Visual Navigation for Flying Robots

Planning under Uncertainty, Exploration and Coordination

Dr. Jürgen Sturm

Agenda for Today

- Planning under Uncertainty
- Exploration with a single robot
- Coordinated exploration with a team of robots
- Coverage

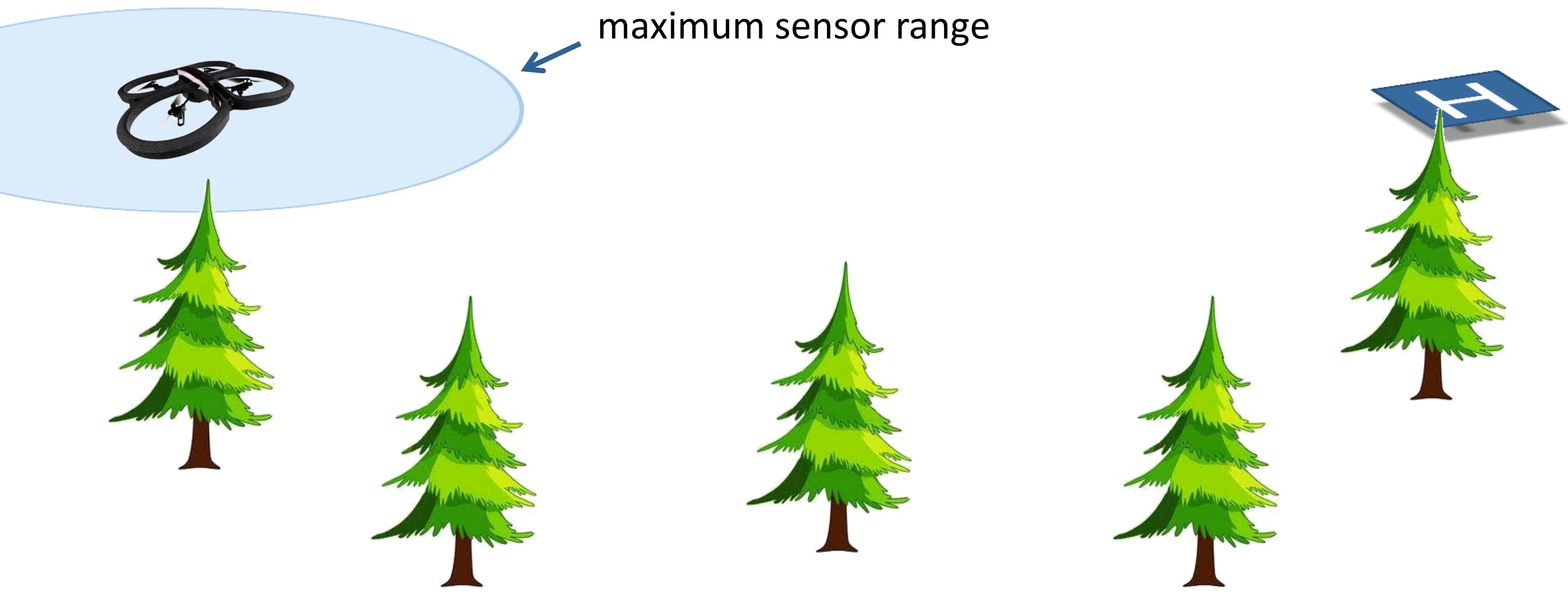
Agenda For Next Week

- **First half:** Good practices for experimentation, evaluation and benchmarking
- **Second half:** Time for your questions on course material

→Prepare your questions (if you have)

Motivation: Planning under Uncertainty

- Consider a robot with range-limited sensors and a feature-poor environment
- Which route should the robot take?



Reminder: Performance Metrics

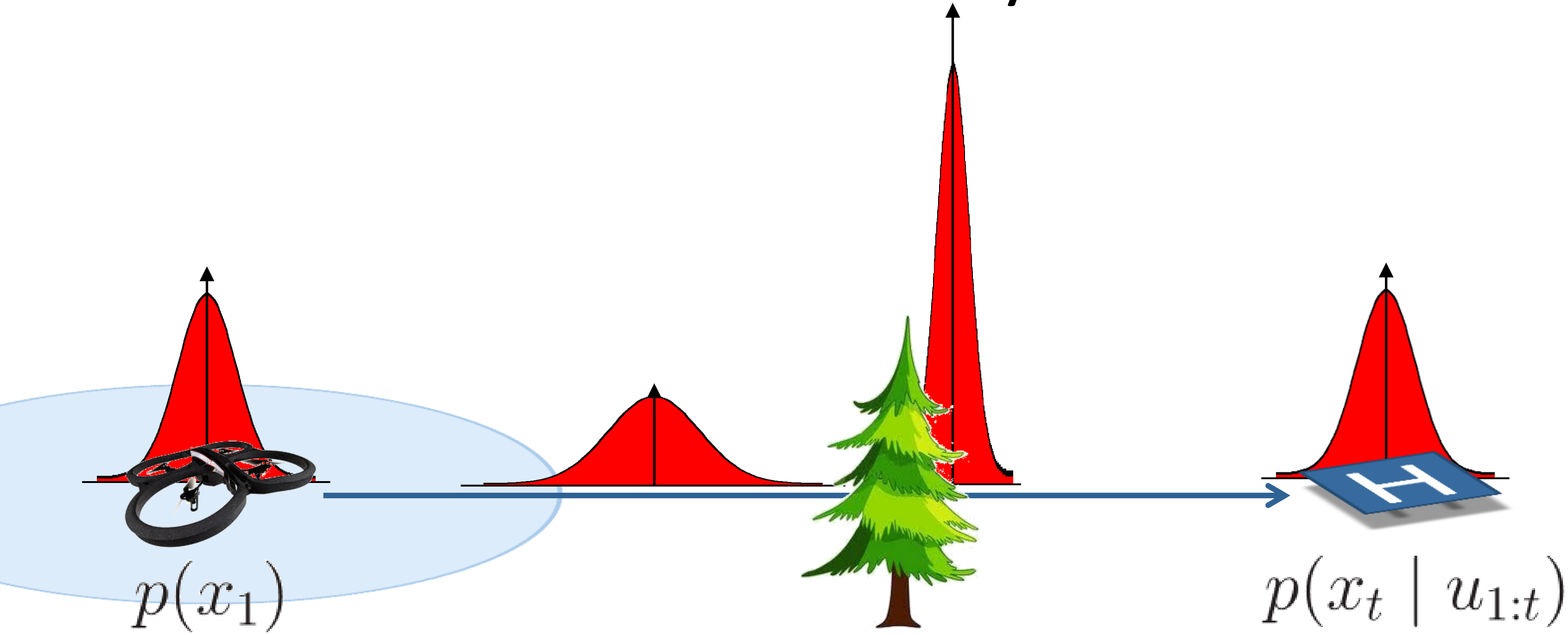
- Execution speed / path length
- Energy consumption
- Planning speed
- Safety (minimum distance to obstacles)
- **Robustness against disturbances**
- **Probability of success**
- …

Reminder: Belief Distributions

- In general, actions of the robot are not carried out perfectly
- Position estimation ability depends on map
- Let’s look at the belief distributions…

Reminder: Belief Distributions

- Actions increase the uncertainty (in general)
- Observations decrease the uncertainty (always)
- Observations are not always available

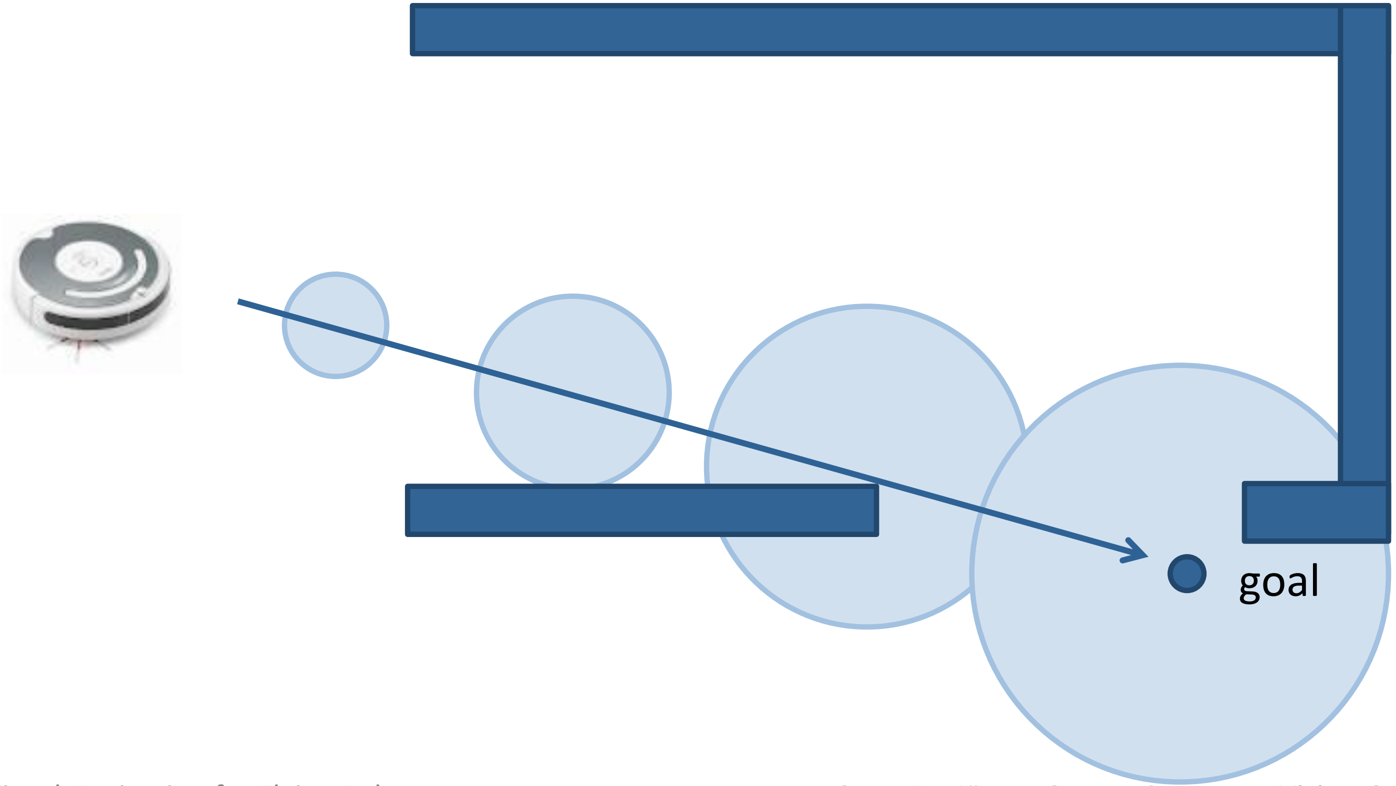


Solution 1: Shape The Environment To Decrease Uncertainty

- Assume a robot without sensors
- What is a good navigation plan?

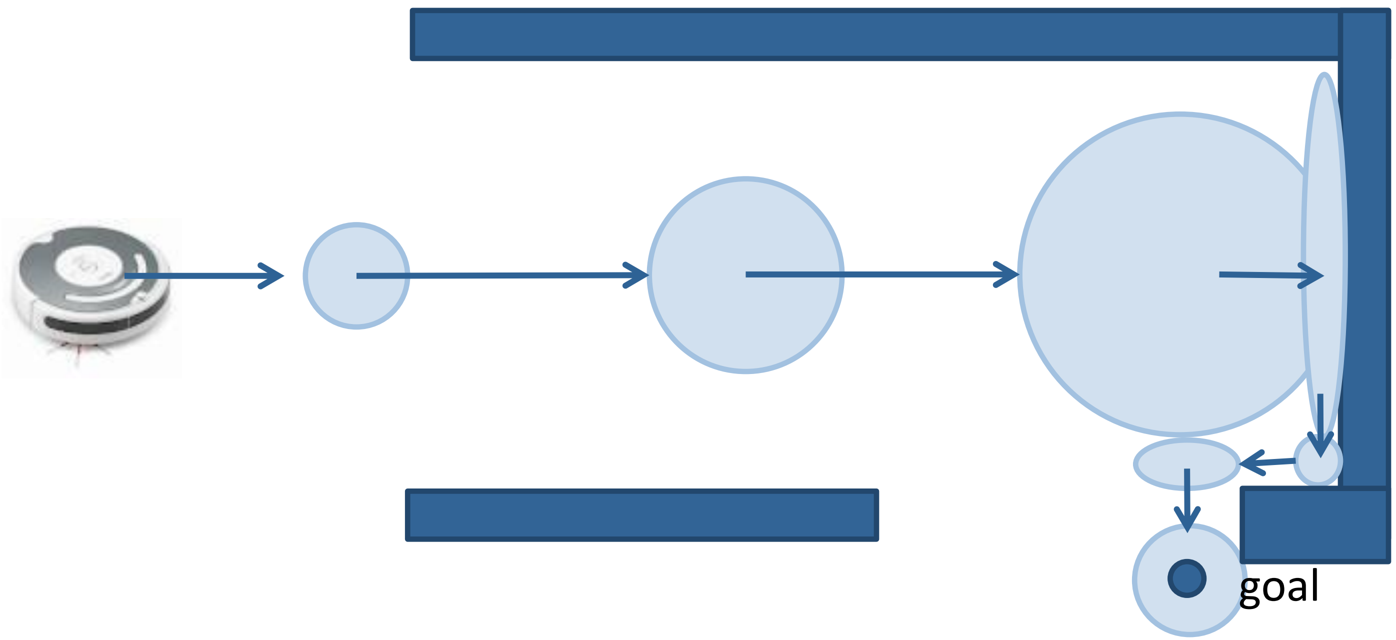
Solution 1: Shape The Environment To Decrease Uncertainty

- Plan 1: Take the shortest path
- What is the probability of success of plan 1?



Solution 1: Shape The Environment To Decrease Uncertainty

- What is the probability of success of plan 2?



Solution 1: Shape The Environment To Decrease Uncertainty

- **Pro:** Simple solution, need fewer/no sensors
- **Con:** Requires task specific design/engineering of both the robot and the environment
- Applications:
  - Docking station
  - Perception-less manipulation (on conveyer belts)
  - …

Solution 2: Add (More/Better) Sensors

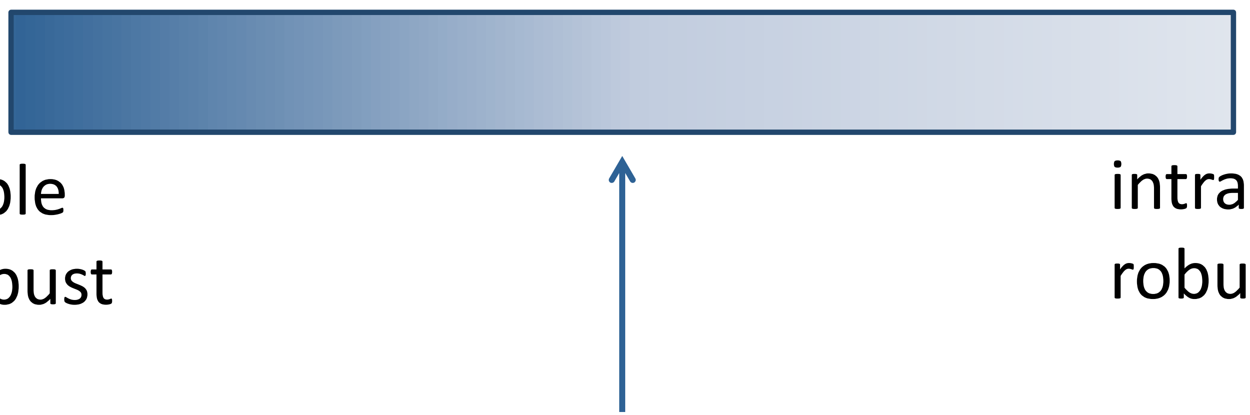
Solution 3: POMDPs

- Partially observable Markov decision process (POMDP)
- Considers uncertainty of the motion model and sensor model
- Finite/infinite time horizon
- Resulting policy is optimal
- One solution technique: Value iteration
- **Problem:** In general (and in practice) computationally intractable (PSPACE-hard)

Continuum of Possible Approaches to Motion Planning

Conventional
path planner

POMDP

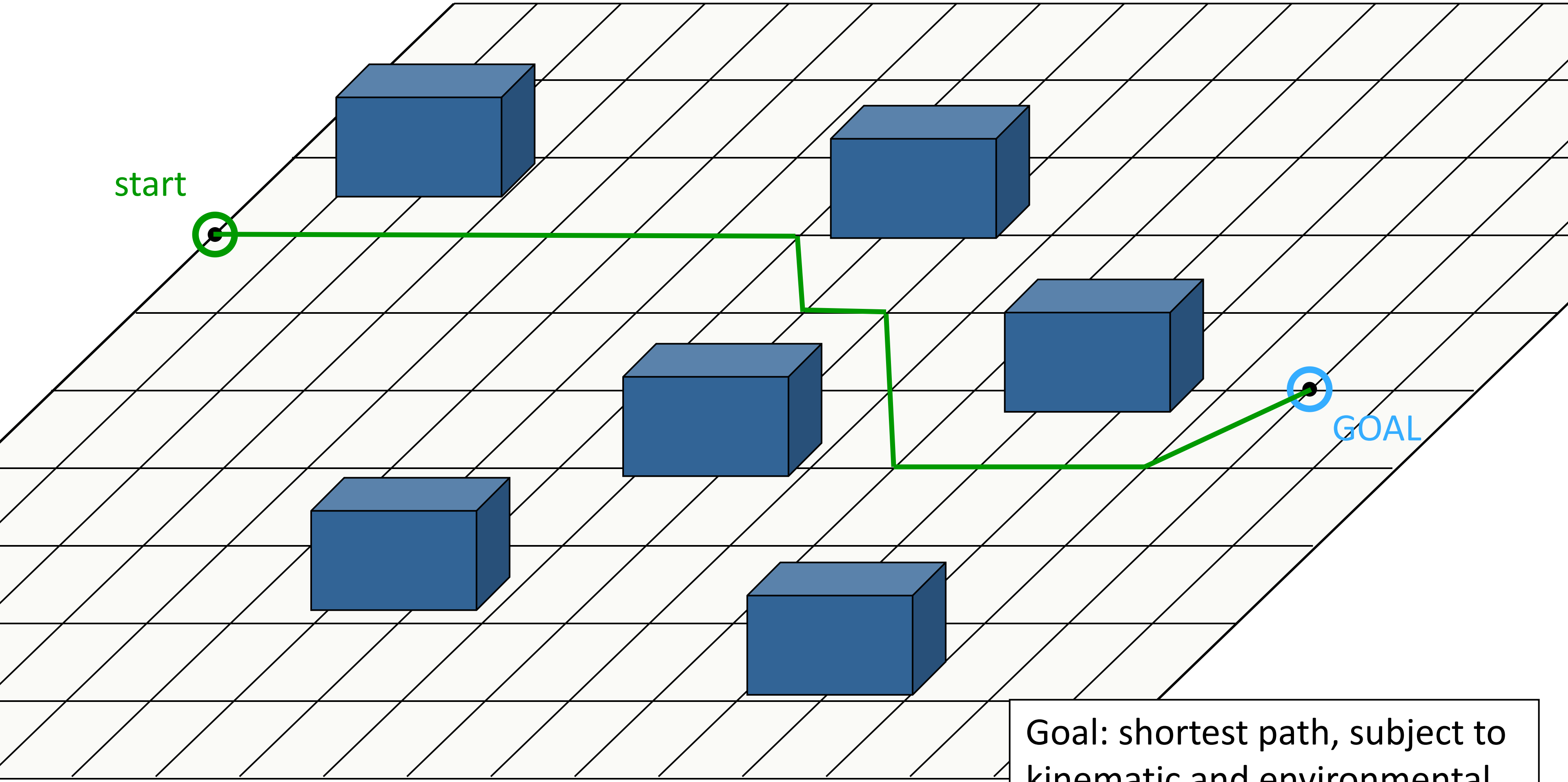


tractable
not robust

intractable
robust

maybe we can find
something in between…

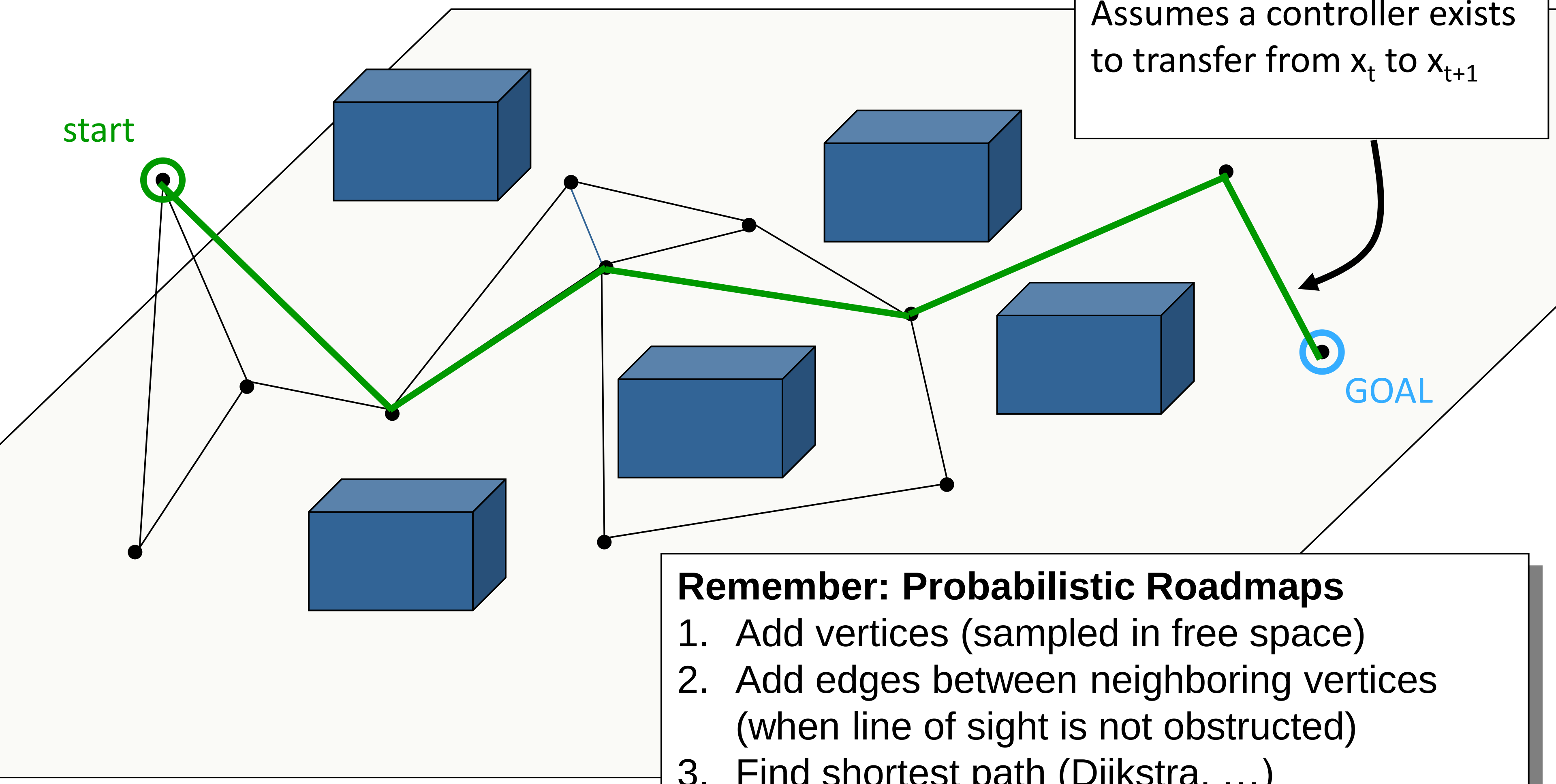
Remember: Motion Planning



Goal: shortest path, subject to kinematic and environmental constraints

Slides adopted from Nick Roy

Remember: Motion Planning in High-Dimensional Configuration Spaces



Assumes a controller exists to transfer from x_t to x_{t+1}

Remember: Probabilistic Roadmaps

1. Add vertices (sampled in free space)
2. Add edges between neighboring vertices (when line of sight is not obstructed)
3. Find shortest path (Dijkstra, …)

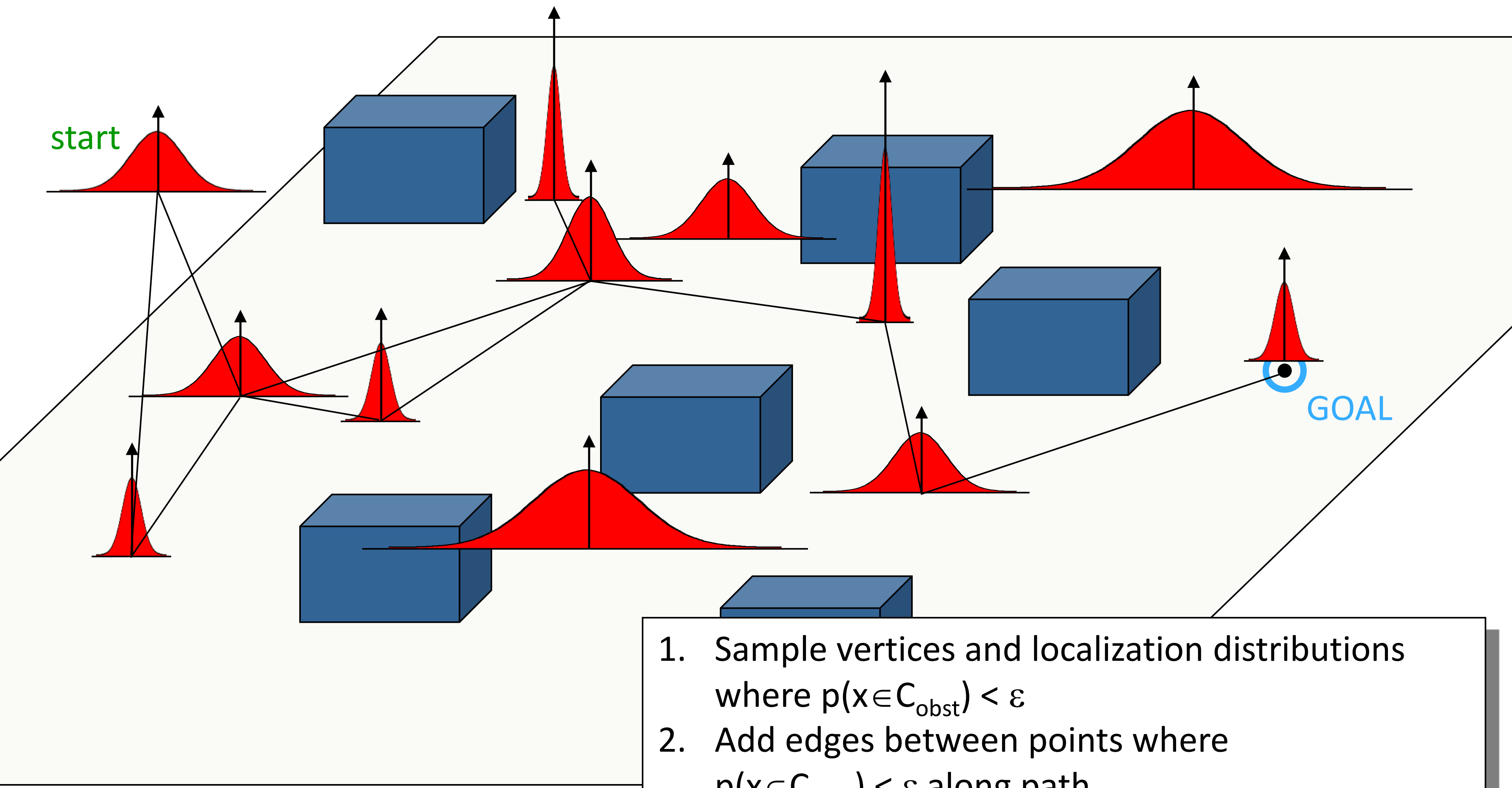
Slides adopted from Nick Roy

Remember: Motion Planning in High-Dimensional Configuration Spaces

- **Problem:** The roadmap does not consider the sensor capabilities of the robot
- Can the robot actually keep position at each vertex?
  - Can it localize at the vertex?
  - Given localization abilities, what is the probability of hitting into an obstacle?
- Can the robot robustly navigate between two vertices?
  - Line of sight is not enough
  - Robot might get lost or hit into an obstacle

Motion Planning in Information Space

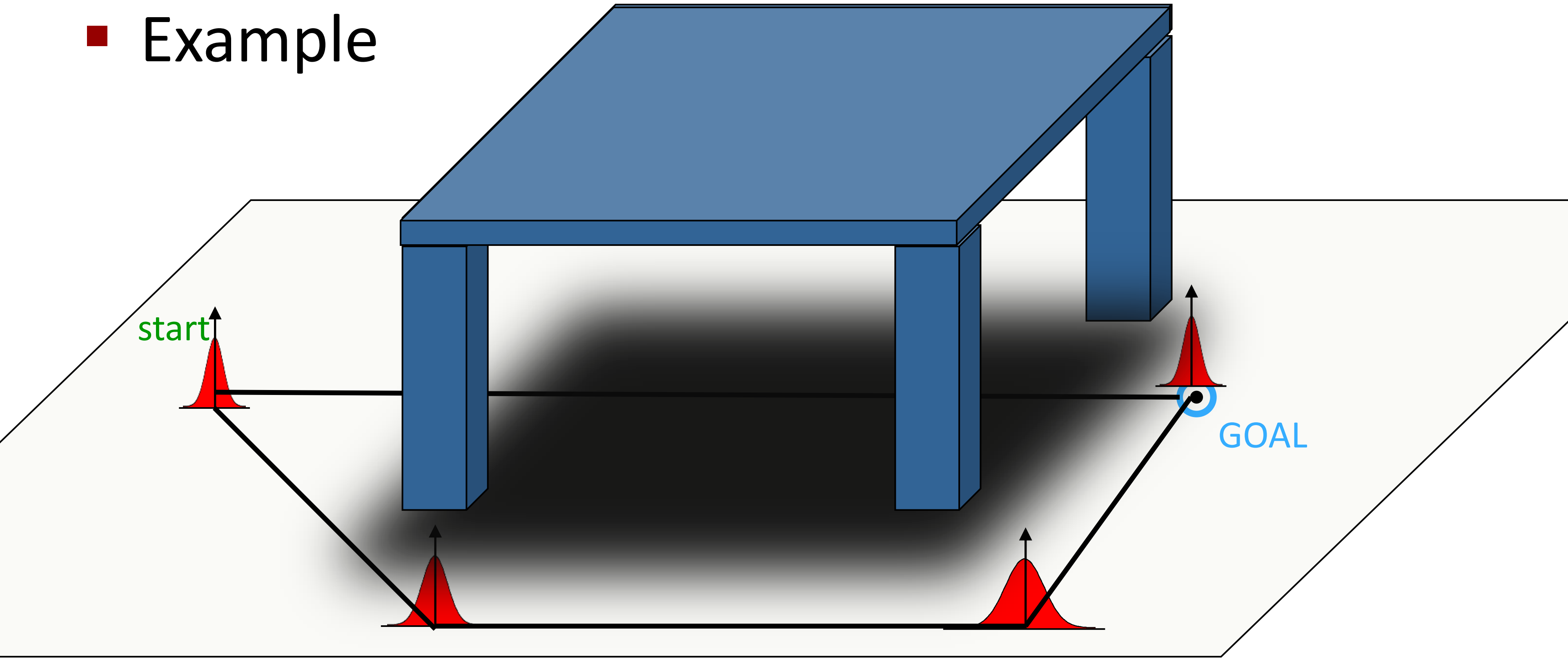
[Roy et al.]



1. Sample vertices and localization distributions where $p(x \in C_{obst}) < \varepsilon$
2. Add edges between points where $p(x \in C_{obst}) < \varepsilon$ along path
3. Perform graph search

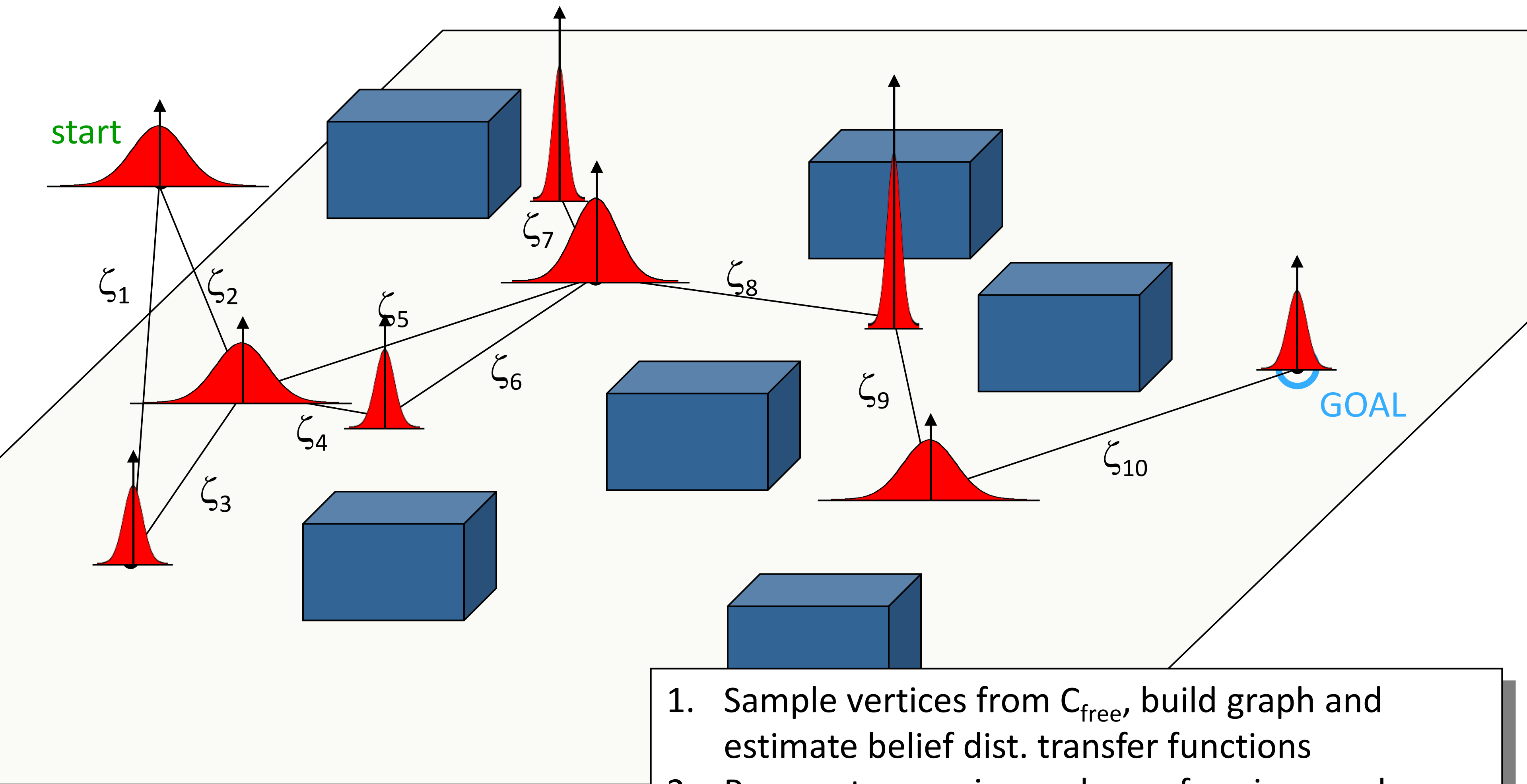
Motion Planning in Information Space

- **Problem:** Posterior distribution depends also on the path taken to the vertex
- Example



Belief Roadmap

[He et al., 2008]



1. Sample vertices from C_{free}, build graph and estimate belief dist. transfer functions
2. Propagate covariances by performing graph search

Slides adopted from Nick Roy

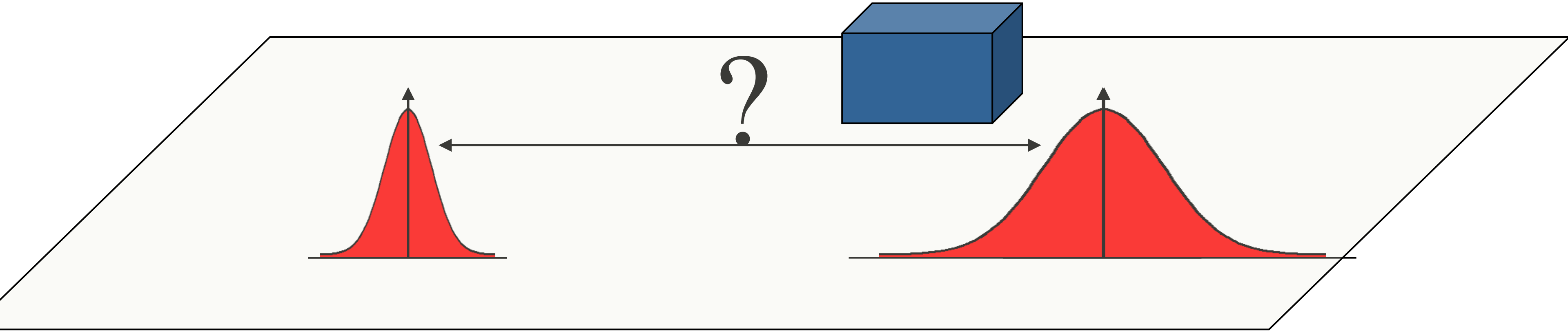
Planning in Information Spaces
[He et al., 2008]

- **Given:** Roadmap
- **Goal:** Find path from start to goal nodes that results in minimum uncertainty at goal
- **Problem:** How can we estimate the belief distribution at the goal (efficiently)?

Planning in Information Spaces
[He et al., 2008]

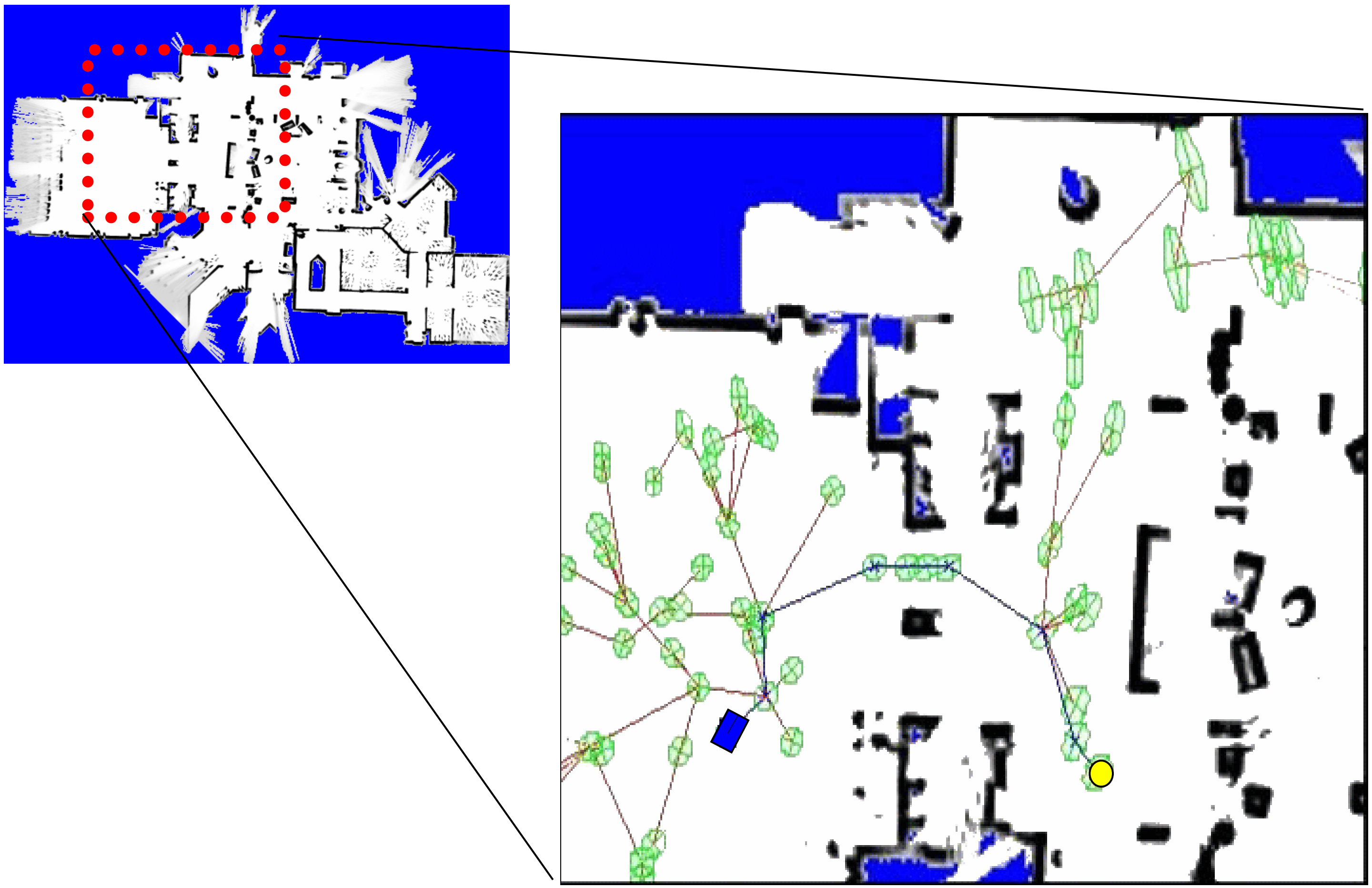
How can we propagate the belief distribution along an edge?

1. Sample waypoints, use forward simulation to compute full posterior
2. Linearize model and use Kalman filter



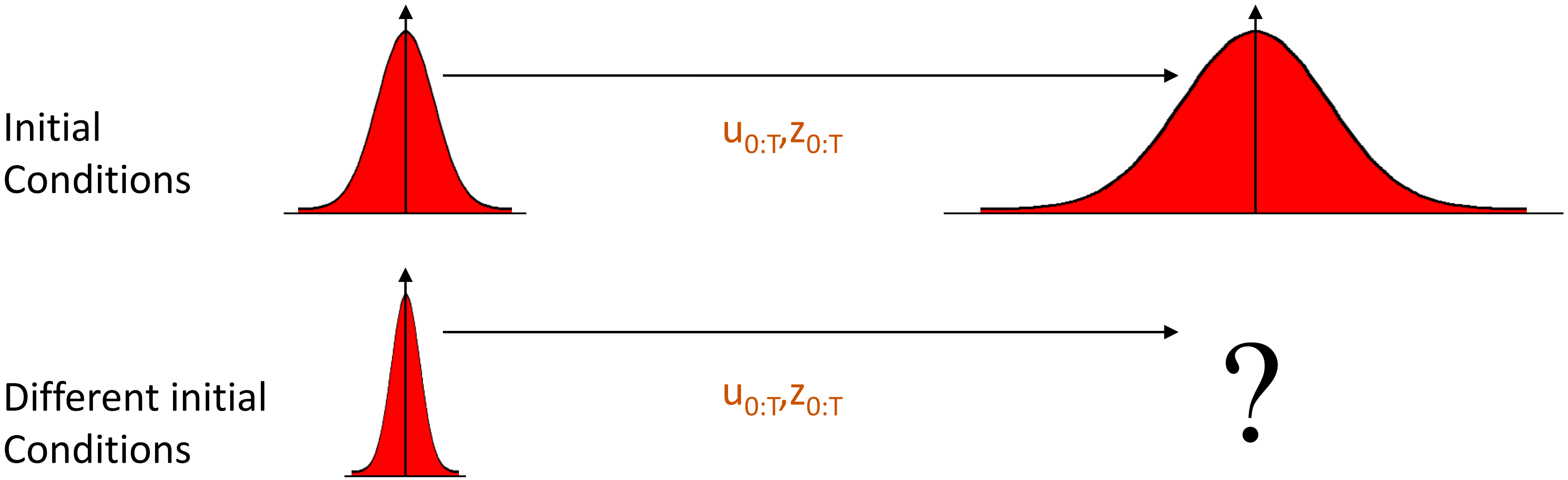
Example: Belief Roadmap

[He et al., 2008]



Belief Propagation
[He et al., 2008]

- The posterior distribution depends on the prior distribution



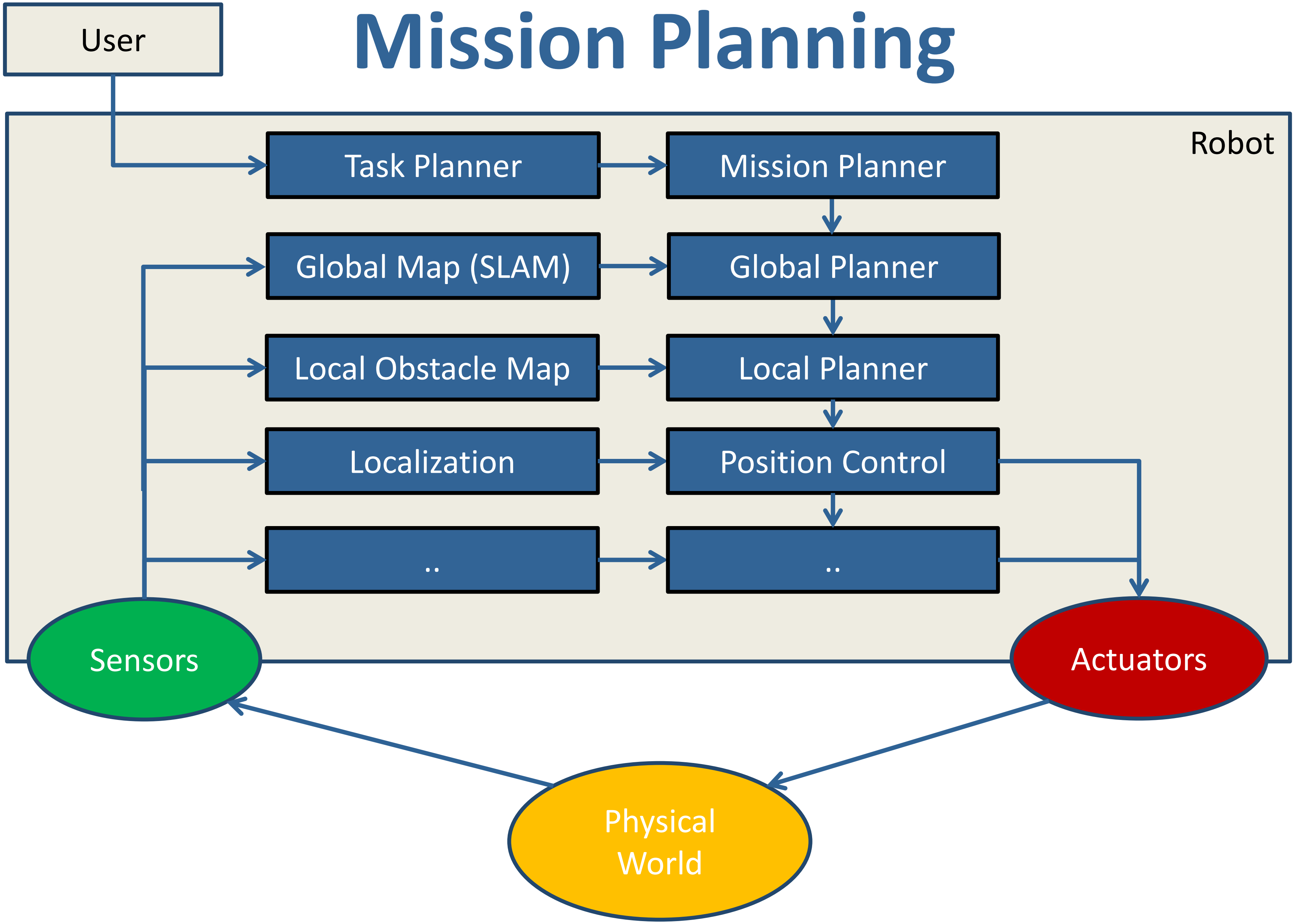
Planning in Information Spaces
[He et al., 2008]

- The posterior distribution at a vertex depends on the prior distribution (and thus on path to the vertex)
- Need to perform forward simulation (and belief prediction) along each edge for every start state
- Computing minimum cost path of 30 edges: ≈100 seconds

Summary: Planning Under Uncertainty

- Actions and observations are inherently noisy
- Planners neglecting this are not robust
- Consider the uncertainty during planning to increase robustness

Mission Planning



Mission Planning

- **Goal:** Generate and execute a plan to accomplish a certain (navigation) task
- Example tasks
  - Exploration
  - Coverage
  - Surveillance
  - Tracking
  - …

Task Planning

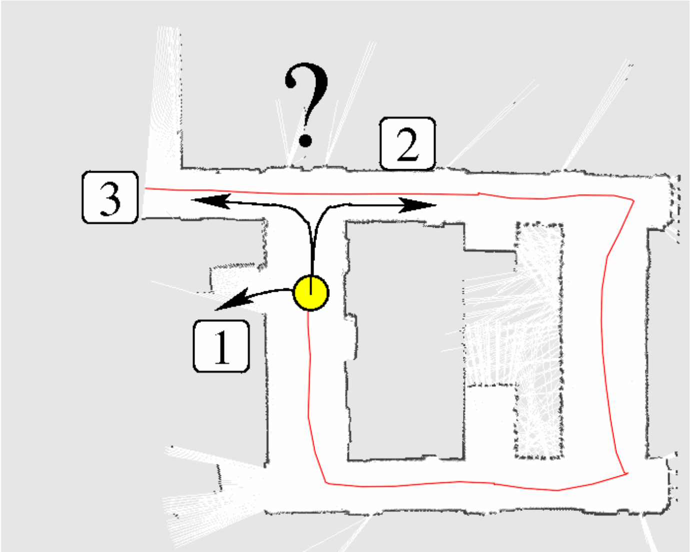
- **Goal:** Generate and execute a high level plan to accomplish a certain task
- Often symbolic reasoning (or hard-coded)
  - Propositional or first-order logic
  - Automated reasoning systems
  - Common programming languages: Prolog, LISP
- Multi-agent systems, communication
- Artificial Intelligence

Exploration and SLAM

- SLAM is typically passive, because it consumes incoming sensor data
- Exploration actively guides the robot to cover the environment with its sensors
- Exploration in combination with SLAM: Acting under pose and map uncertainty
- Uncertainty should/needs to be taken into account when selecting an action

Exploration

- By reasoning about control, the mapping process can be made much more effective
- Question: **Where to move next?**



- This is also called the **next-best-view problem**

Exploration

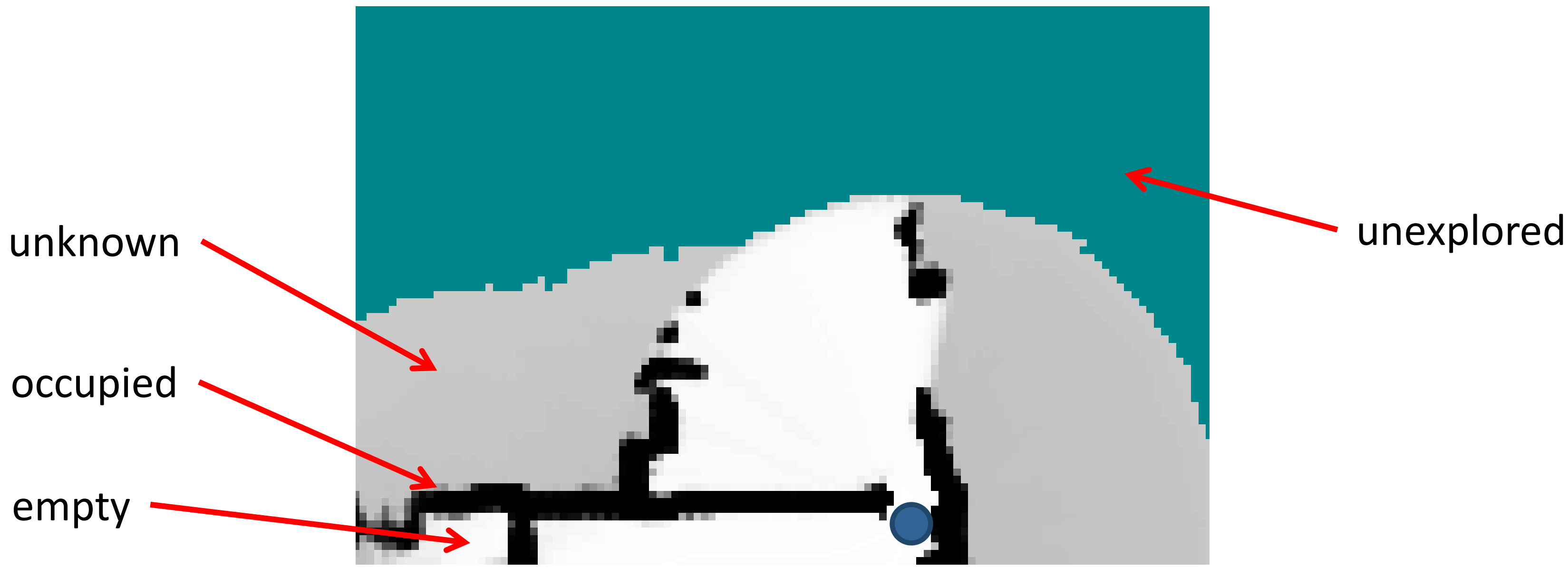
- Choose the action that maximizes utility

$$a^* = \arg\max_{a \in A} U(m, a)$$

- **Question:** How can we define utility?

Example

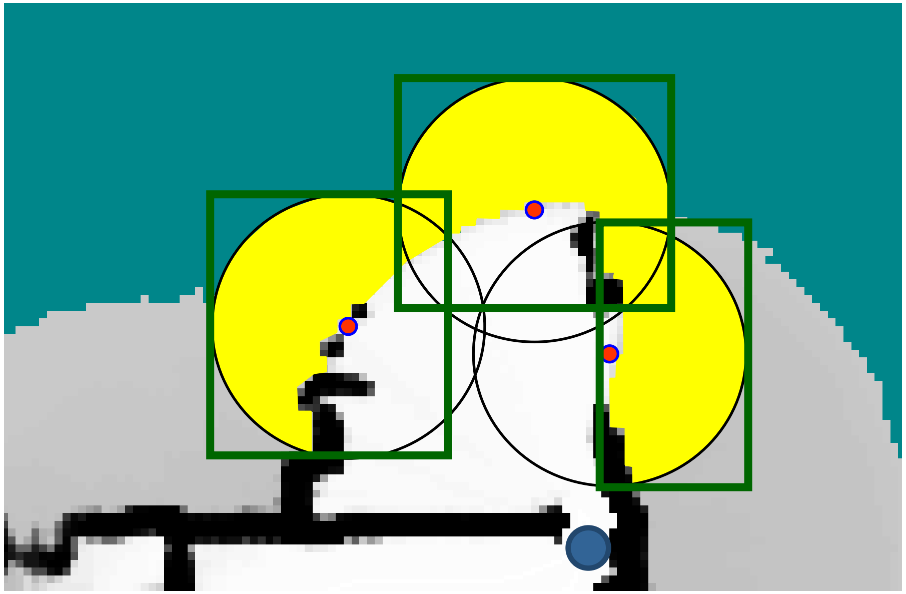
- Where should the robot go next?



Maximizing the Information Gain

- Pick the action a that maximizes the **information gain** given a map m

$$a^* = \arg\max_{a \in A} IG(m, a)$$



Information Theory

- **Entropy** is a general measure for the uncertainty of a probability distribution
- Entropy = Expected amount of information needed to encode an outcome $X = x$

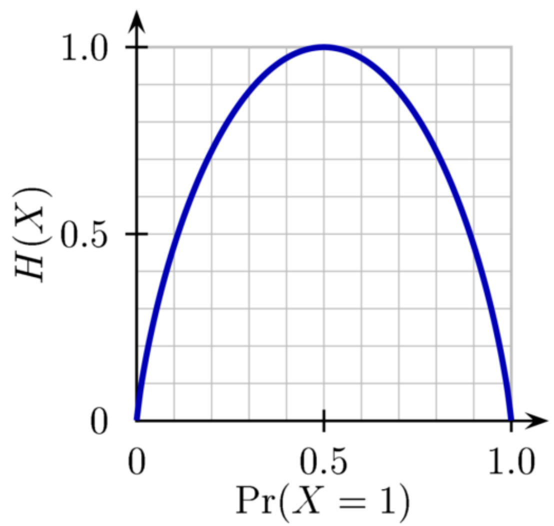
$$
\begin{aligned}
H(X) &= E(I(X)) \\
&= E(-\log p(X)) \\
&= -\sum_{i=1}^{n} p(x_i) \log p(x_i)
\end{aligned}
$$

Example: Binary Random Variable

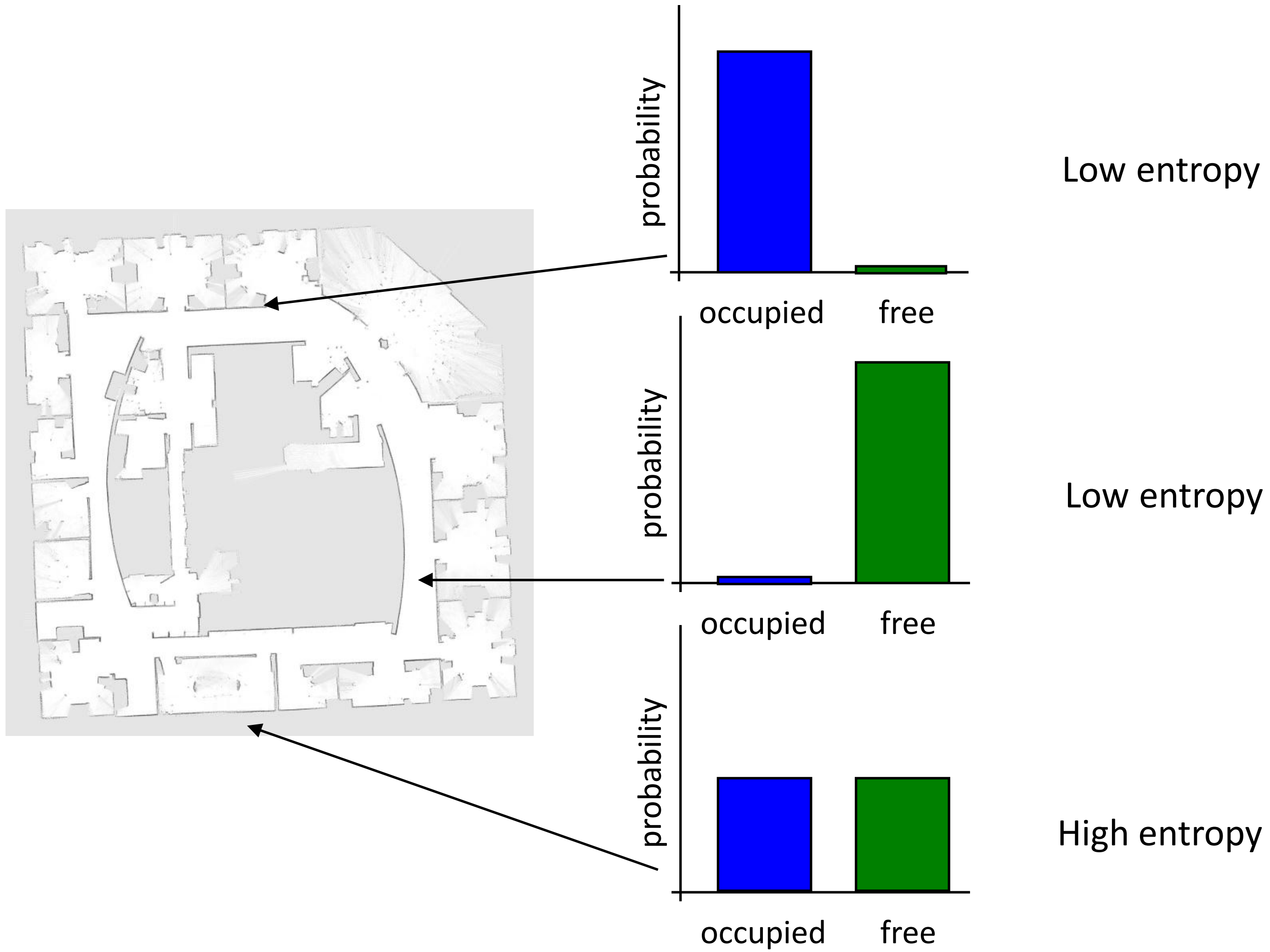
- Binary random variable $X \in \{0, 1\}$
- Probability distribution $P(X = 1) = p$
- How many bits do we need to transmit one sample of $p(X)$?
  - For p=0?
  - For p=0.5?
  - For p=1?

Example: Binary Random Variable

- Binary random variable $X \in \{0, 1\}$
- Probability distribution $P(X = 1) = p$
- How many bits do we need to transmit one sample of $p(X)$?
- Answer:



Example: Map Entropy



The overall entropy is the sum of the individual entropy values

Information Theory

- **Information gain** = Uncertainty reduction

$$IG(X, Y) = H(X) - H(X \mid Y)$$

- **Conditional entropy**

$$H(X \mid Y) = \sum_{i,j} p(x_i, y_j) \log \frac{p(y_j)}{p(x_i, y_j)}$$

Maximizing the Information Gain

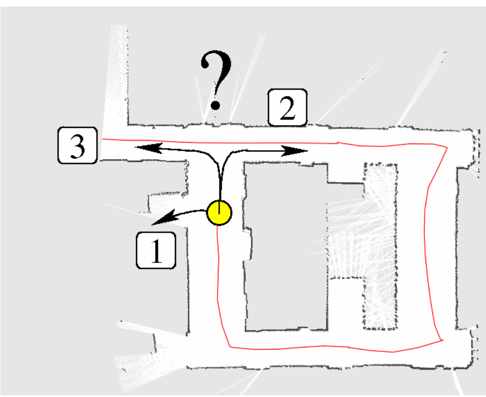
- To compute the information gain one needs to know the observations obtained when carrying out an action

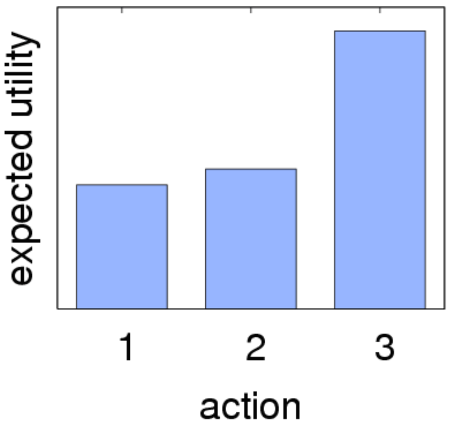
$$a^* = \arg\max_{a \in A} IG(m, a)$$

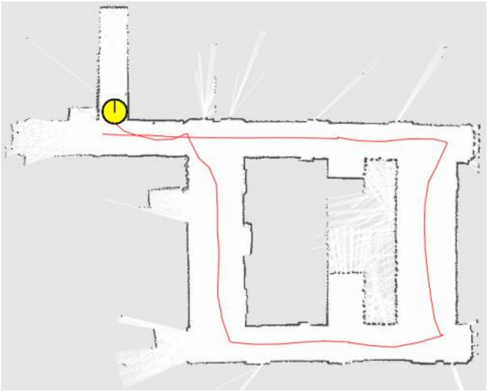
- This quantity is not known! Reason about potential measurements

$$a^* = \arg\max_{a \in A} \int IG(m, z) p(z \mid a) \mathrm{d}z$$

Example







Exploration Costs

- So far, we did not consider the cost of executing an action (e.g., time, energy, …)

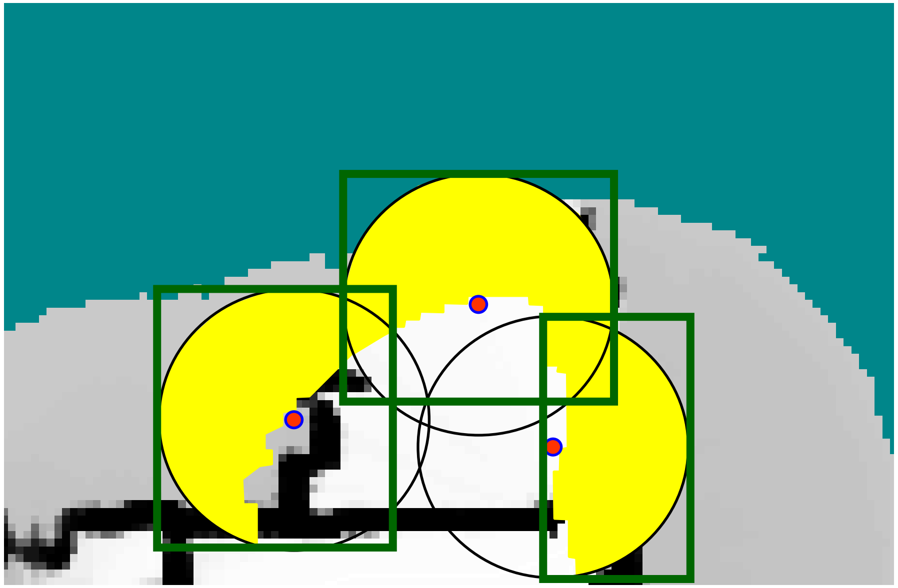
- **Utility = uncertainty reduction – cost**

- Select the action with the highest expected utility

$$a^* = \arg\max_{a \in A} IG(m, a) - \alpha \cdot E(cost(m, a))$$

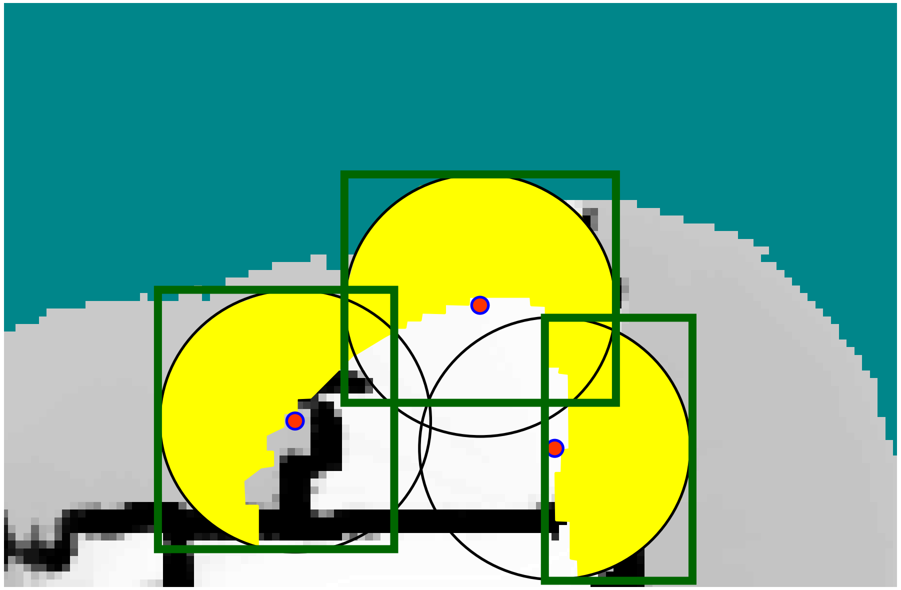
Exploration

- For each location <x,y>
  - Estimate the number of cells robot can sense (e.g., simulate laser beams using current map)
  - Estimate the cost of getting there



Exploration

- **Greedy strategy:** Select the candidate location with the highest utility, then repeat…



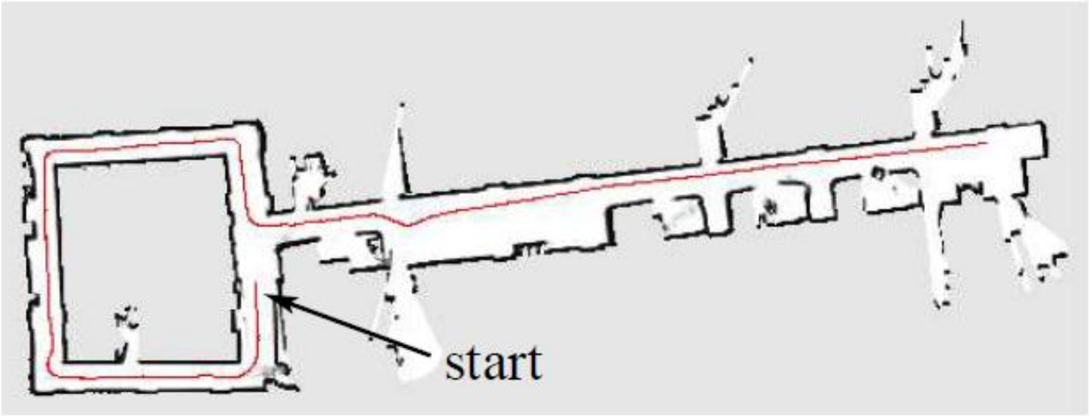
Exploration Actions

- So far, we only considered reduction in map uncertainty
- In general, there are many sources of uncertainty that can be reduced by exploration
  - Map uncertainty (visit unexplored areas)
  - Trajectory uncertainty (loop closing)
  - Localization uncertainty (active re-localization by re-visiting known locations)

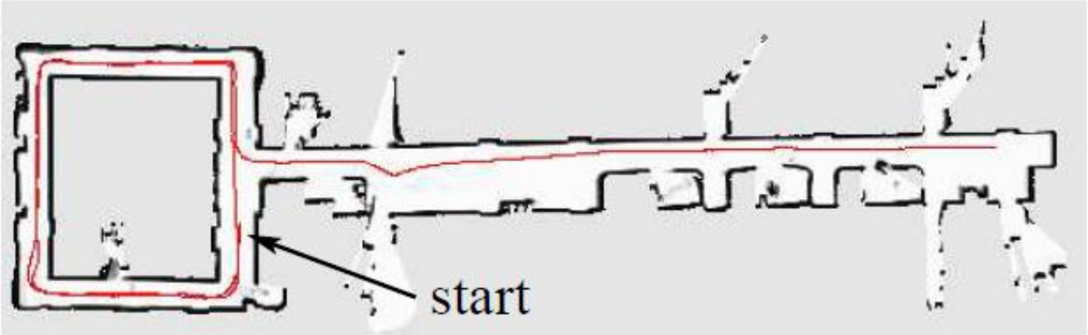
Example: Active Loop Closing

[Stachniss et al., 2005]

- Reduce map uncertainty

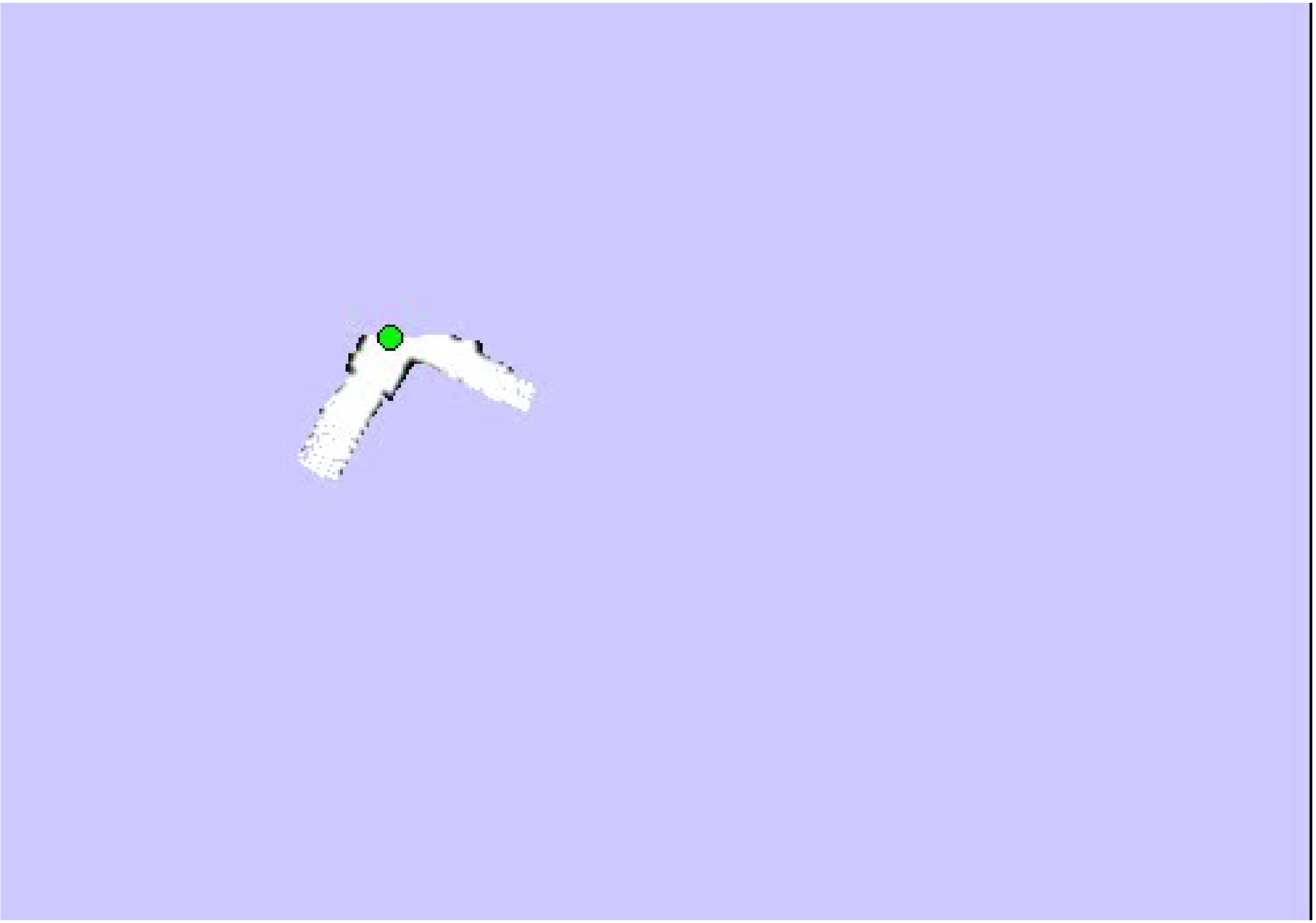


- Reduce map + path uncertainty



Example: Active Loop Closing

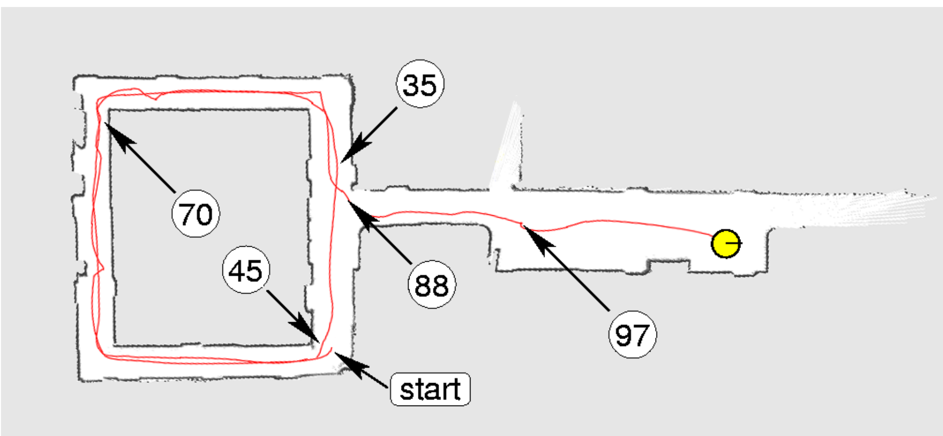
[Stachniss et al., 2005]

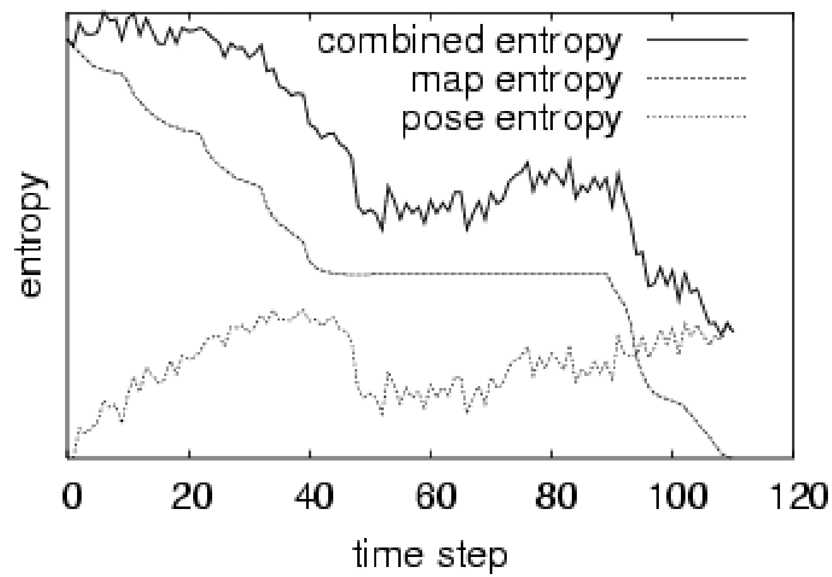


Example: Active Loop Closing

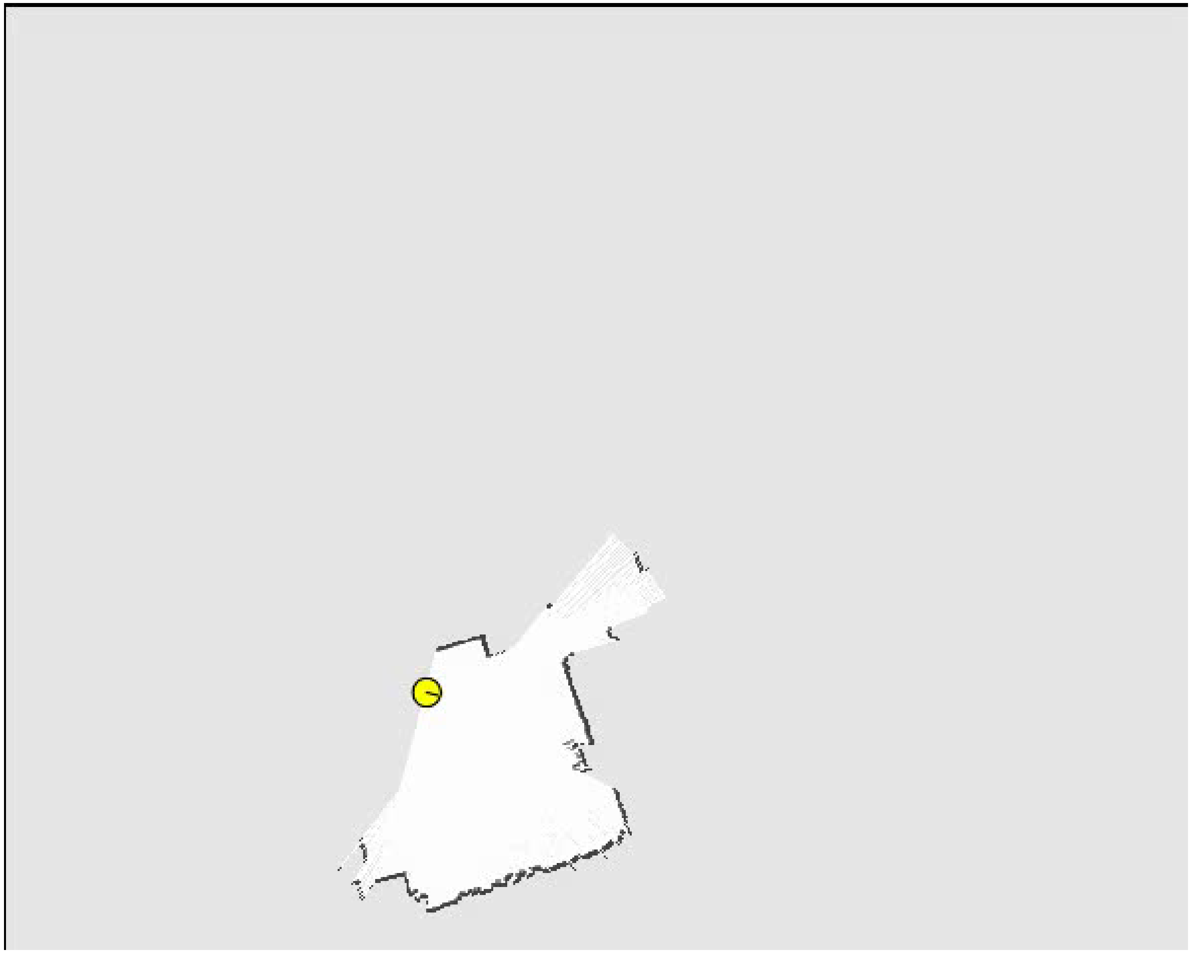
[Stachniss et al., 2005]

- Entropy evolution

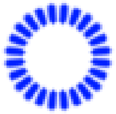




Example: Reduce uncertainty in map, path, and pose [Stachniss et al., 2005]

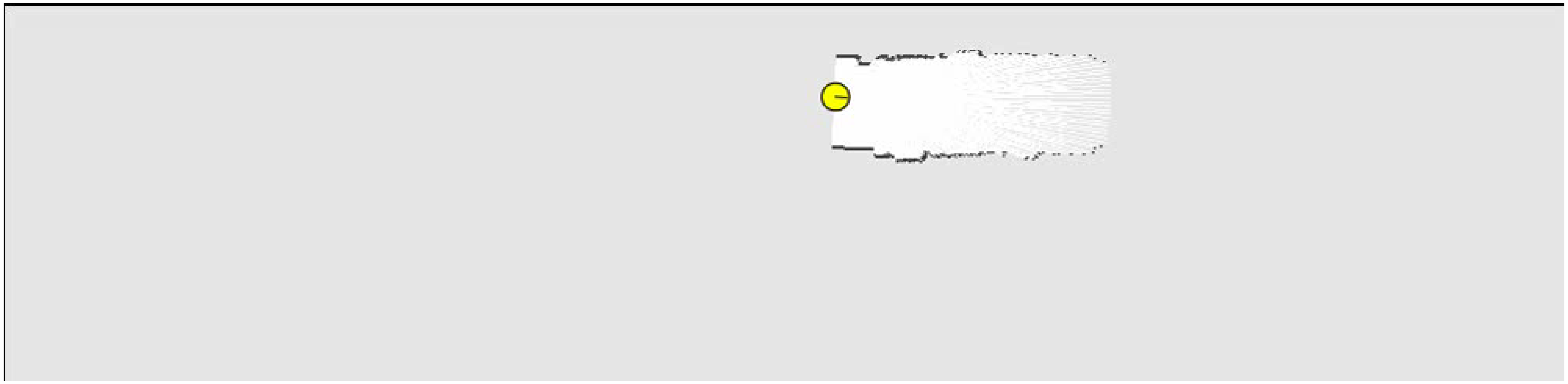


Selected target location



Corridor Exploration

[Stachniss et al., 2005]

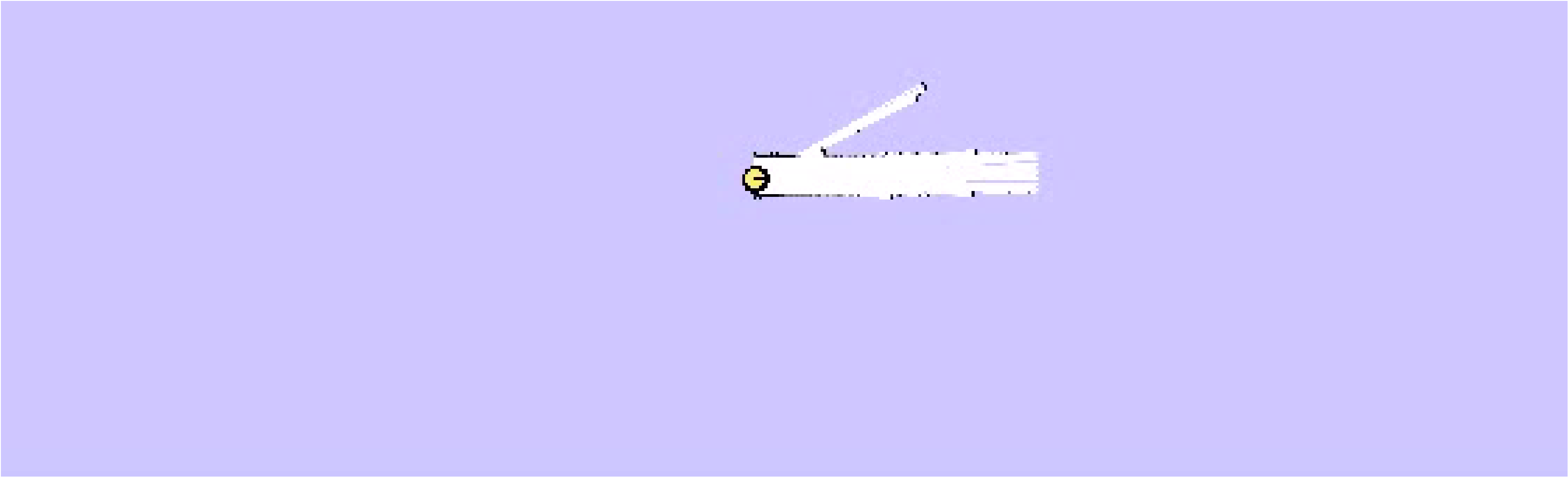


- The decision-theoretic approach leads to **intuitive behaviors:** "re-localize before getting lost"
- Some animals show a similar behavior (dogs marooned in the tundra of north Russia)

Multi-Robot Exploration

Given: Team of robots with communication

Goal: Explore the environment as fast as possible



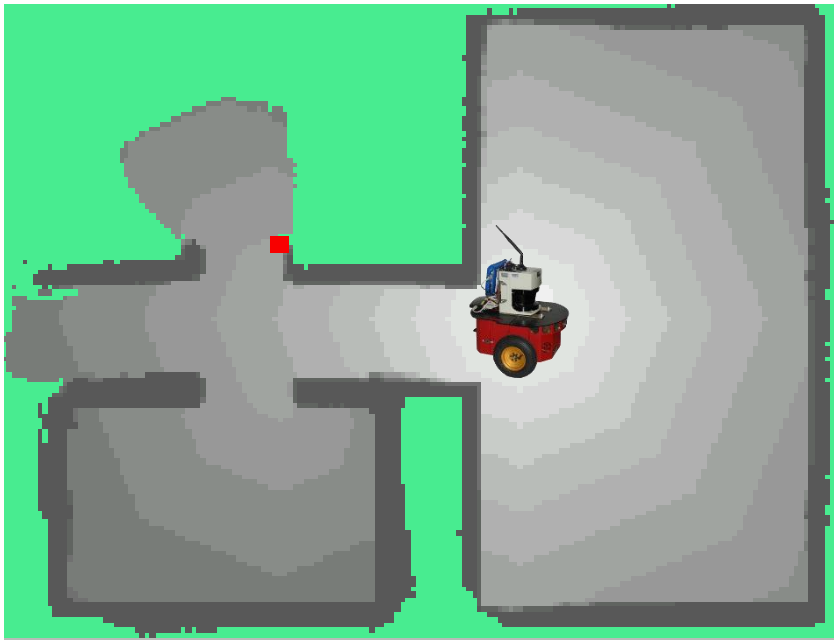
[Wurm et al., IROS 2011]

Complexity

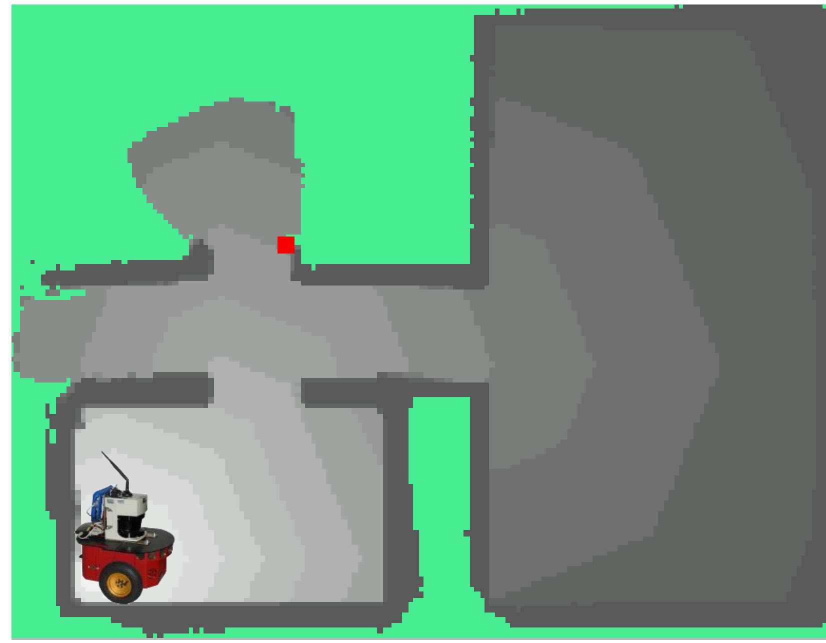
- Single-robot exploration in known, graph-like environments is in general **NP-hard**
- Proof: Reduce traveling salesman problem to exploration
- Complexity of multi-robot exploration is **exponential** in the number of robots

Motivation: Why Coordinate?

Robot 1



Robot 2



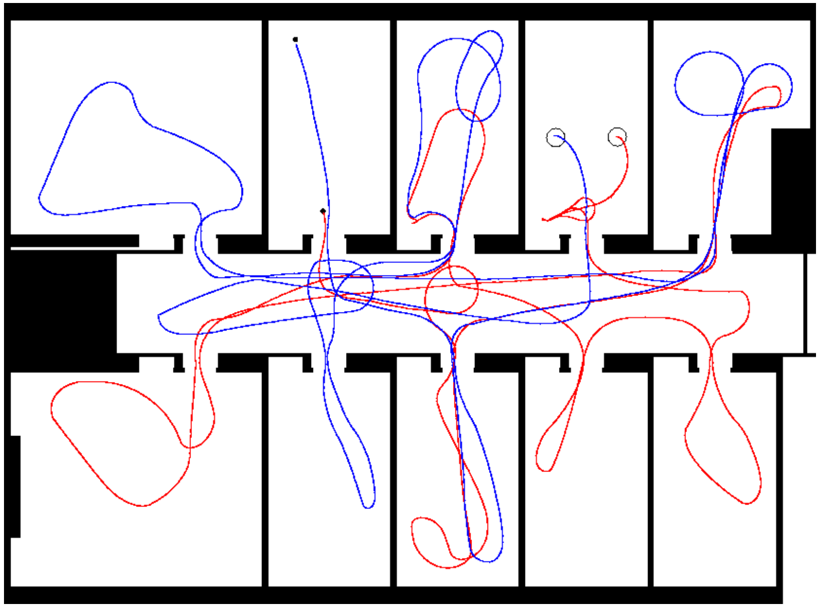
- Without coordination, two robots might choose the same exploration frontier

Levels of Coordination

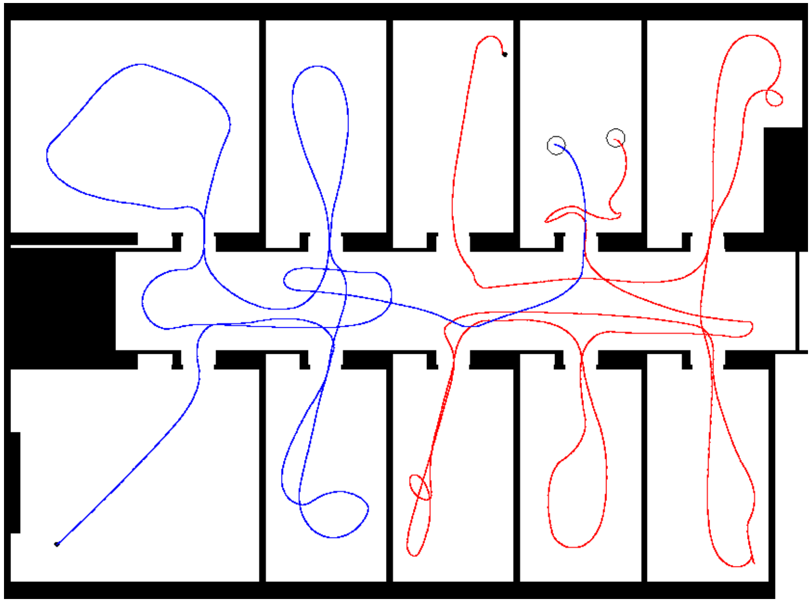
1. **No exchange of information**
2. **Implicit coordination**: Sharing a joint map
   - Communication of the individual maps and poses
   - Central mapping system
3. **Explicit coordination:** Determine better target locations to distribute the robots
   - Central planner for target point assignment
   - Minimize expected path cost / information gain / …

Typical Trajectories

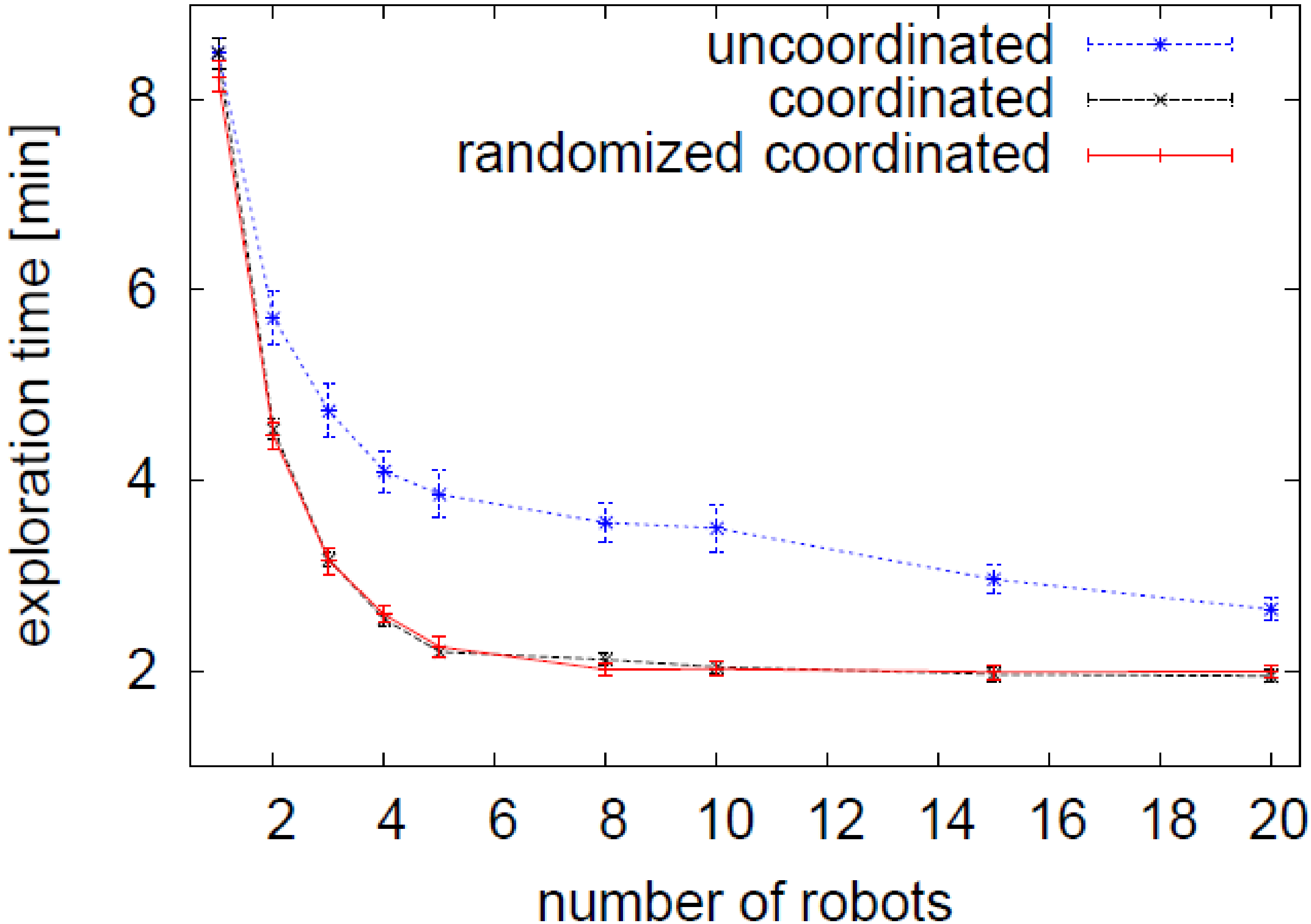
Implicit coordination:



Explicit coordination:



Exploration Time
[Stachniss et al., 2006]



Coordination Algorithm

In each time step:

- Determine set of exploration targets

$$S = \{s_1, \ldots, s_n\}$$

- Compute for each robot i and each target j the expected cost/utility C_{ij}
- Assign robots to targets using the **Hungarian algorithm**

Hungarian Algorithm

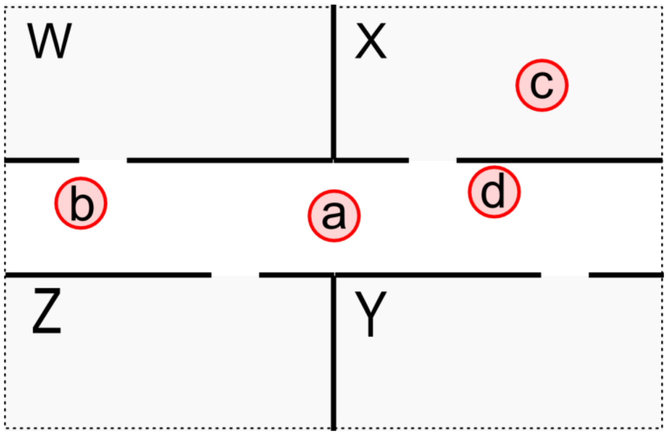
[Kuhn, 1955]

- Combinatorial optimization algorithm
- Solves the assignment problem in polynomial time $O(n^3)$
- General idea: Algorithm modifies the cost matrix until there is zero cost assignment

Hungarian Algorithm: Example

targets

robots	W	X	Y	Z
a	3	2	3	2
b	2	5	6	3
c	7	1	3	5
d	6	2	3	5



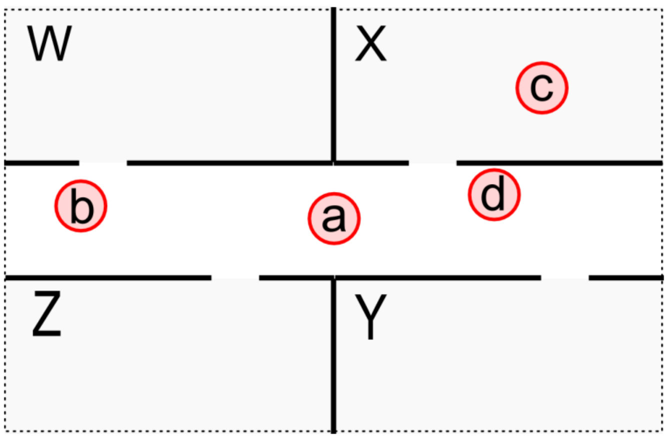
1. Compute the cost matrix (non-negative)

Hungarian Algorithm: Example

targets

robots

	W	X	Y	Z
a	3	2	3	2
b	2	5	6	3
c	7	1	3	5
d	6	2	3	5

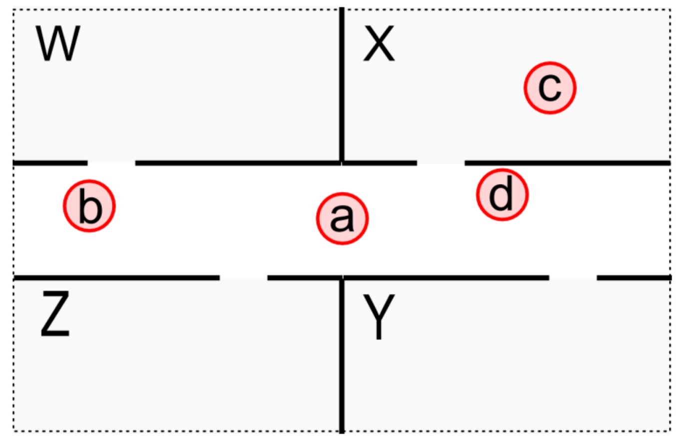


2. Find minimum element in each row

Hungarian Algorithm: Example

targets

robots	W	X	Y	Z	
a	3	2	3	2	2
b	2	5	6	3	2
c	7	1	3	5	1
d	6	2	3	5	2



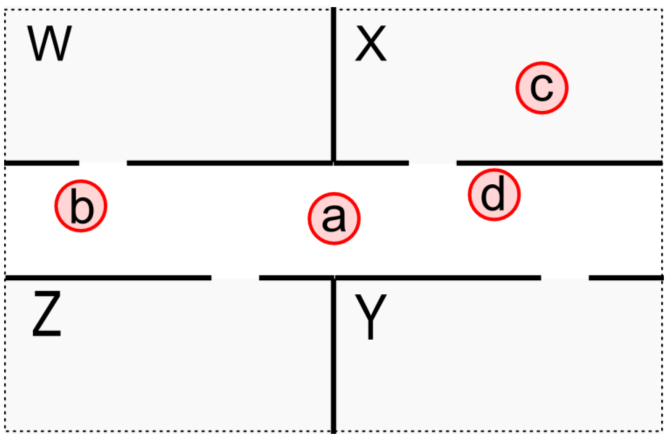
3. Subtract minimum from each row element

Hungarian Algorithm: Example

targets

robots

	W	X	Y	Z
a	1	0	1	0
b	0	3	4	1
c	6	0	2	4
d	4	0	1	3
	0	0	1	0

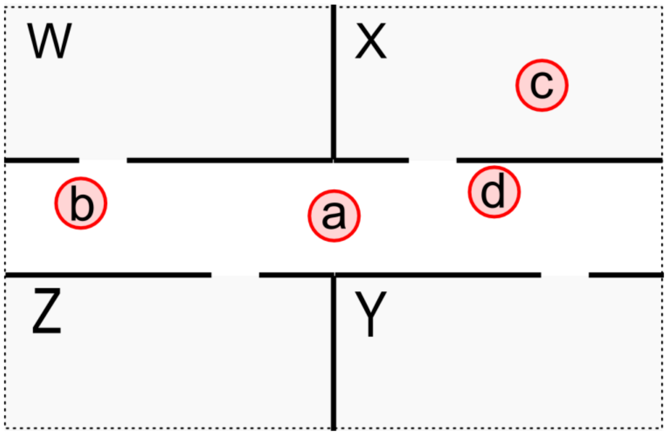


4. Find minimum element in each column

Hungarian Algorithm: Example

targets

robots	W	X	Y	Z
a	1	0	0	0
b	0	3	3	1
c	6	0	1	4
d	4	0	0	3
	0	0	1	0



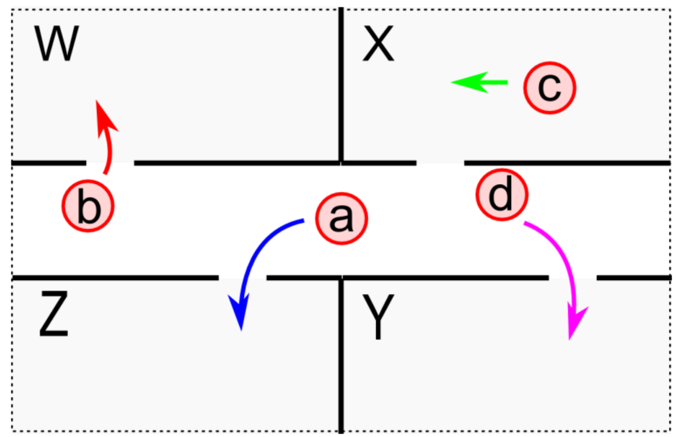
5. Subtract minimum from each column element

Hungarian Algorithm: Example

targets

robots

	W	X	Y	Z
a	1	0	0	$\boxed{0}$
b	$\boxed{0}$	3	3	1
c	6	$\boxed{0}$	1	4
d	4	0	$\boxed{0}$	3



6a. Assign (if possible)

Hungarian Algorithm: Example

targets

robots

	W	X	Y	Z
a	1	0	0	$\boxed{0}$
b	$\boxed{0}$	3	3	1
c	6	$\boxed{0}$	1	4
d	4	0	$\boxed{0}$	3

6b. If no assignment is possible:

- Connect all 0's by lines
- Find the minimum in all remaining elements and subtract
- Repeat step 2 – 6

Hungarian Algorithm: Example

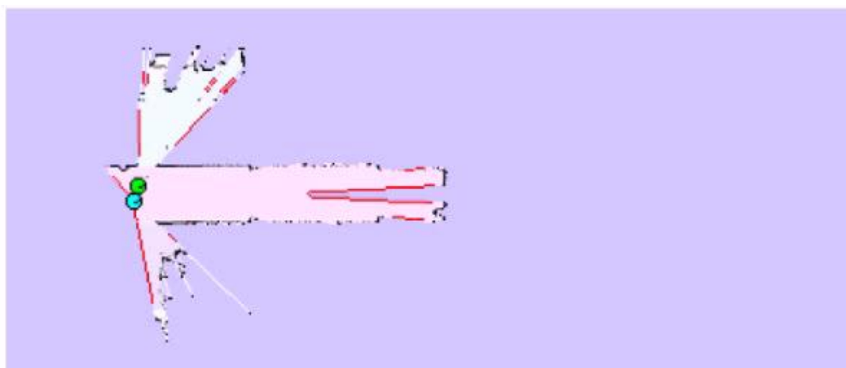
targets

robots	X	Y	X'	Y'
a	2	3	2	3
b	5	6	5	6
c	1	3	1	3
d	2	3	2	3

If there are not enough targets:
Copy targets to allow multiple assignments

Example: Segmentation-based Exploration
[Wurm et al., IROS 2008]

- Two-layer hierarchical role assignments using Hungarian algorithm (1: rooms, 2: targets in room)
- Reduces exploration time and risk of interferences



Summary: Exploration

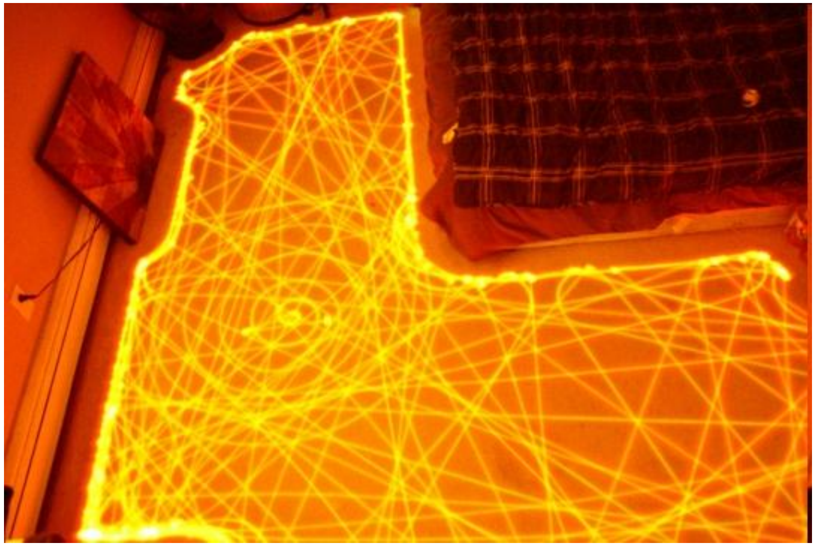
- Exploration aims at generating robot motions so that an **optimal map** is obtained
- **Coordination** reduces exploration time
- **Hungarian algorithm** efficiently solves the assignment problem (centralized, 1-step lookahead)
- Challenges (active research):
  - Limited bandwidth and **unreliable communication**
  - **Decentralized planning** and task assignment

Coverage Path Planning

- **Given:** Known environment with obstacles
- **Wanted:** The shortest trajectory that ensures complete (sensor) coverage

[images from Xu et al., ICRA 2011]

Coverage Path Planning



Coverage Path Planning: Applications

- For flying robots
  - Search and rescue
  - Area surveillance
  - Environmental inspection
  - Inspection of buildings (bridges)
- For service robots
  - Lawn mowing
  - Vacuum cleaning
- For manipulation robots
  - Painting
  - Automated farming

Coverage Path Planning

- What is a good coverage strategy?
- What would be a good cost function?

Coverage Path Planning

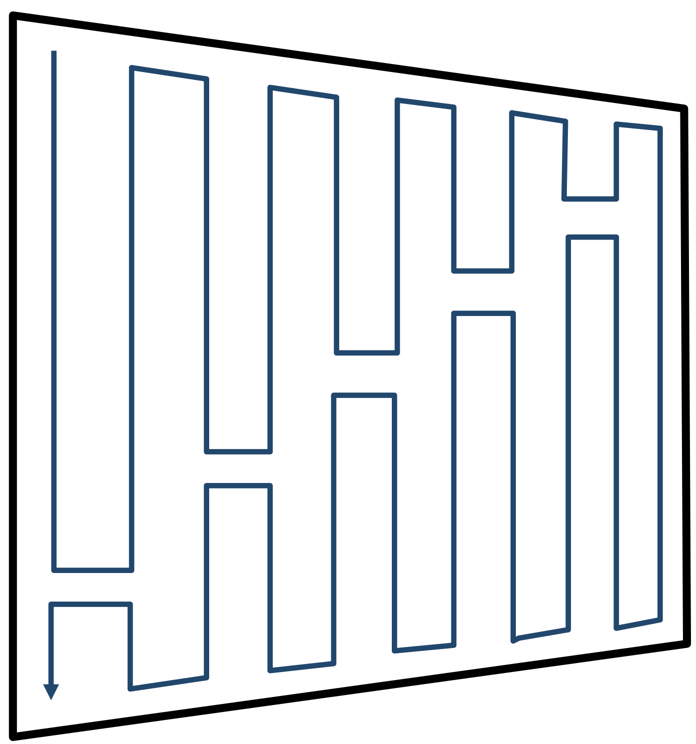
- What is a good coverage strategy?
- What would be a good cost function?
  - Amount of redundant traversals
  - Number of stops and rotations
  - Execution time
  - Energy consumption
  - Robustness
  - Probability of success
  - …

Coverage Path Planning

- Related to the traveling salesman problem (TSP):
  "Given a weighted graph, compute a path that visits every vertex once"
- In general **NP-complete**
- Many approximations exist
- Many approximate (and exact) solvers exist

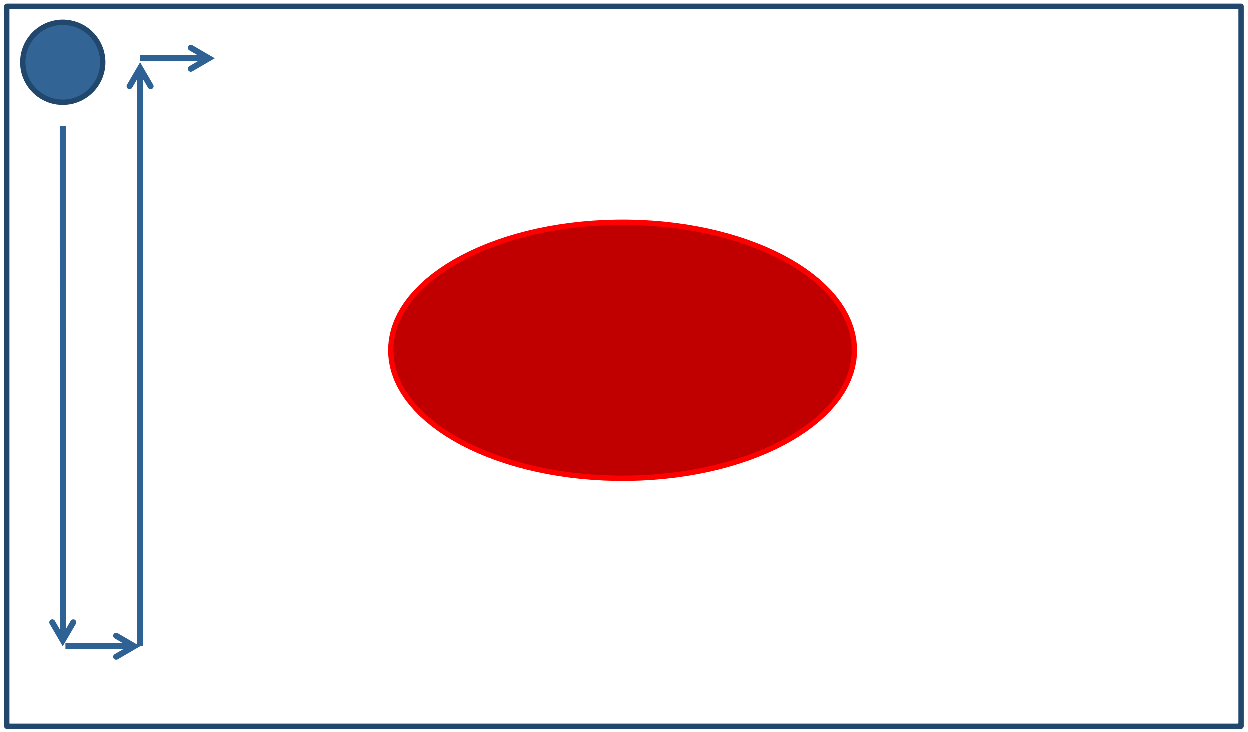
Coverage of Simple Shapes

- Approximately optimal solution often easy to compute for simple shapes (e.g., trapezoids)



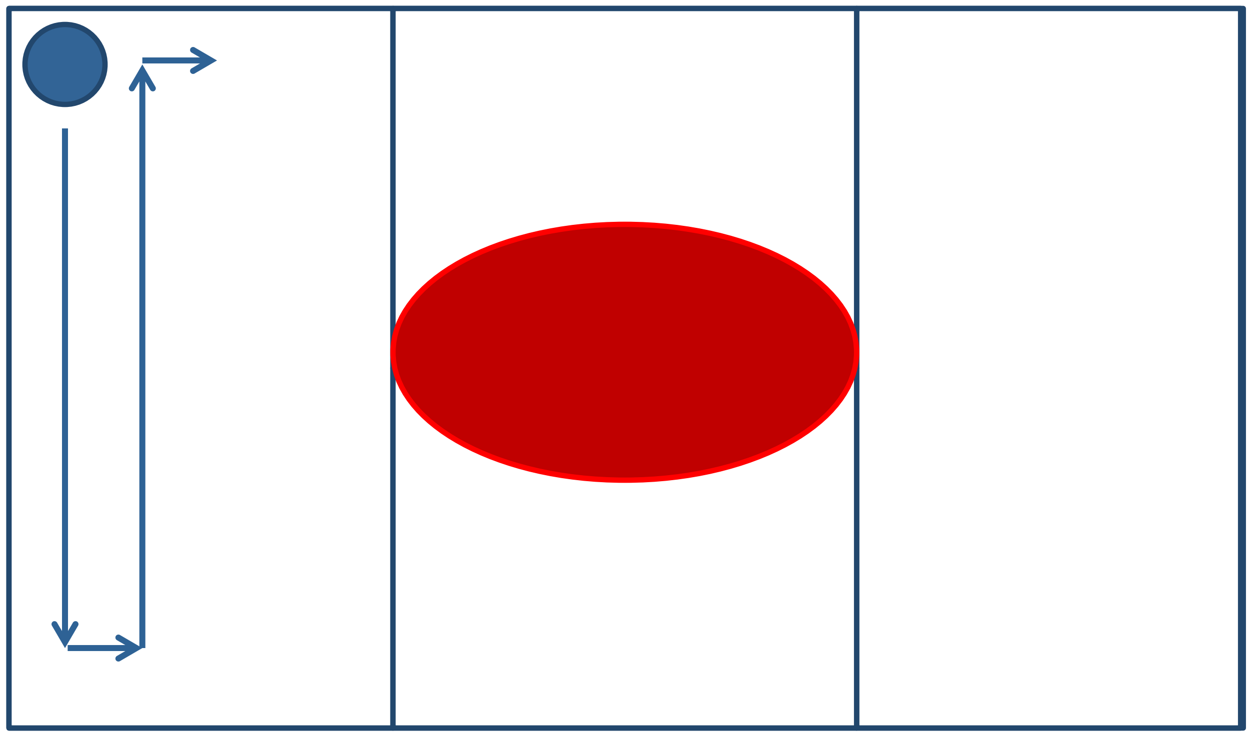
Idea

[Mannadiar and Rekleitis, ICRA 2011]



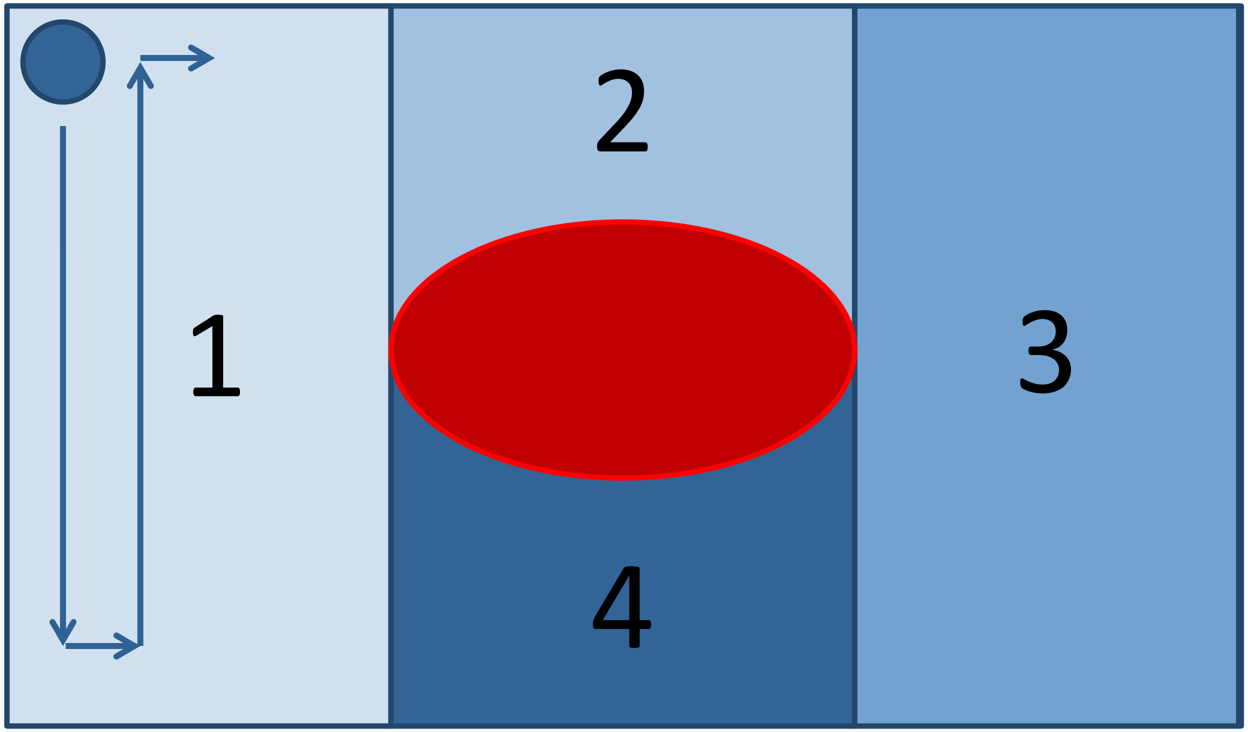
Idea

[Mannadiar and Rekleitis, ICRA 2011]



Idea

[Mannadiar and Rekleitis, ICRA 2011]



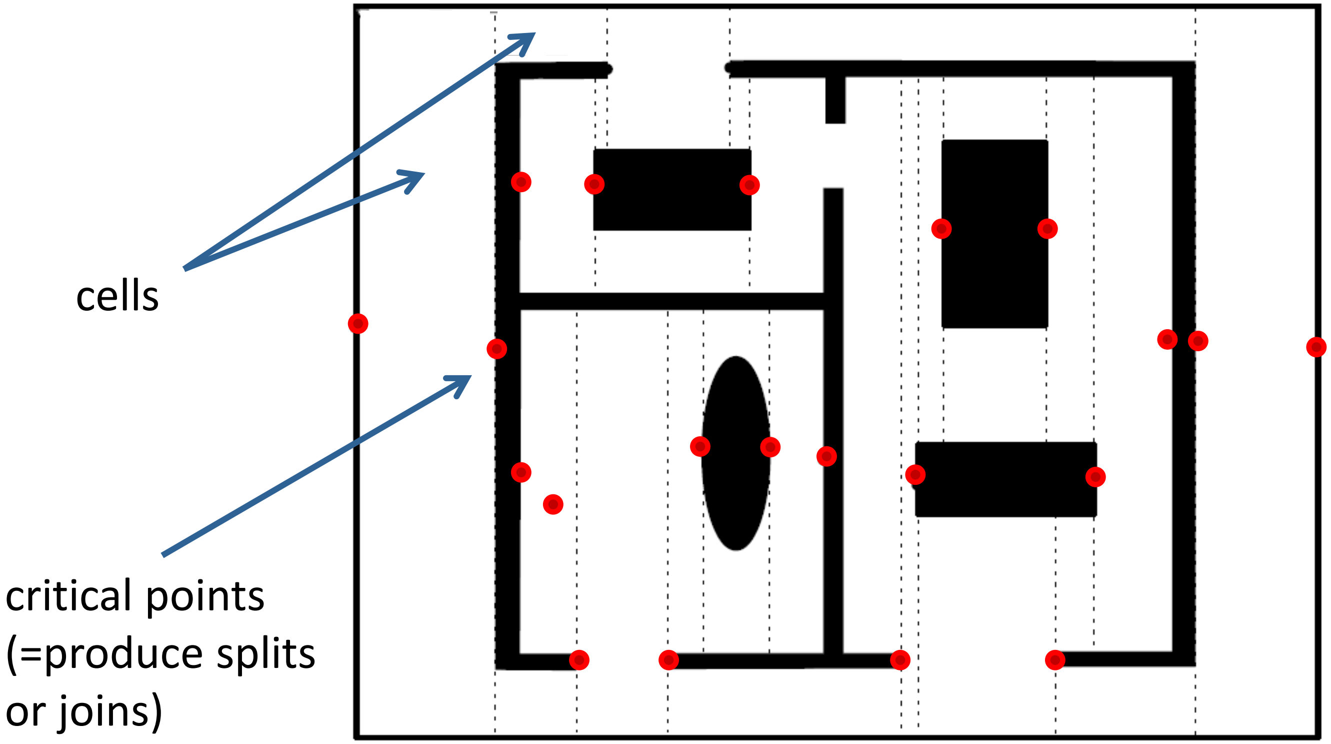
Coverage Based On Cell Decomposition
[Mannadiar and Rekleitis, ICRA 2011]

Approach:

1. Decompose map into "simple" cells
2. Compute connectivity between cells and build graph
3. Solve coverage problem on reduced graph

Step 1: Boustrophedon Cellular Decomposition [Mannadiar and Rekleitis, ICRA 2011]

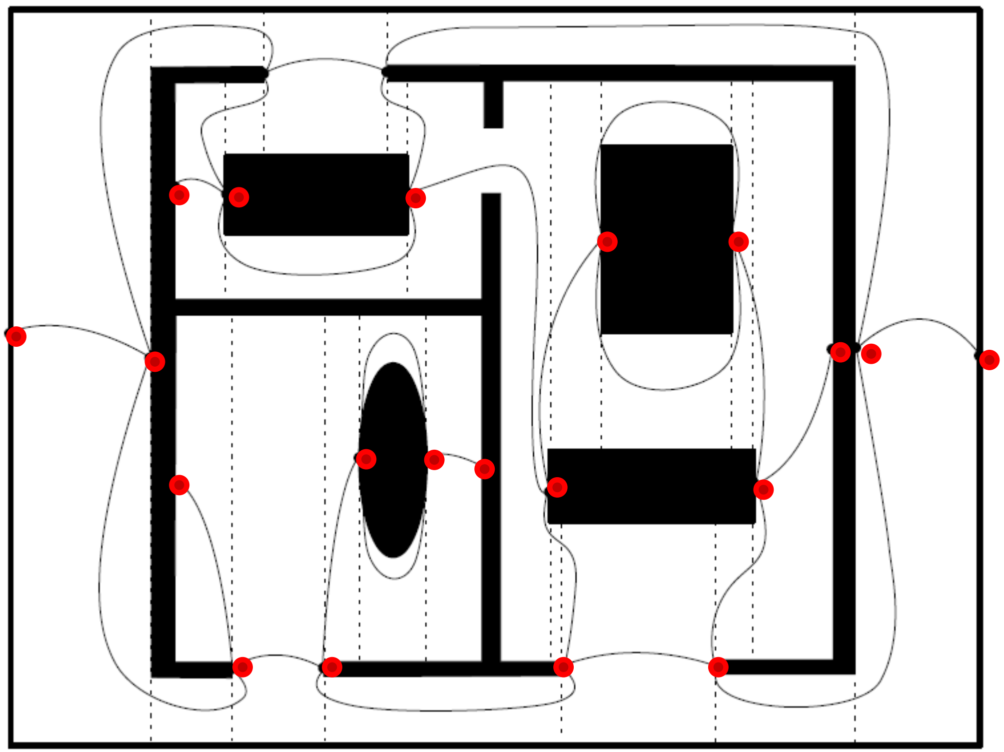
- Similar to trapezoidal decomposition
- Can be computed efficiently



Step 2: Build Reeb Graph

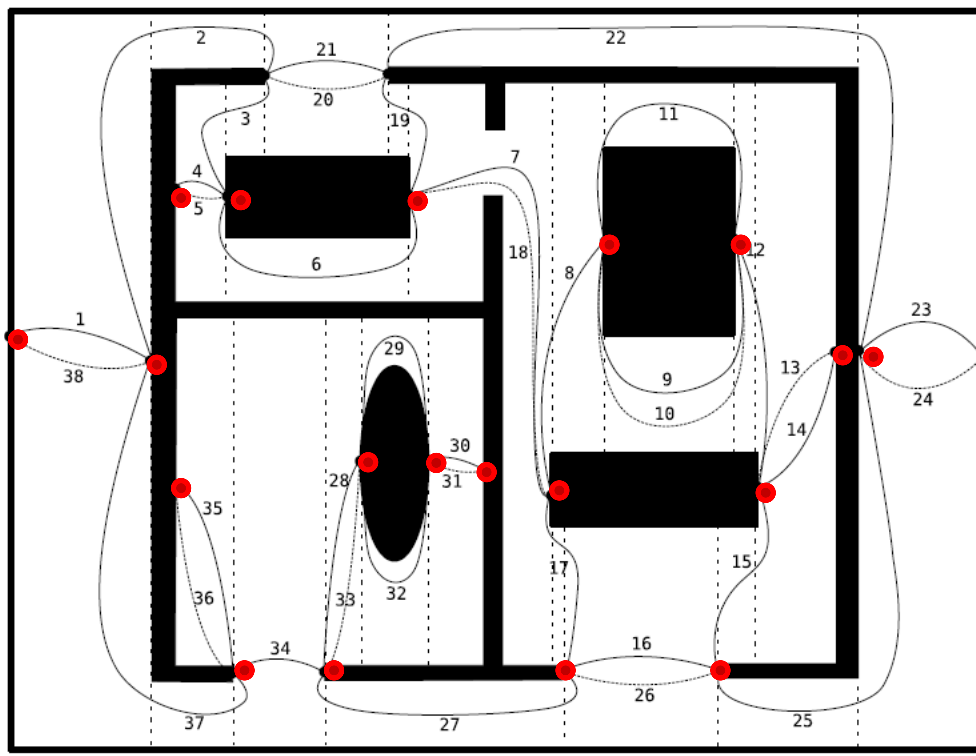
[Mannadiar and Rekleitis, ICRA 2011]

- Vertices = Critical points (that triggered the split)
- Edges = Connectivity between critical points



Step 3: Compute Euler Tour
[Mannadiar and Rekleitis, ICRA 2011]

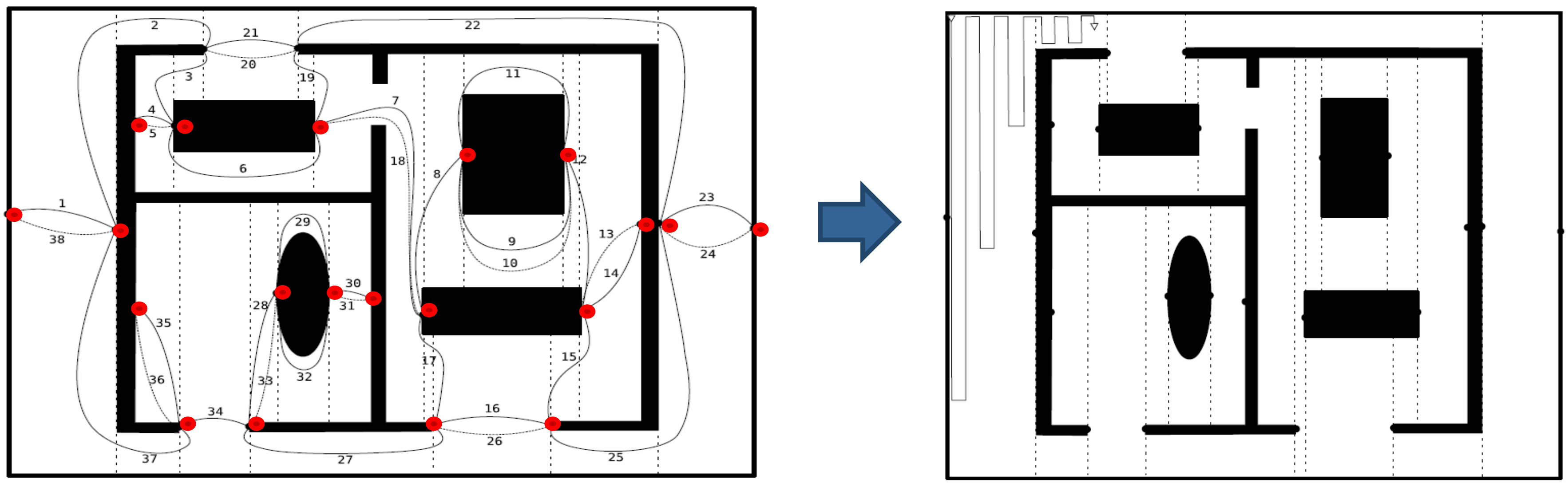
- Extend graph so that vertices have even order
- Compute Euler tour (linear time)



Resulting Coverage Plan

[Mannadiar and Rekleitis, ICRA 2011]

- Follow the Euler tour
- Use simple coverage strategy for cells
- Note: Cells are visited once or twice



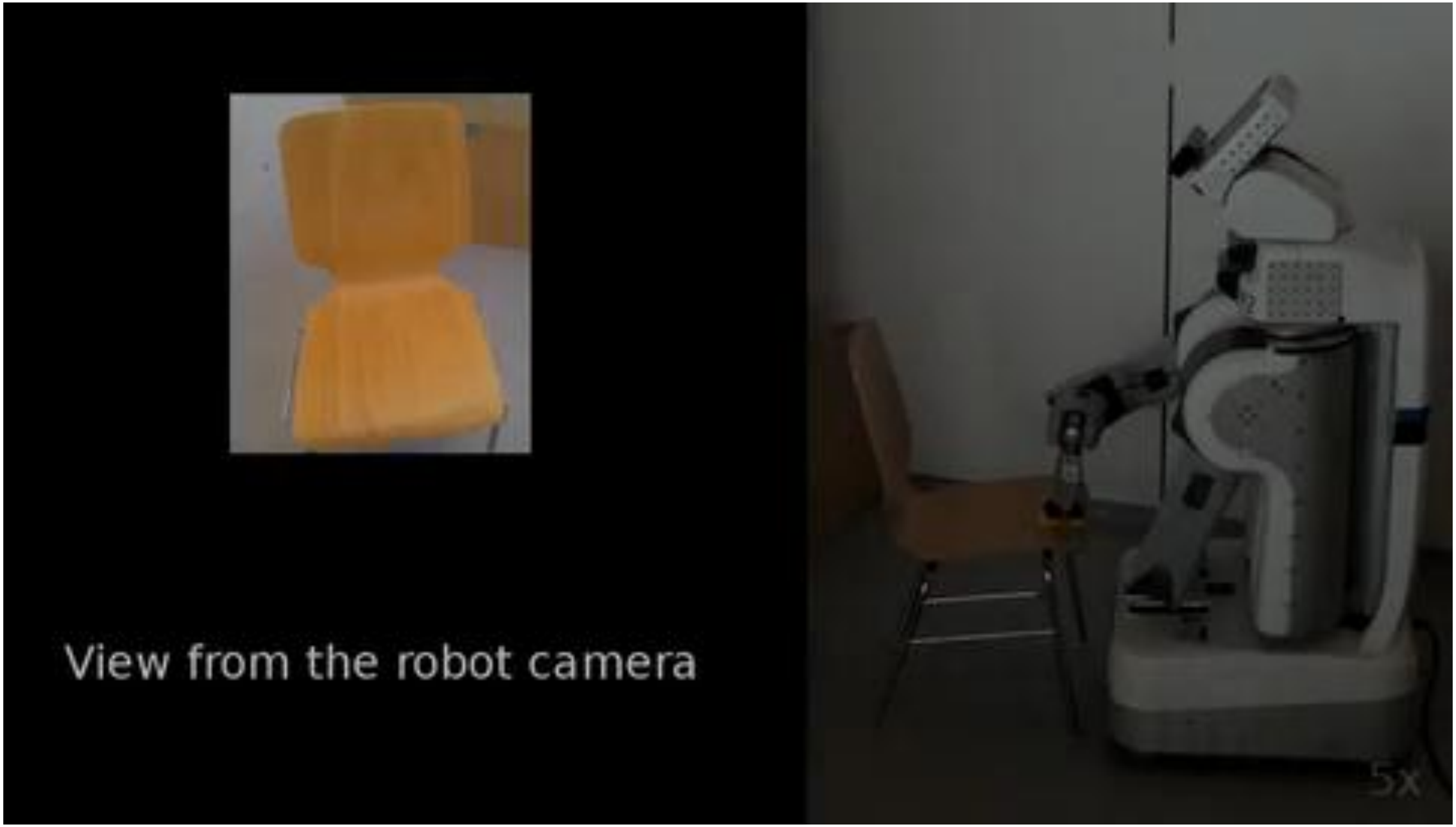
Robotic Cleaning of 3D Surfaces
[Hess et al., IROS 2012]

- **Goal:** Cover entire surface while minimizing trajectory length in configuration space

- **Approach**:
  - Discretize 3D environment into patches
  - Build a neighborhood graph
  - Formulate the problem as generalized TSP (GTSP)

Robotic Cleaning of 3D Surfaces

[Hess et al., IROS 2012]



Lessons Learned Today

- How to generate plans that are robust to uncertainty in sensing and locomotion
- How to explore an unknown environment
  - With a single robot
  - With a team of robots
- How to generate plans that fully cover known environments

Video: SFLY Final Project Demo (2012)