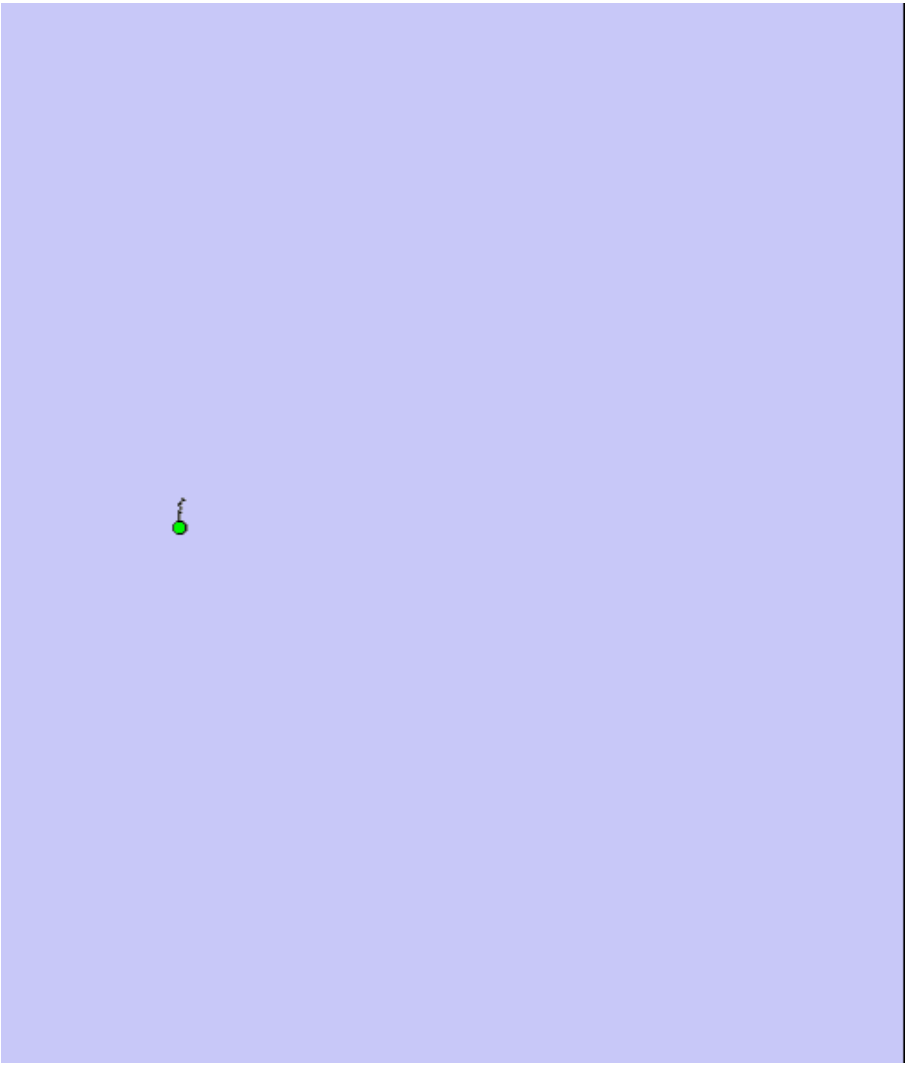
Localization: given map and observations, update pose estimation

Mapping: given pose and observation, update map

SLAM: given observations, update map and pose

- New observations increase uncertainty
- Loop closures reduce uncertainty

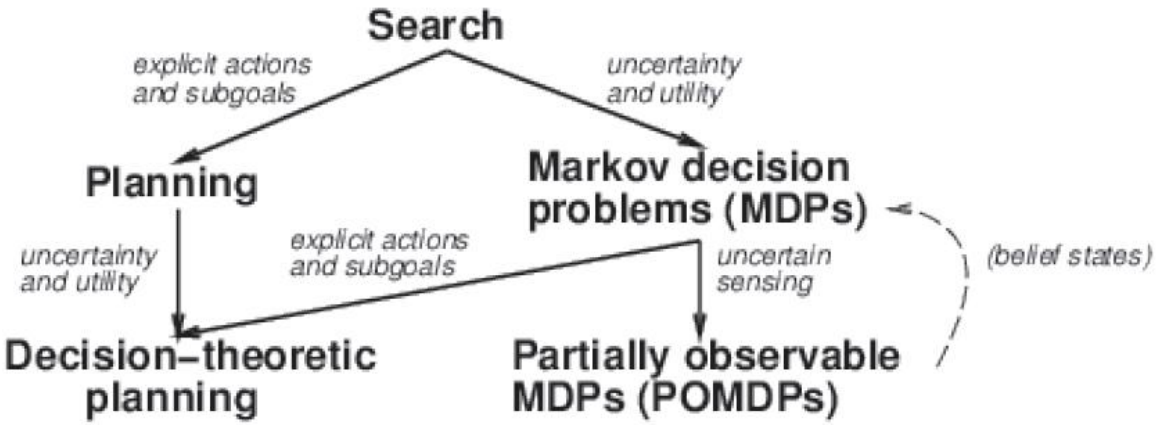
SLAM in action



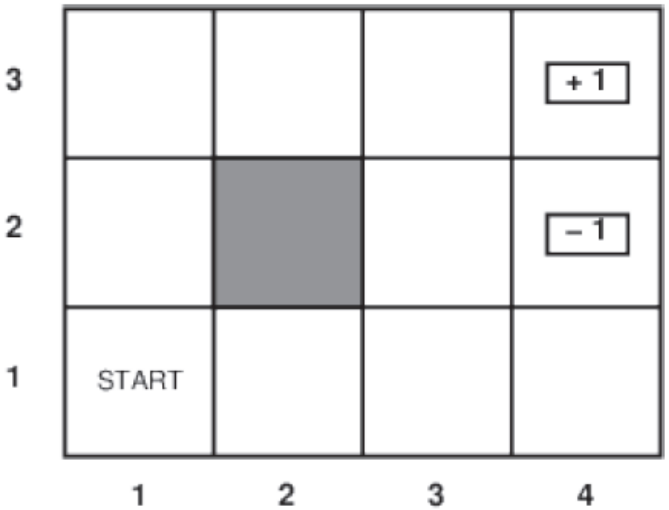
Courtesy of Sebastian Thrun and Dirk Haehnel (link for the video)

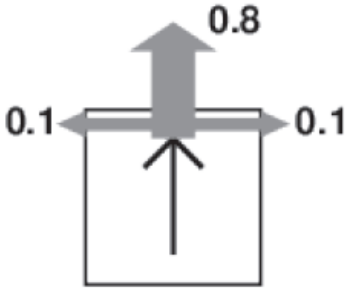
Markov Decision Process

- Mathematical model to plan <u>sequences of actions</u> in face of <u>uncertainty</u>



Example MDP





States $s \in S$, actions $a \in A$

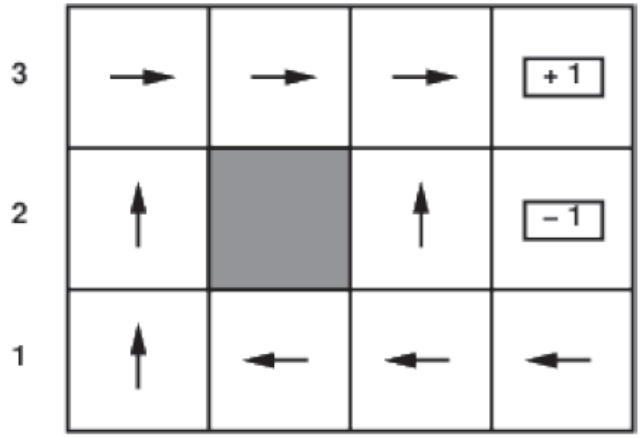
<u>Model</u> $T(s, a, s') \equiv P(s'|s, a)$ = probability that a in s leads to s'

<u>Reward function</u> $R(s)$ (or $R(s, a)$, $R(s, a, s')$)

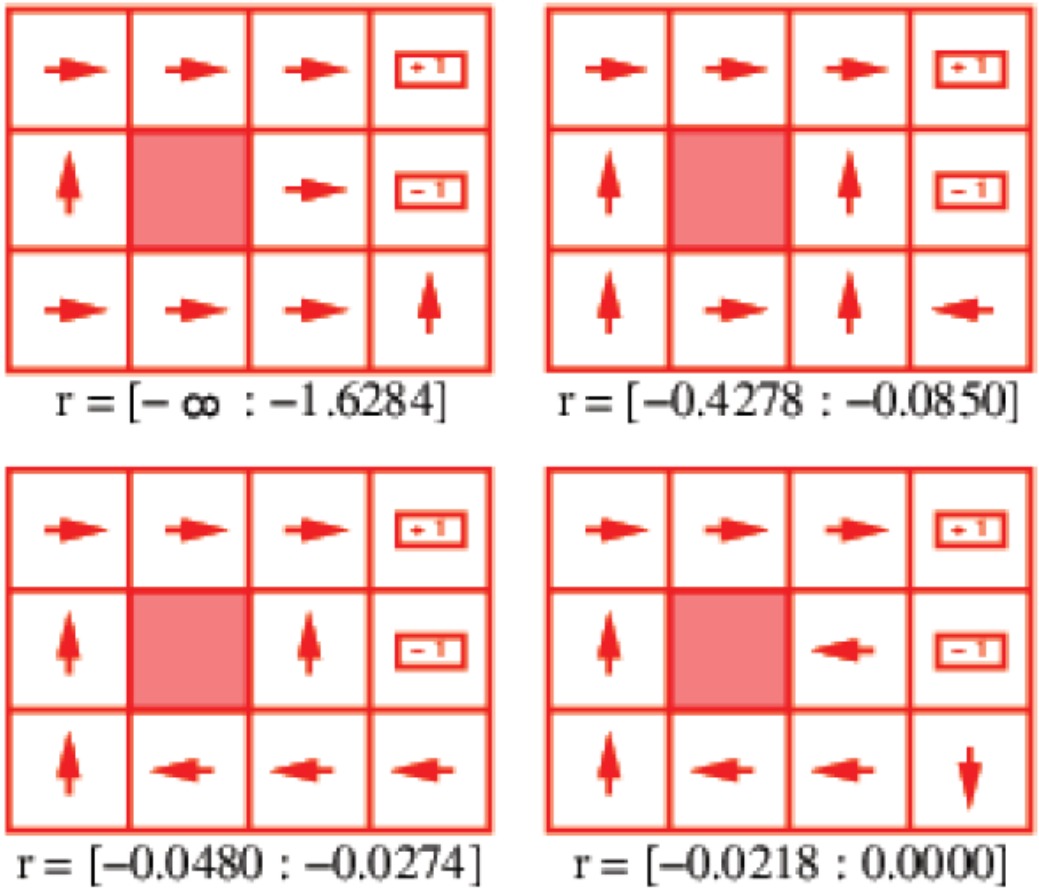
$$= \begin{cases} -0.04 & \text{(small penalty) for nonterminal states} \\ \pm 1 & \text{for terminal states} \end{cases}$$

Solving MDPs

In MDPs, aim is to find an optimal policy $\pi(s)$
i.e., best action for every possible state s
(because can't predict where one will end up)
The optimal policy maximizes (say) the expected sum of rewards
Optimal policy when state penalty $R(s)$ is -0.04:



Risk and Reward



Utility of State Sequences

Need to understand preferences between sequences of states
Typically consider stationary preferences on reward sequences

$$[r, r_0, r_1, r_2, \ldots] \succ [r, r'_0, r'_1, r'_2, \ldots] \Leftrightarrow [r_0, r_1, r_2, \ldots] \succ [r'_0, r'_1, r'_2, \ldots]$$

Theorem: there are only two ways to combine rewards over time.

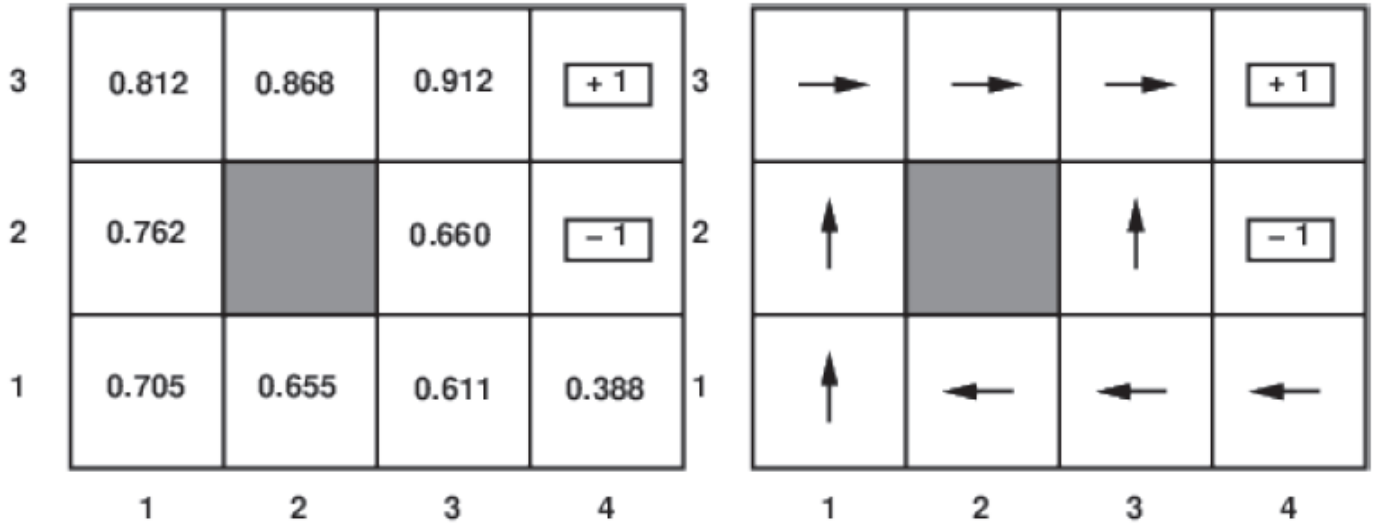
1) Additive utility function:
$U([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \cdots$
2) Discounted utility function:
$U([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$
where γ is the discount factor

Utility of States

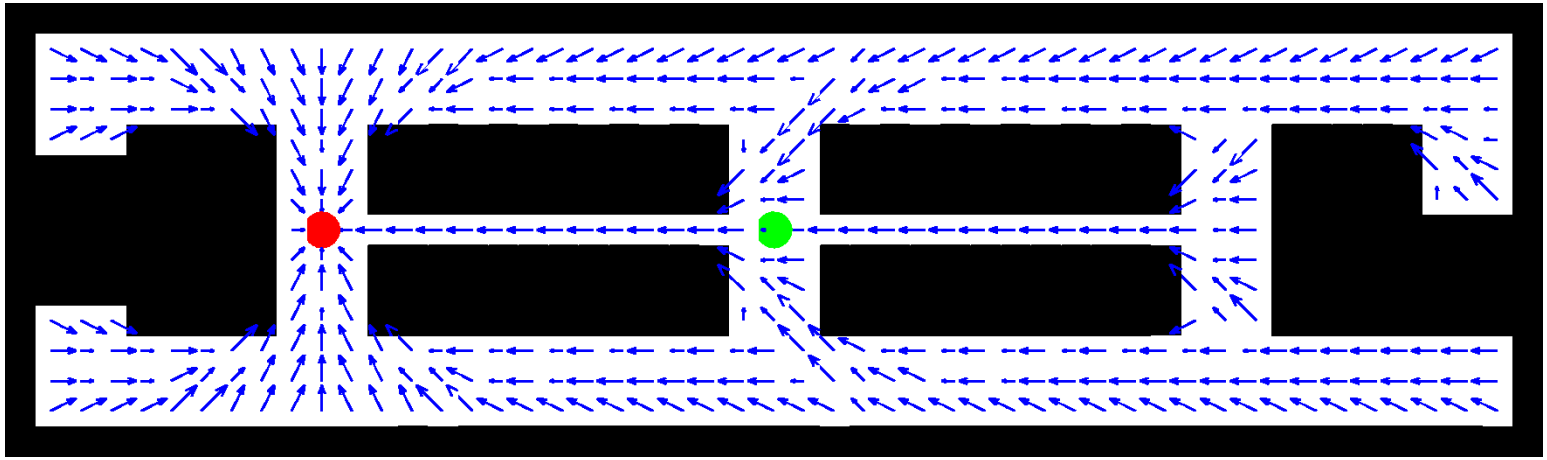
Utility of a state (a.k.a. its value) is defined to be

$U(s) =$
expected (discounted) sum of rewards (until termination)
assuming optimal actions

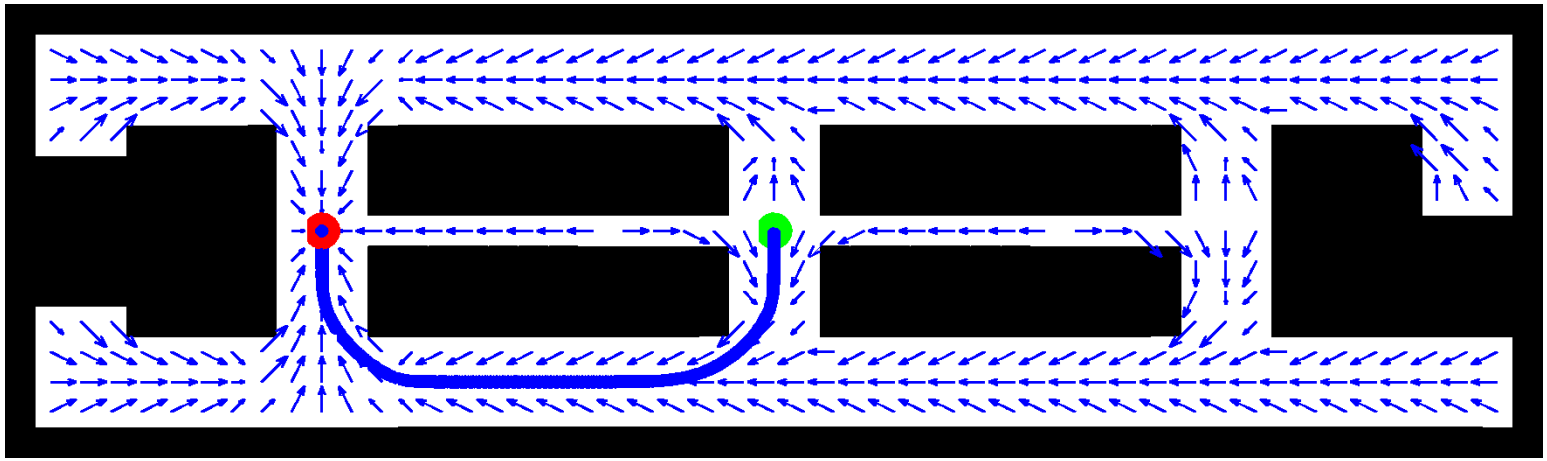
Given the utilities of the states, choosing the best action is just MEU:
maximize the expected utility of the immediate successors



MDPs for mobile robots



Optimal path (shortest) if actions are deterministic



Optimal path (safer) if actions are NOT deterministic

MDPs for mobile robots: formalization

Input:

- States x (Assume state is known)
- Actions u
- Transition probabilities $p(x'|u,x)$
- Reward / payoff function $r(x,u)$
  - *Note: now reward depends on state and action. This is a different notation, but the core concepts do not change.*

output

- 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):

$$\pi : \quad x_t \rightarrow \quad u_t$$

- 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

Main concepts for Policies

- Expected cumulative payoff of policy:

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

- Optimal policy:

$$\pi^* = \arg\max_{\pi} R_T^{\pi}(x_t)$$

- 1-step optimal policy:

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

- Value function of 1-step optimal policy:

$$V_1(x) = \gamma \max_{u} \; r(x,u) = \gamma r\left(x, \pi_1(x)\right)$$

2-step policies

- Optimal Policy

$$\pi_2(x) = \arg\max_u \left[r(x,u) + \int_{x'} V_1(x') p(x' | u, x) dx' \right]$$

- Value function

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

T-step policies

- Optimal Policy

$$\pi_T(x) = \arg\max_u \left[r(x,u) + \int_{x'} V_{T-1}(x') p(x' | u, x) dx' \right]$$

- Value function

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

Infinite Horizon

- Optimal Value Funciton (Bellman equation)

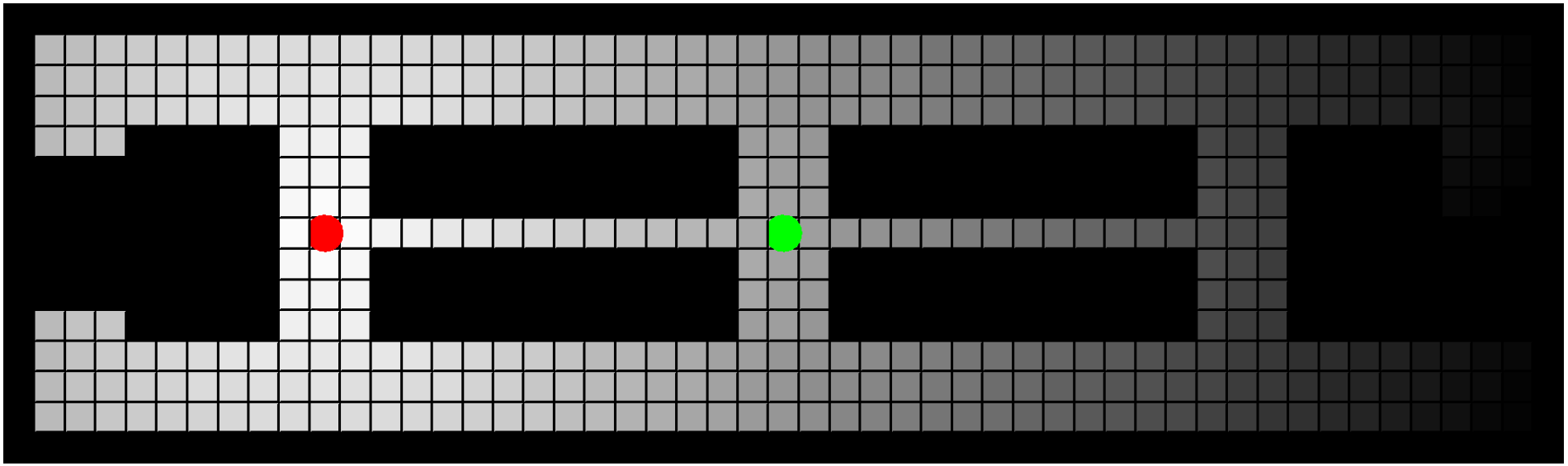
$$V_\infty(x) = \gamma \max_u \left[r(x,u) + \int_{x'} V_\infty(x') p(x' | u, x) dx' \right]$$

- Can be used to compute the optimal policy

$$\pi^*(x) = \arg\max_u \left[r(x,u) + \int_{x'} V_\infty(x') p(x' | u, x) dx' \right]$$

Value Iteration: idea

- Initialize V with random value
- Untill no change
  - For all state x
    - Update V(x) to make it locally consistent



Value Iteration

- For all x

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

- Repeat untill convergence
  - For all x

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

- Compute

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

Value function and policy iteraton

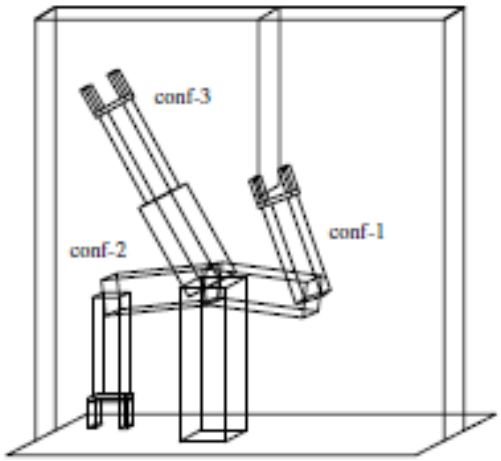
- Often the optimal policy has been reached long before the value function has converged.
- Policy iteration calculates a new policy based on the current value function and then calculates a new value function based on this policy.
- This process often converges faster to the optimal policy.

$$\hat{V}(x) = \gamma\left[r(x, \pi(x)) + \int_{x'} \hat{V}(x') p(x' | \pi(x), x) dx' \right]$$

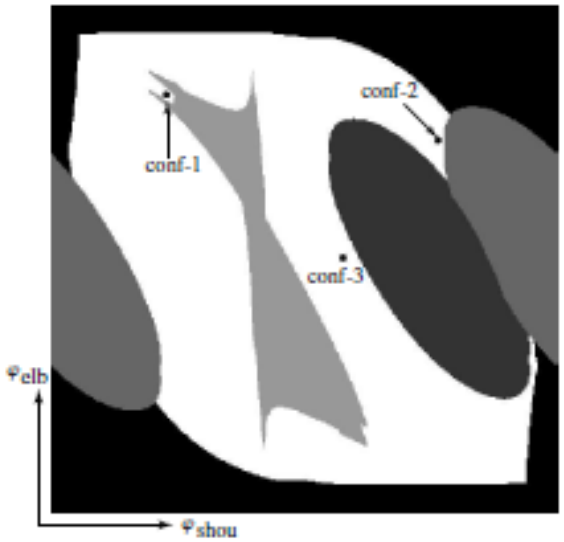
Motion Planning for Mobile Robots

Plan for motion in free configuration space (not workspace)

workspace

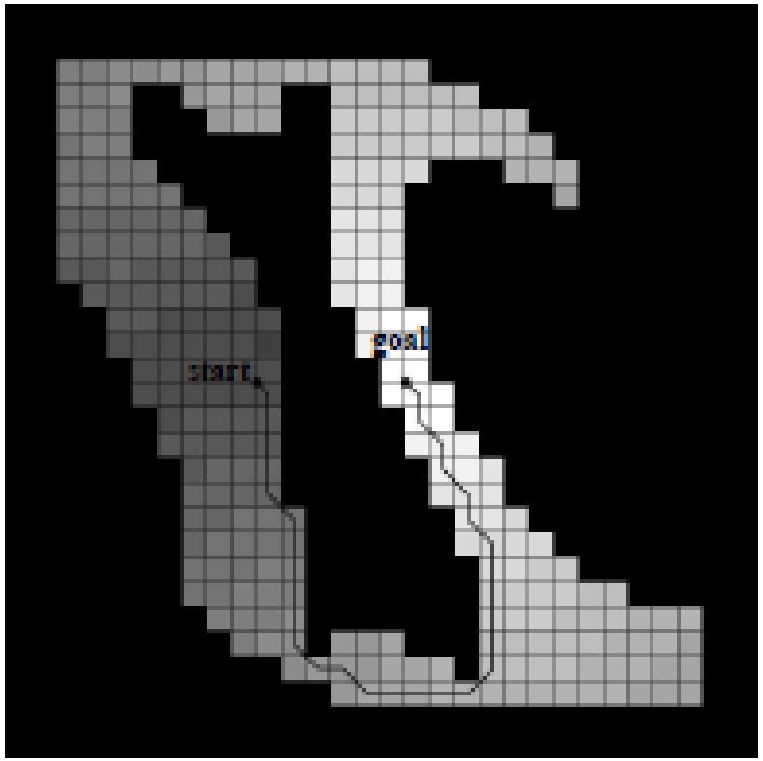


configuration space

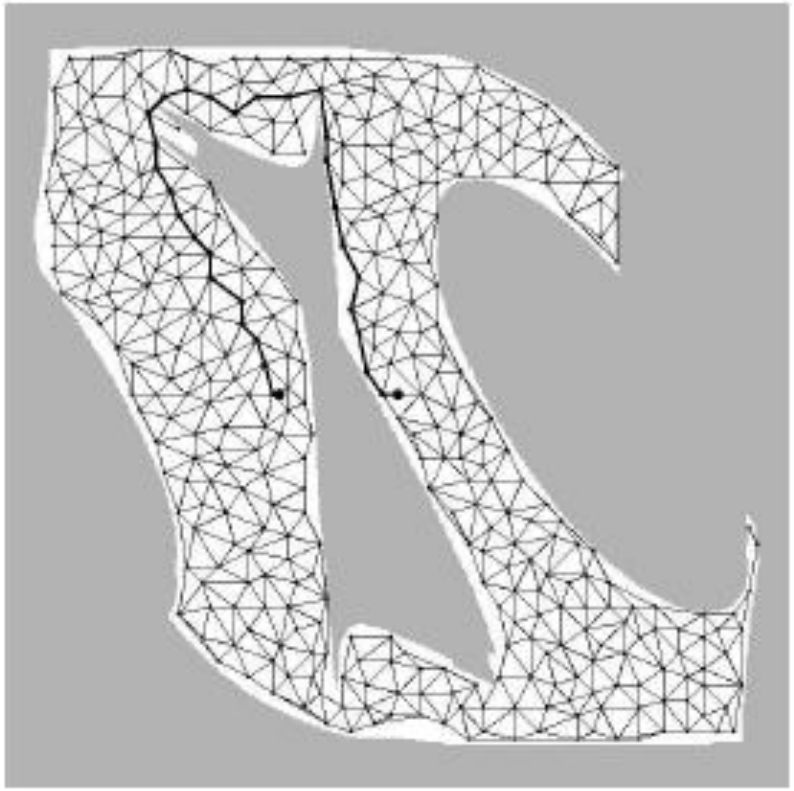


Configuration Space Planning

Convert free configuration space in finite state space



Cell decomposition



Skeletonization (PRM)

Planning the motion

Given finite state space representing free configuration space

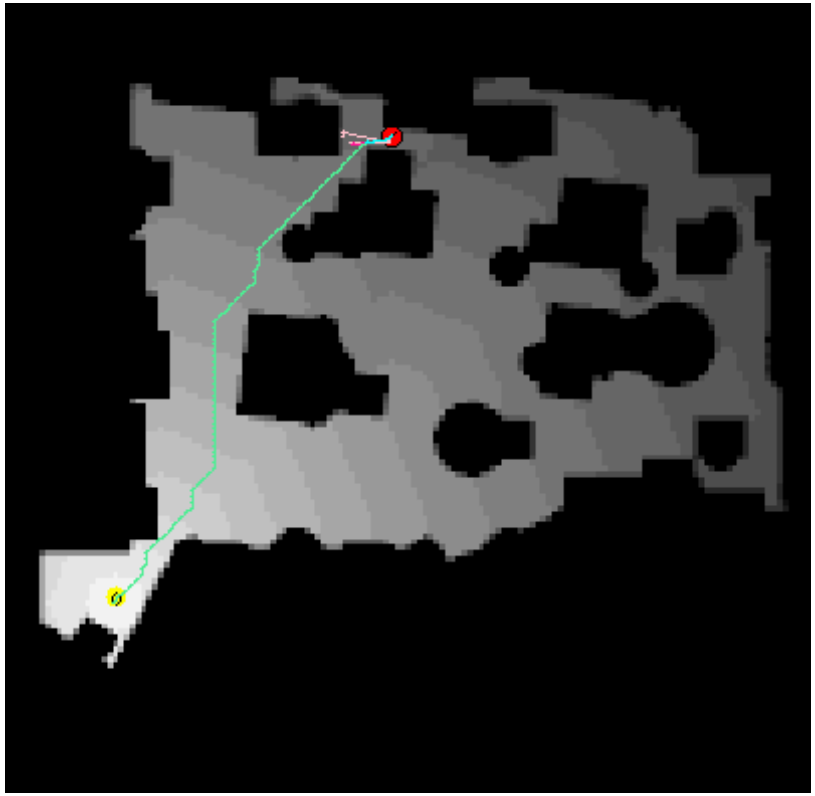
Find a <u>sequence</u> of states from start to goal

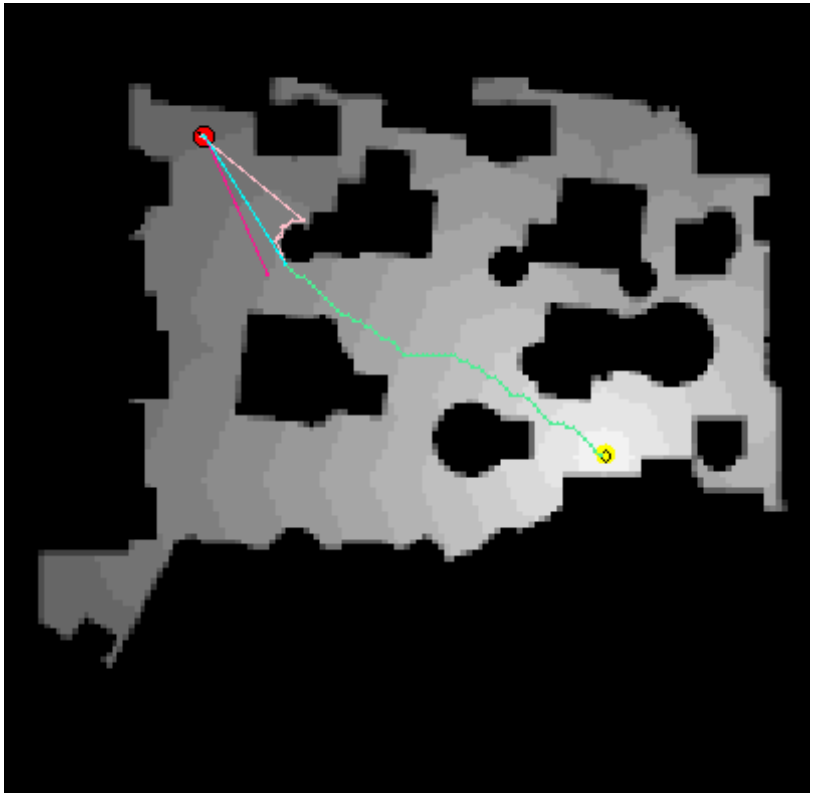
Several approaches:

- Rapidly-exploring Random Trees (RRT)
- Potential Fields
- Markov Decision Processes

  (i.e. building a navigation function)

MDP for robot navigation





NOTE: pose (i.e., state) is unknown → not an MDP!

- Assume localization works (decently)
- State is the most probable pose (mode of the posterior)

Summary

- Robots must consider uncertainty when planning
- Markov Decision Processes
  - Powerful model to plan a sequence of actions under uncertainty
  - Key point: define value of states considering expected cumulative reward
  - Value (policy) iteration to solve the model
- Motion Planning:
  - Planning problem in finite state space (C-free)
  - MDPs powerful techniques to build navigation functions (for low-dimension)

References and Further Readings

Material for the slides

- Russel and Norvig; Artificial Intelligence a Modern Approach (Chapter 25)
- Thrun, Burgard, Fox; Probabilistic Robotics (Chapters 2 and 14)

Further readings

- Latombe; Robot Motion Planning
- La Valle, Kuffner; Randomized Kinodynamic Planning
- Thrun,Fox,Burgard; A probabilistic approach to concurrent mapping and localization for mobile robots