Motion Planning in Exploration Missions

CHRISTIAN KÖRNIG

Exploring an unknown Map

Goal: Safe and autonomous exploration of unknown Terrain

Use Cases:

- Scientific Exploration:

  Other celestial bodies, hardly accessible locations, …

- Commercial Exploration:

  Automated recurrent exploration like industrial inspection, crop mapping, …

Subproblems

Exploration combines several different fields of robotics

- Localization
- Mapping
- Path planning in already explored areas
- Determining best exploration path

Has to run in real time on the robot’s hardware!

Localization and Mapping is assumed to work

Path Planning

Determine the fastest collision-free path between $\vec{x}_{start}$ and $\vec{x}_{goal}$ in the known space V

A* Algorithm

- Split V into cells $\vec{x}_i$
- Set $\vec{x}$ to $\vec{x}_{start}$
- Until $\vec{x}$ is $\vec{x}_{goal}$:
  - Calculate cost for moving from $\vec{x}$ to neighboring cells $\vec{x}_j$
  - Calculate expected total cost for moving from $\vec{x}_j$ to $\vec{x}_{goal}$
  - Set $\vec{x}$ to neighbor with lowest expected total cost

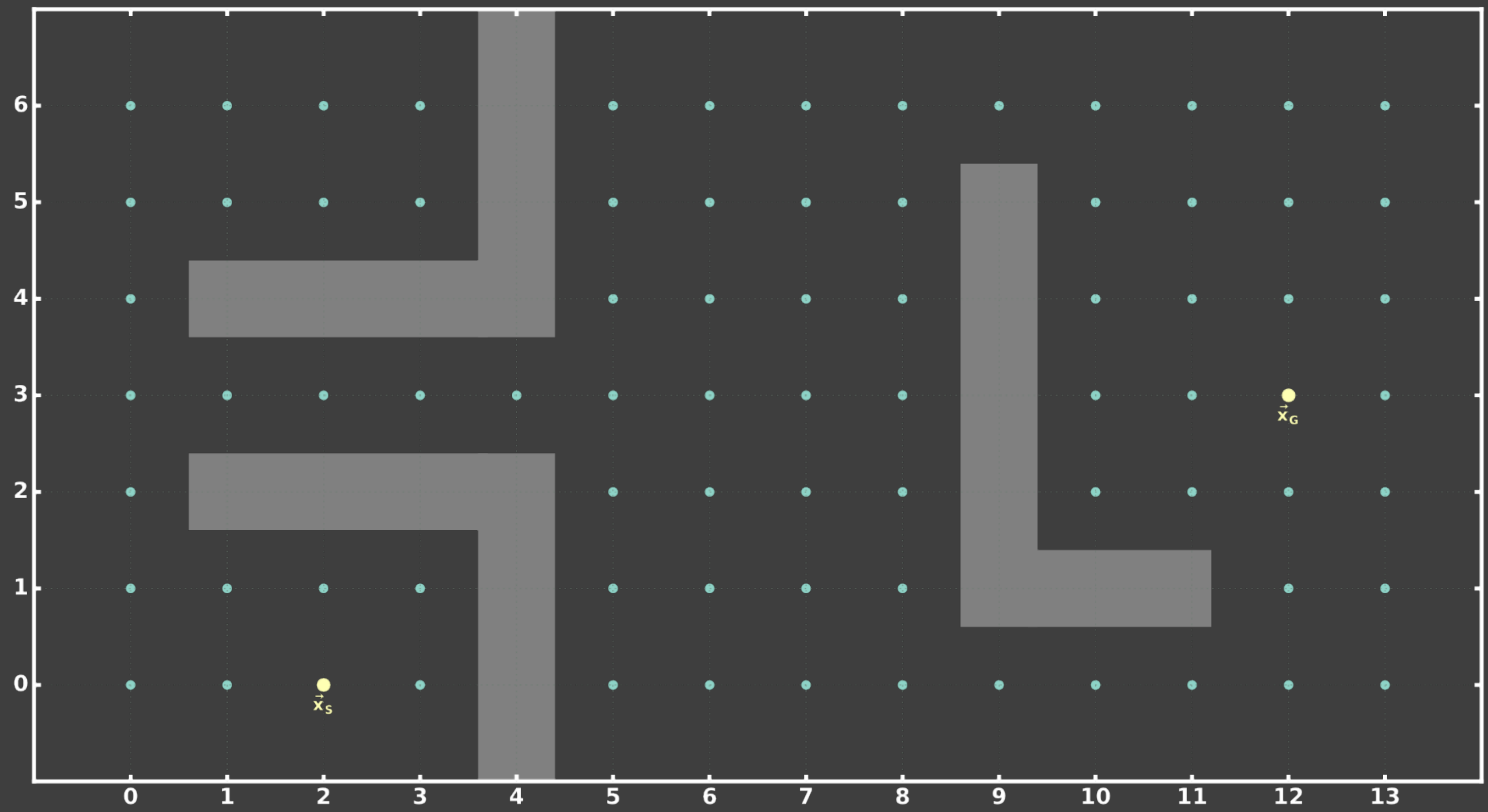
➢ Biases cell evaluation towards $\vec{x}_{goal}$

➢ Heuristic for determining this direction is needed!

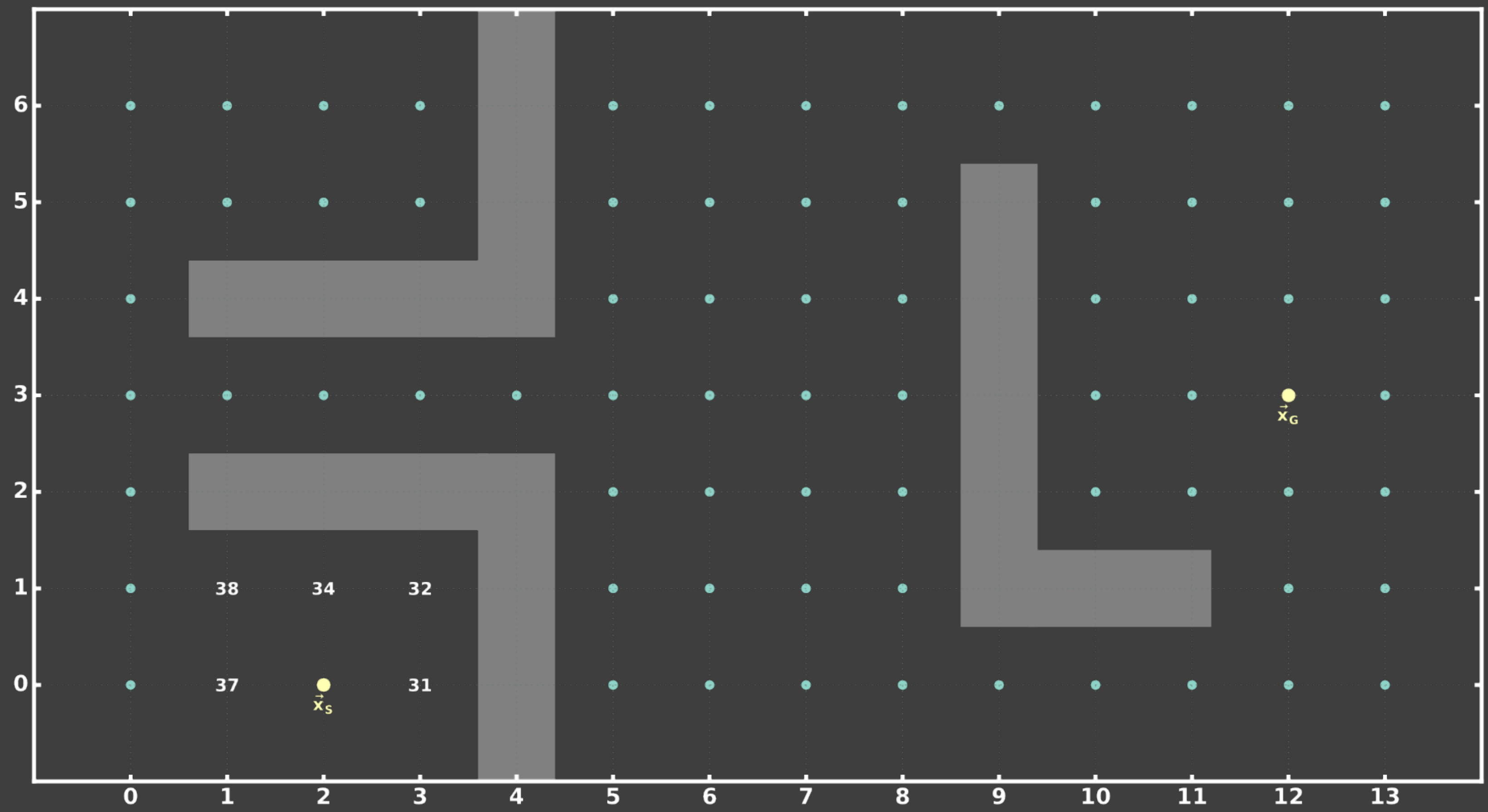
- Usually: linear distance to goal $|\vec{x}_i - \vec{x}_{goal}|$ is used

[P. Hart et. al., 1968]

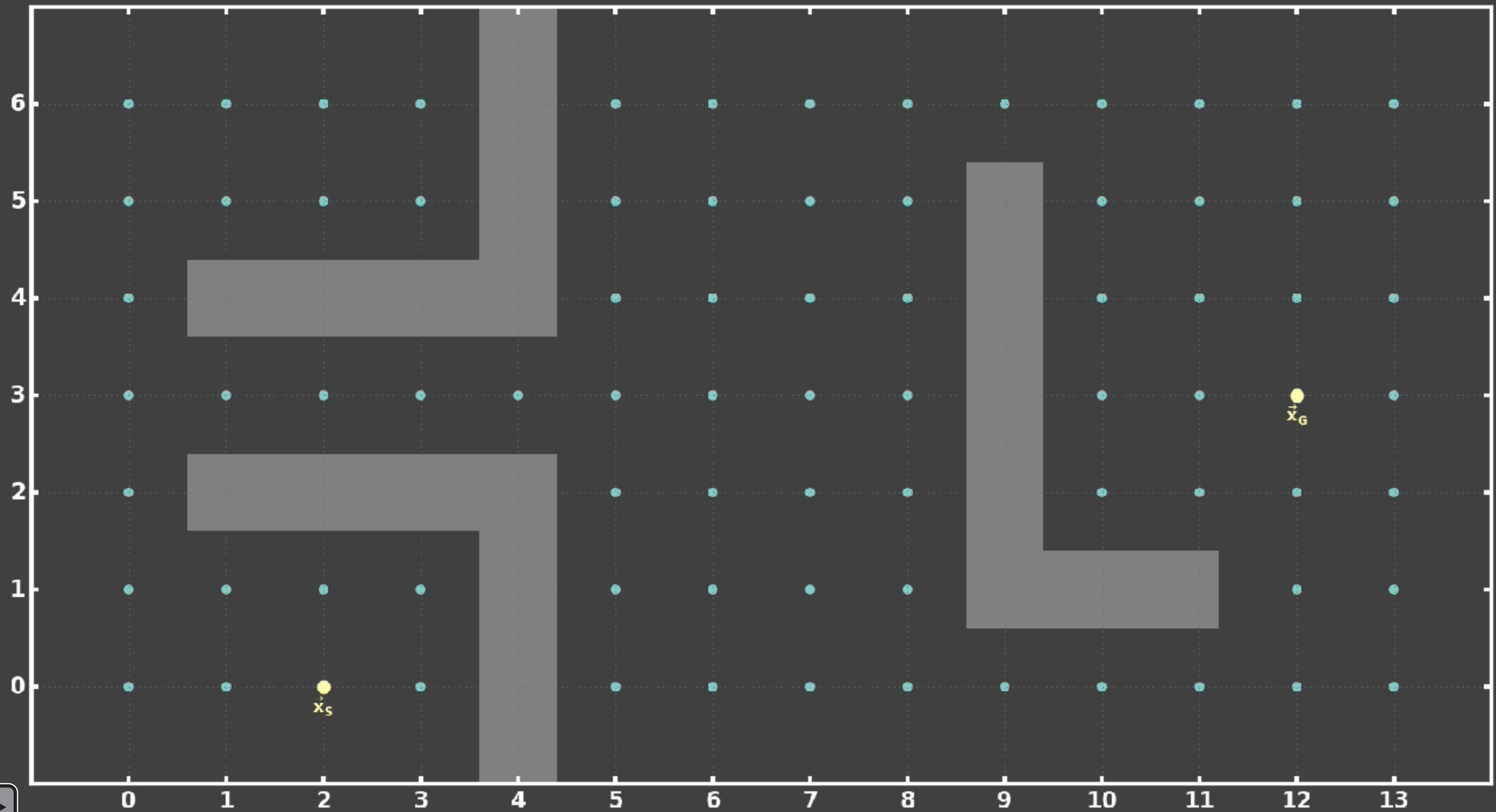
A* Algorithm



A* Algorithm



A* Algorithm

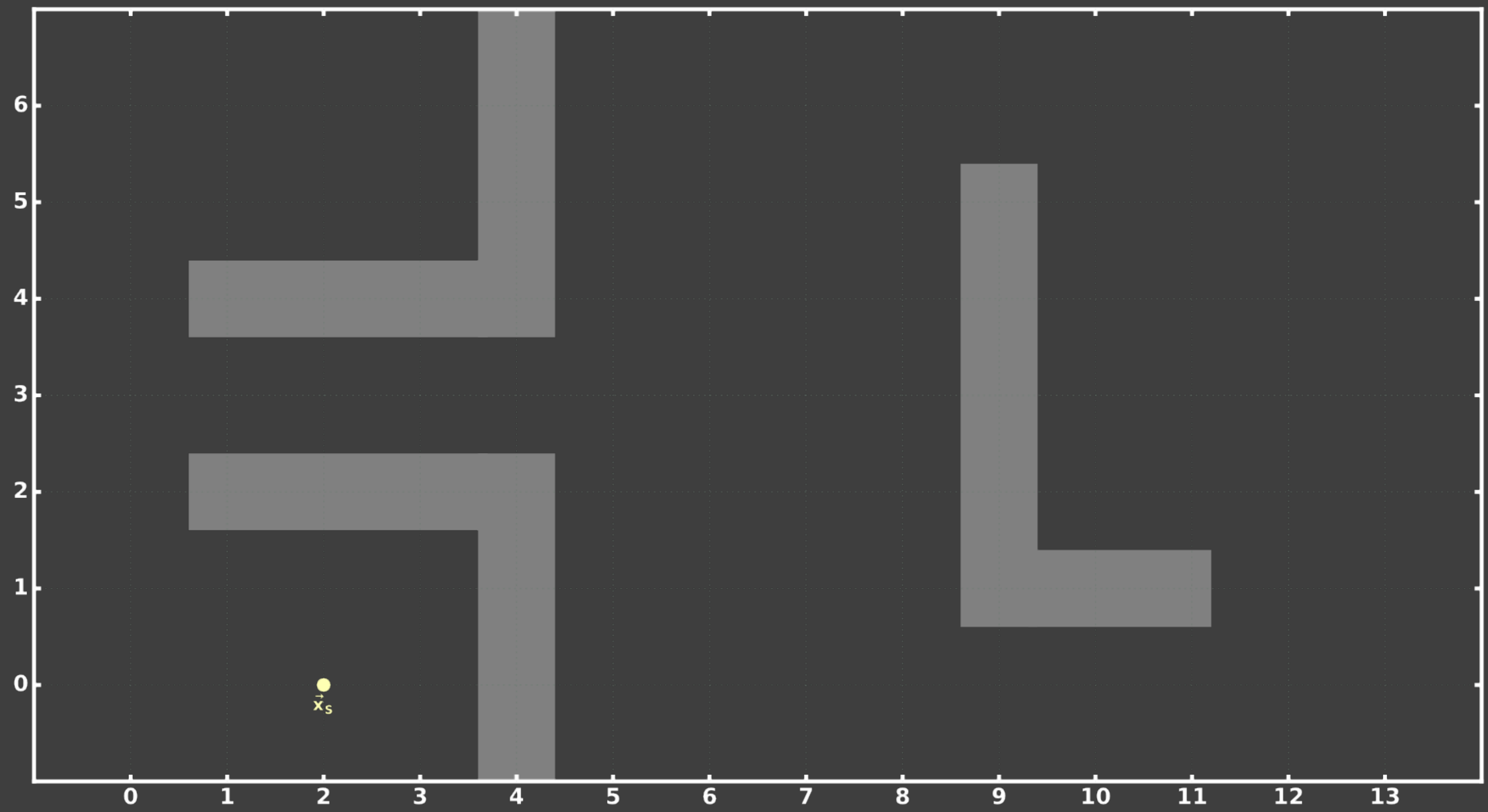


Rapidly-exploring random tree

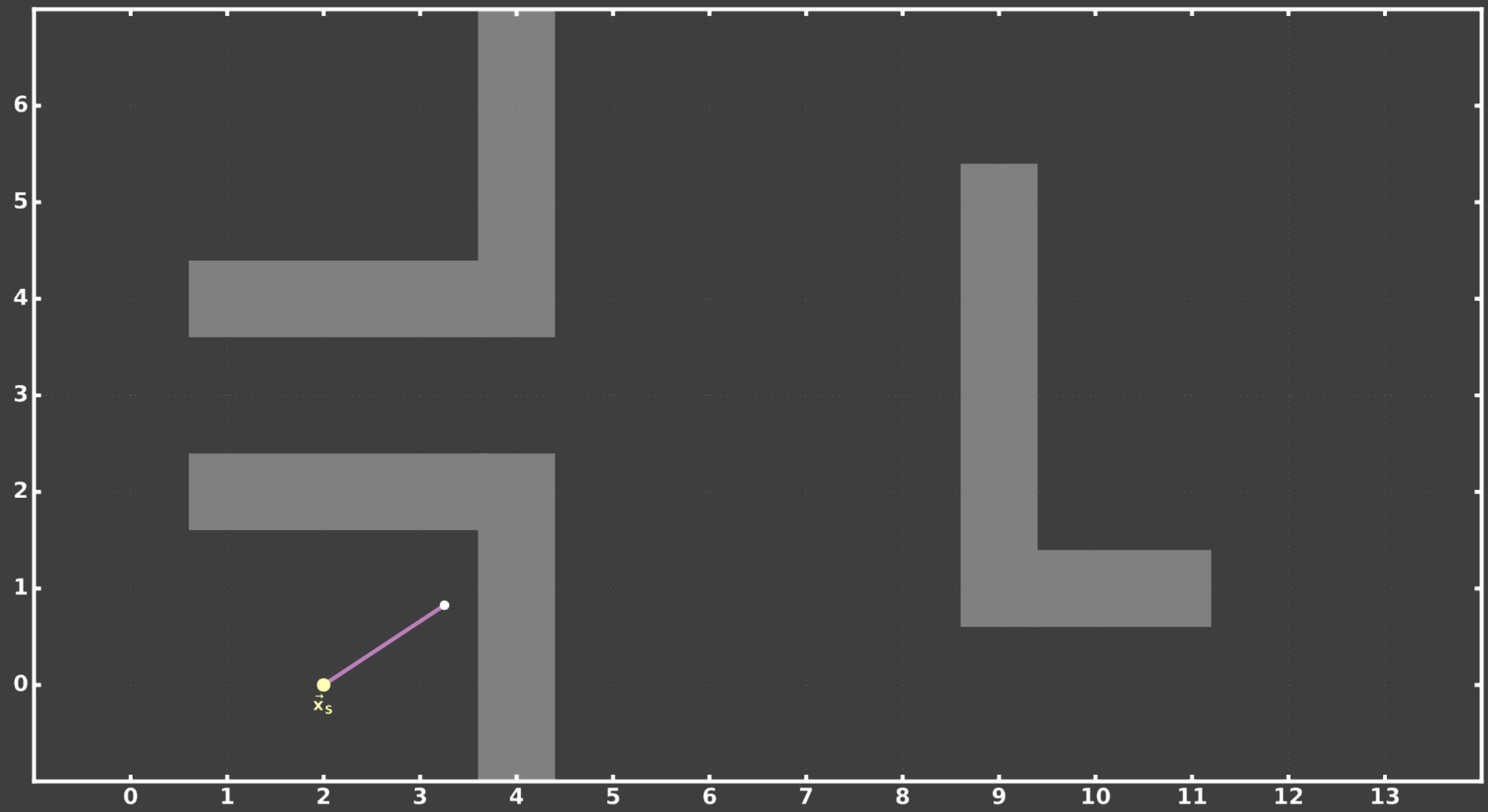
- Initialize tree T with root node $\vec{x}_{start}$
- For k points:
  - Generate random point $\vec{x}_R$
  - Find closest node $\vec{x}_C \in T$
  - Add edge from $\vec{x}_C$ towards $\vec{x}_R$ (maximum length Δx)

[S. LaValle, 1996]

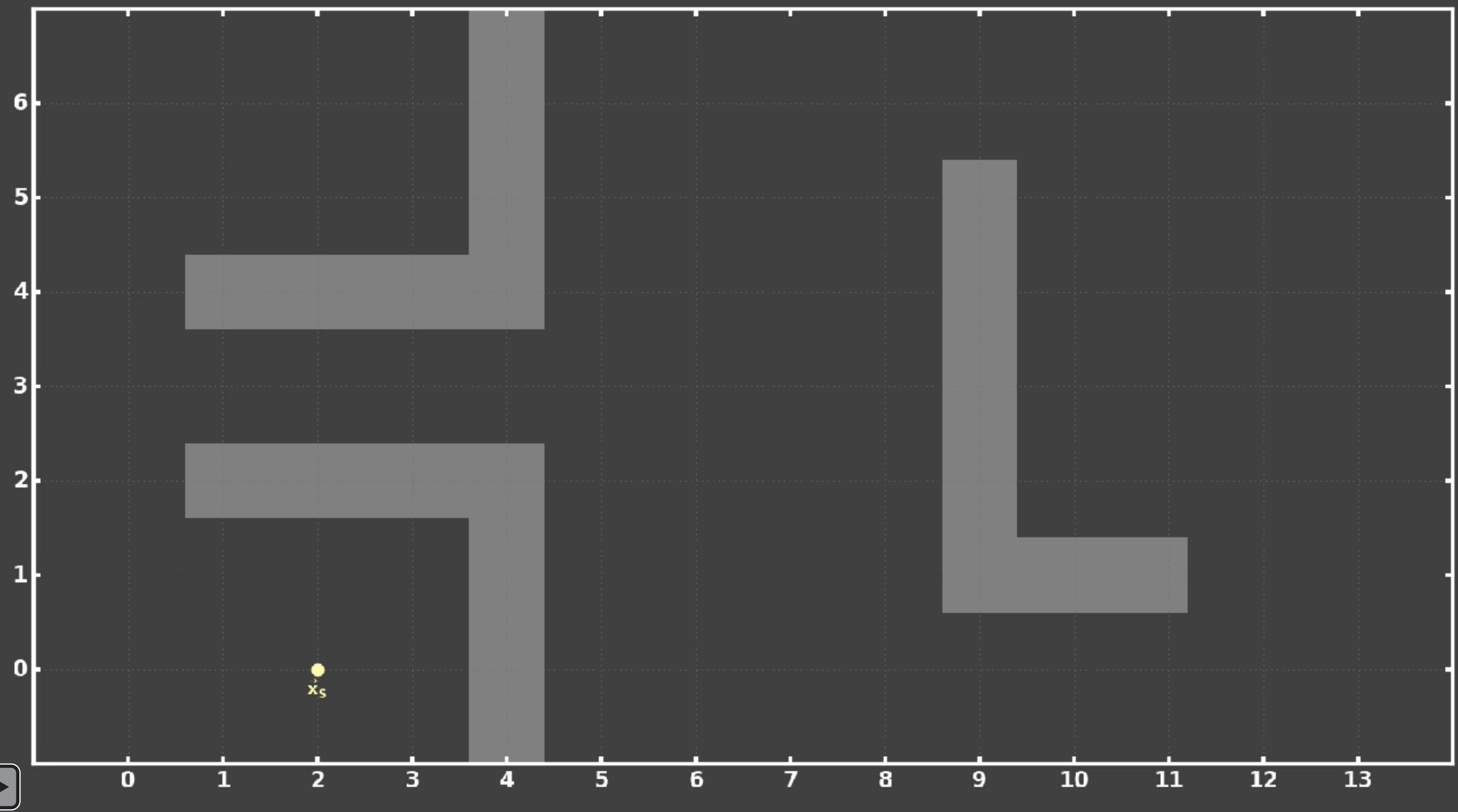
Rapidly-exploring random tree



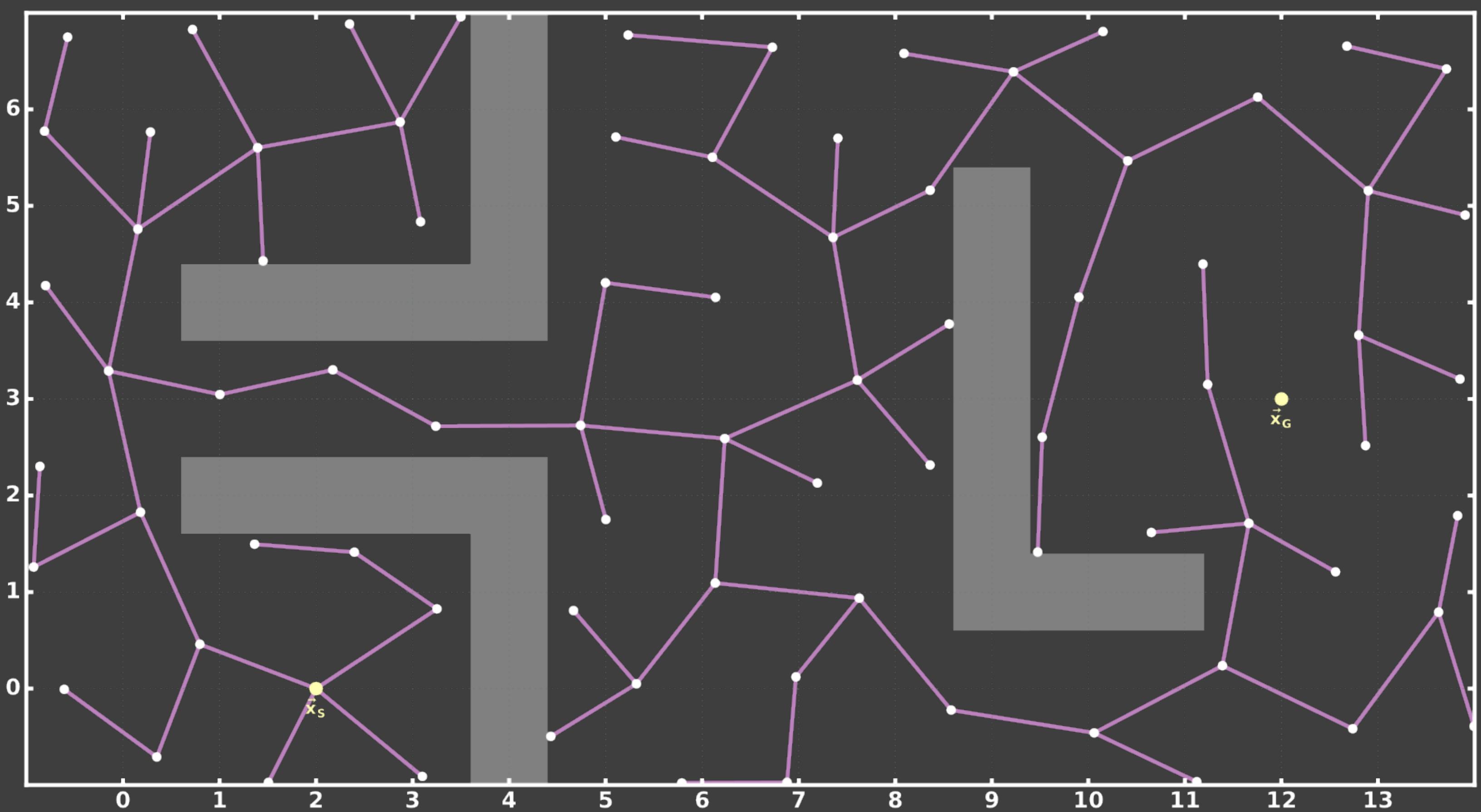
Rapidly-exploring random tree



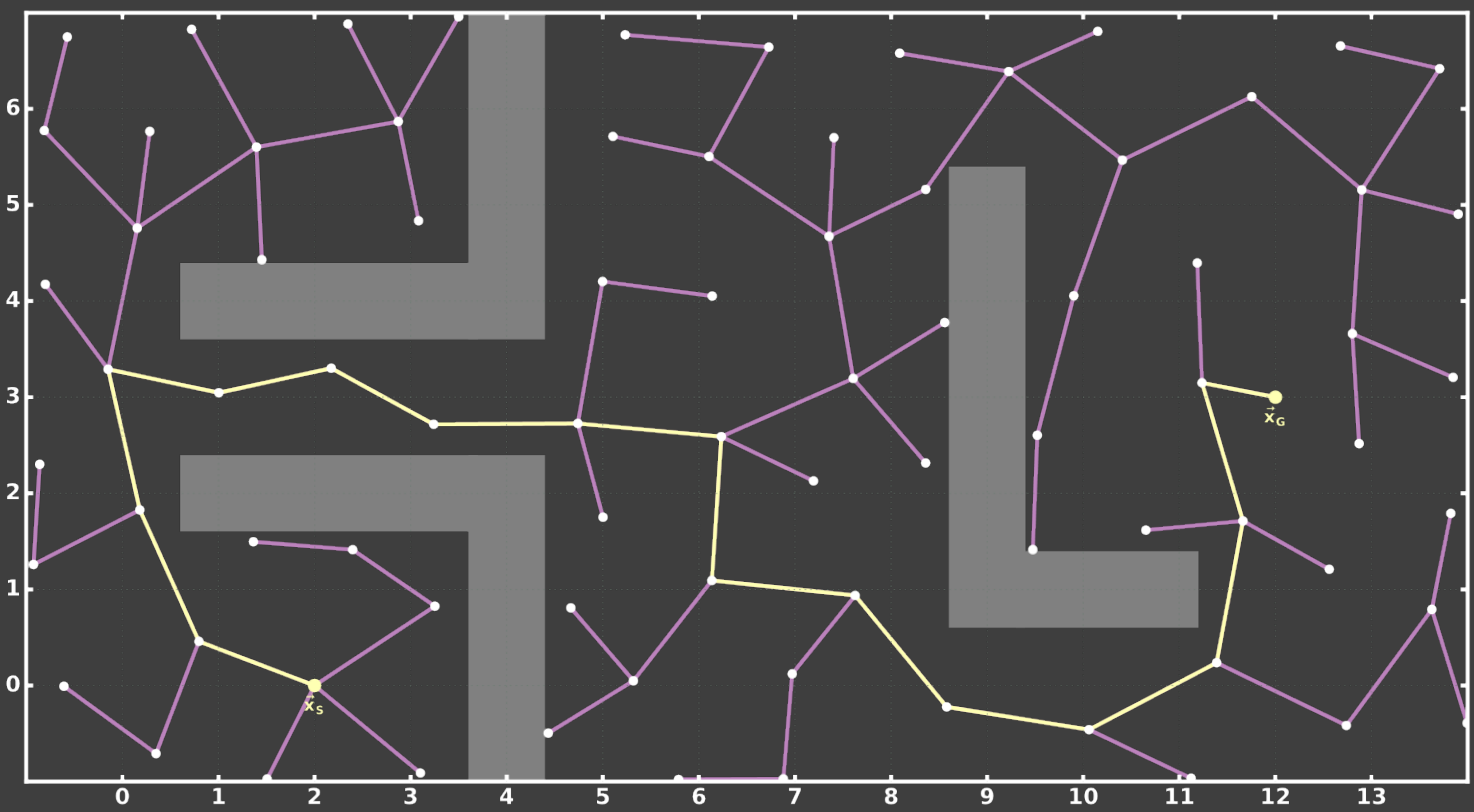
Rapidly-exploring random tree



Rapidly-exploring random tree



Rapidly-exploring random tree



Comparison A* vs RRT

A*	RRT
◦ Grid-based ◦ Builds around starting point ◦ Finds (close to) optimal path ◦ Needs good heuristic	◦ Continuous space ◦ Biased towards large unexplored regions ◦ Path cost higher by factor 1.3 – 2.0 ◦ Handles kinematic constraints and dynamics
➢ Good if optimal path in small space is needed	➢ Good for large volumes and high-dimensional problems

Exploration Planning

Determine the shortest set of paths to map the unknown space $V_{unmapped}$ while staying in the known space V_{mapped}

Determining the exploration path

Information gain

- How much additional information $G(\vec{x})$ do I gain by moving to position $\vec{x}$

$$G(\vec{x}) = \int_V d^3\vec{x}' \; Visible(\vec{x},\vec{x}') \cdot Unmapped(\vec{x}')$$

$$Visible(\vec{x},\vec{x}') \in \{0,1\}, \quad Unmapped(\vec{x}') \in \{0,1\}$$

- In applications: discretize volume into voxels $\vec{x}_i$

$$G(\vec{x}) \to \tilde{G}(\vec{x}_i) = \sum_j Visible\left(\vec{x}_i,\vec{x}_j\right) \cdot Unmapped\left(\vec{x}_j\right)$$

Next-Best-View

- Find the point $\vec{x}_{best}$ where the information gain is at maximum
- Often a cost factor is added: $\tilde{G}'(\vec{x}_i) = \tilde{G}(\vec{x}_i)\, e^{-\lambda c(\vec{x}_i)}$

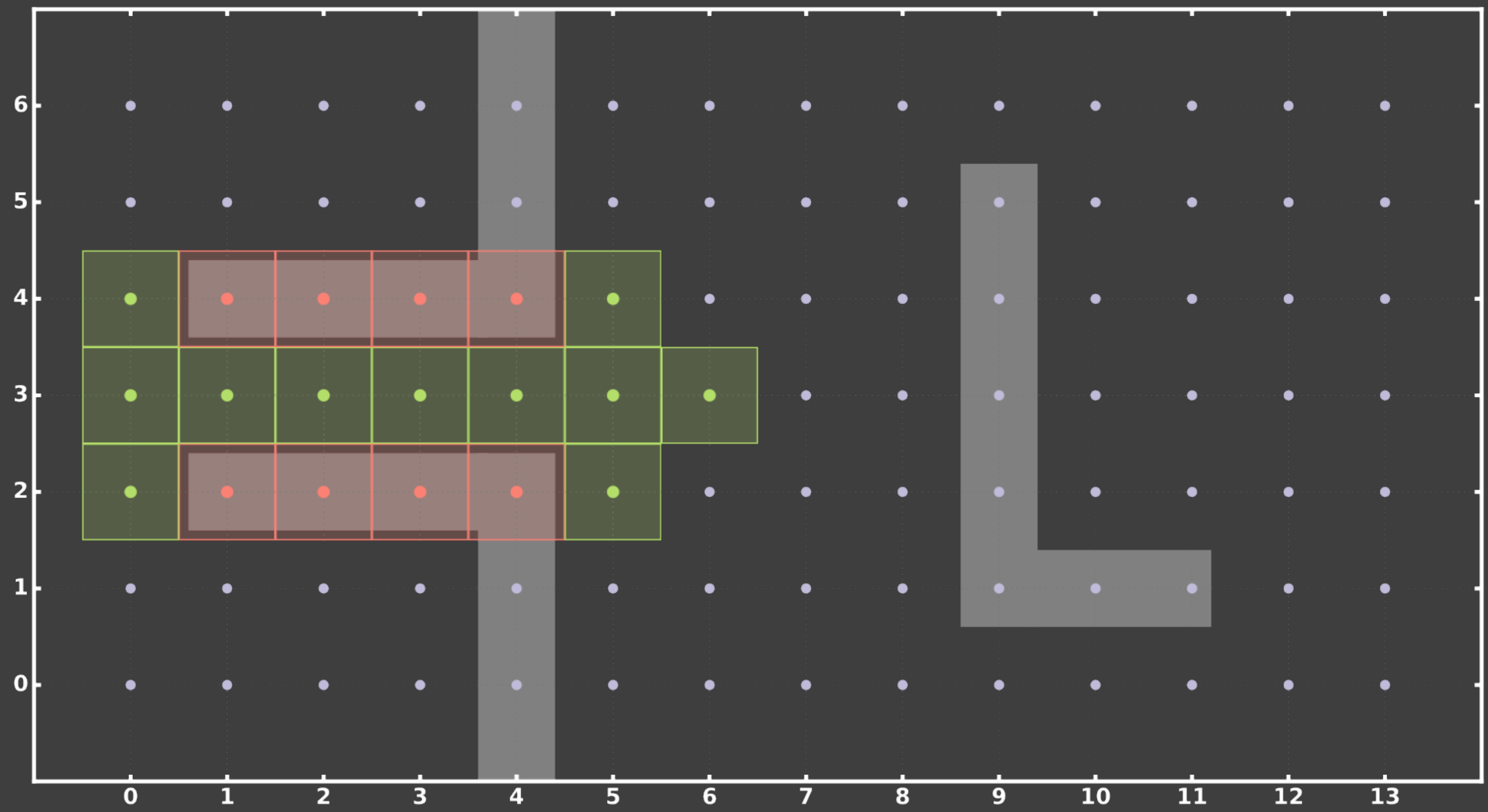
Frontier-based Planning

Until map is explored:

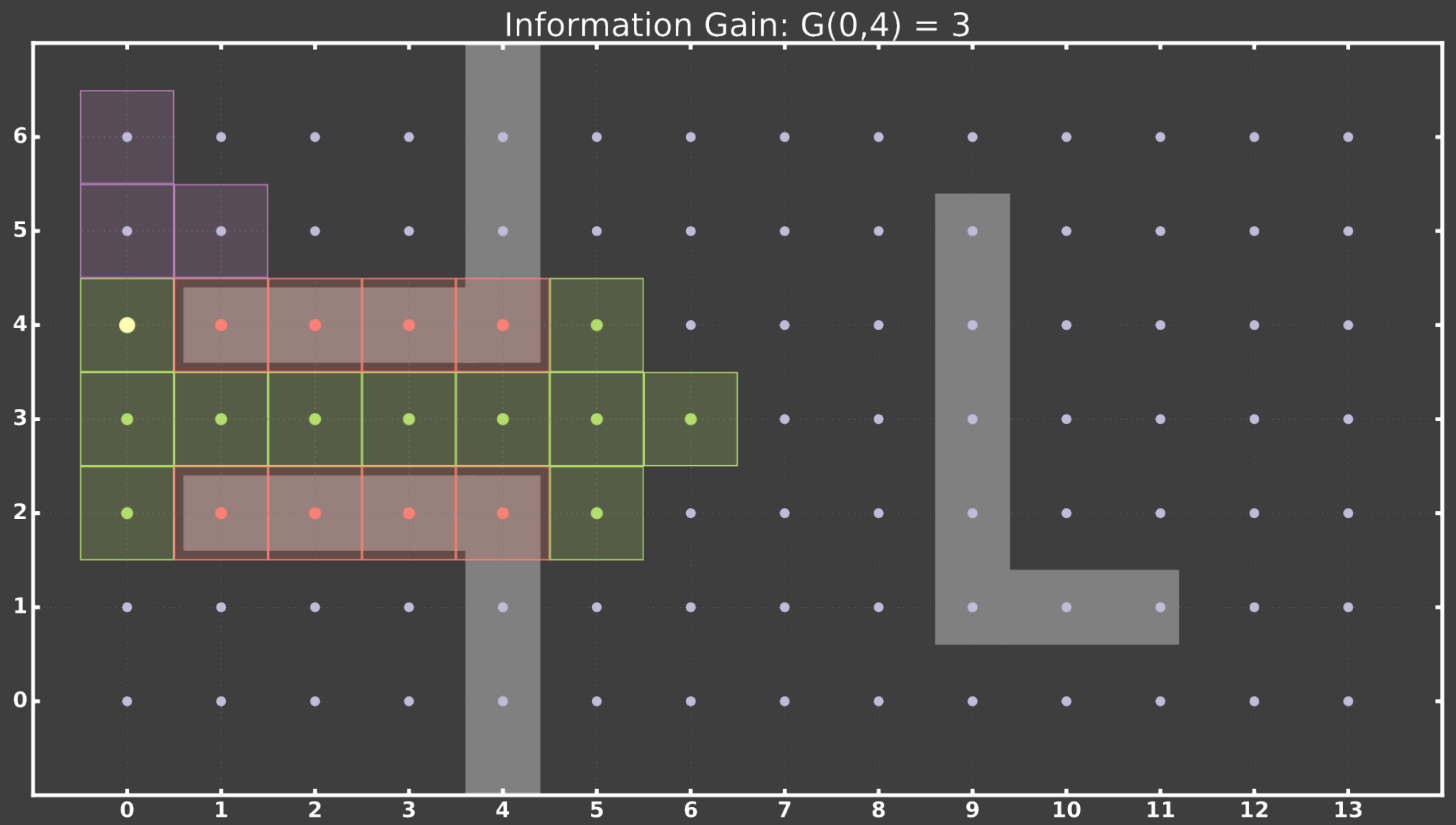
- Find borders between free and unknown space in the map
- Evaluate information gain at each border
- Go to best point
- Update map from this view

[B. Yamauchi, CIRA, 1997]

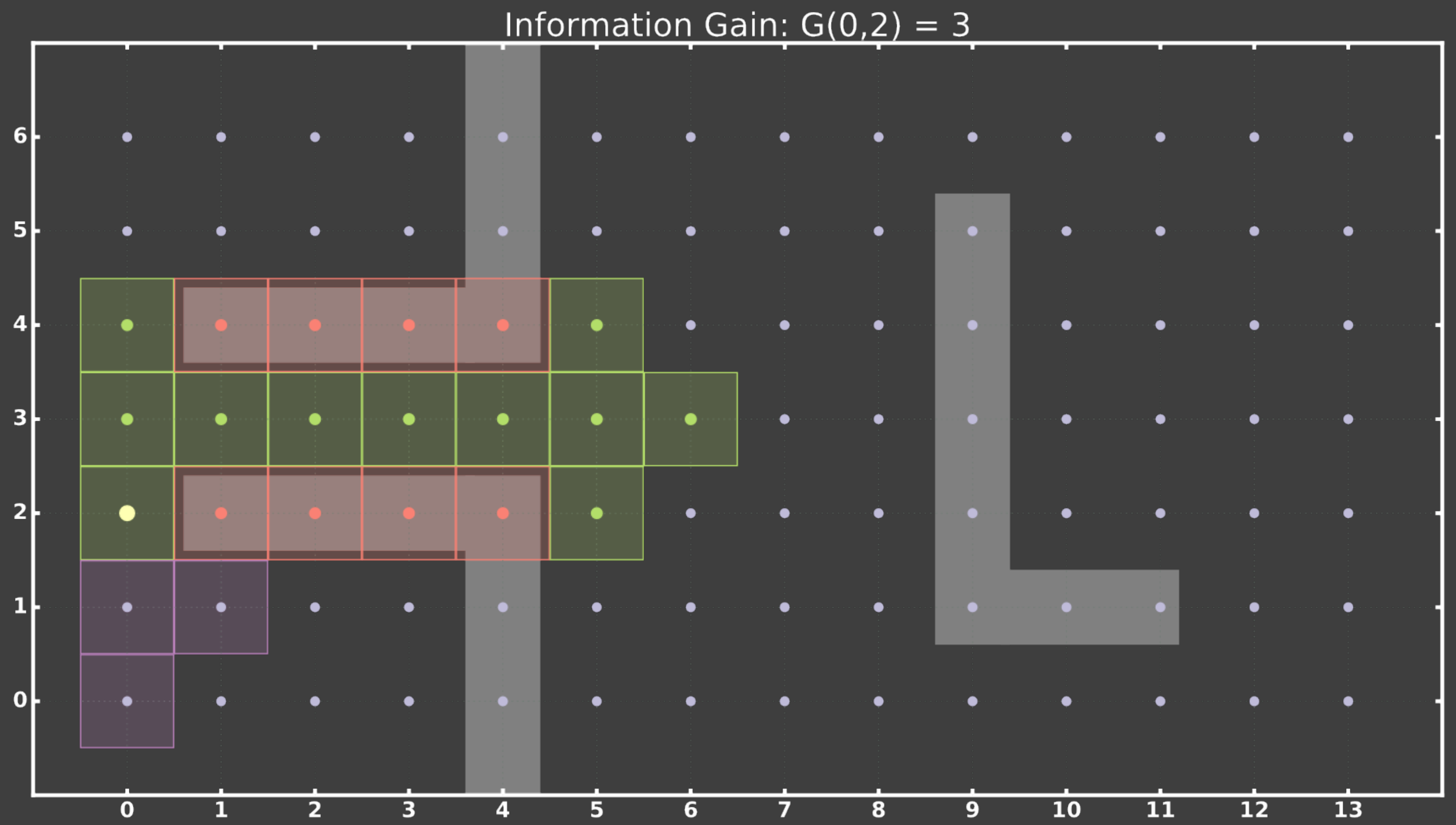
Frontier-based Planning



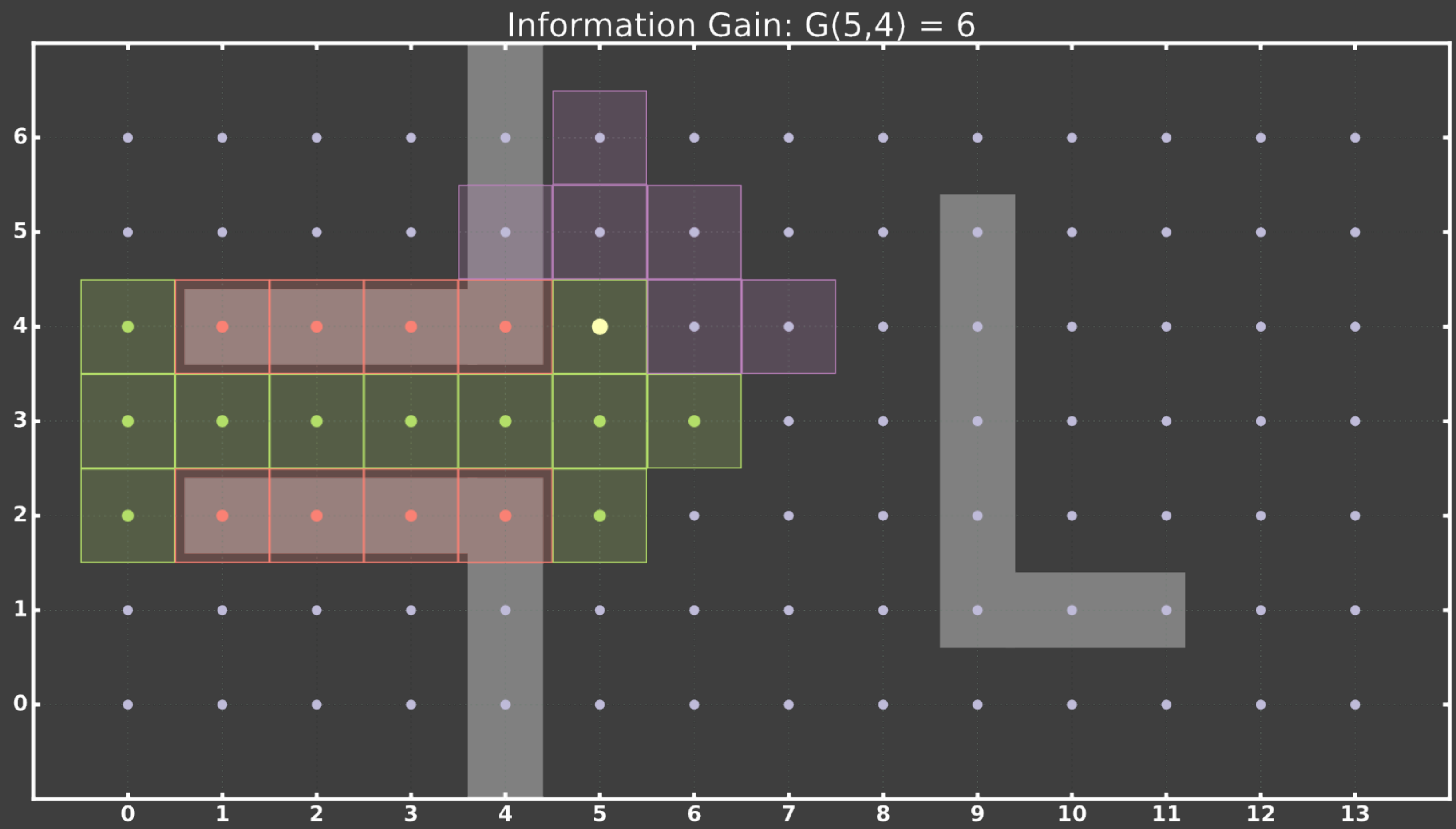
Frontier-based Planning



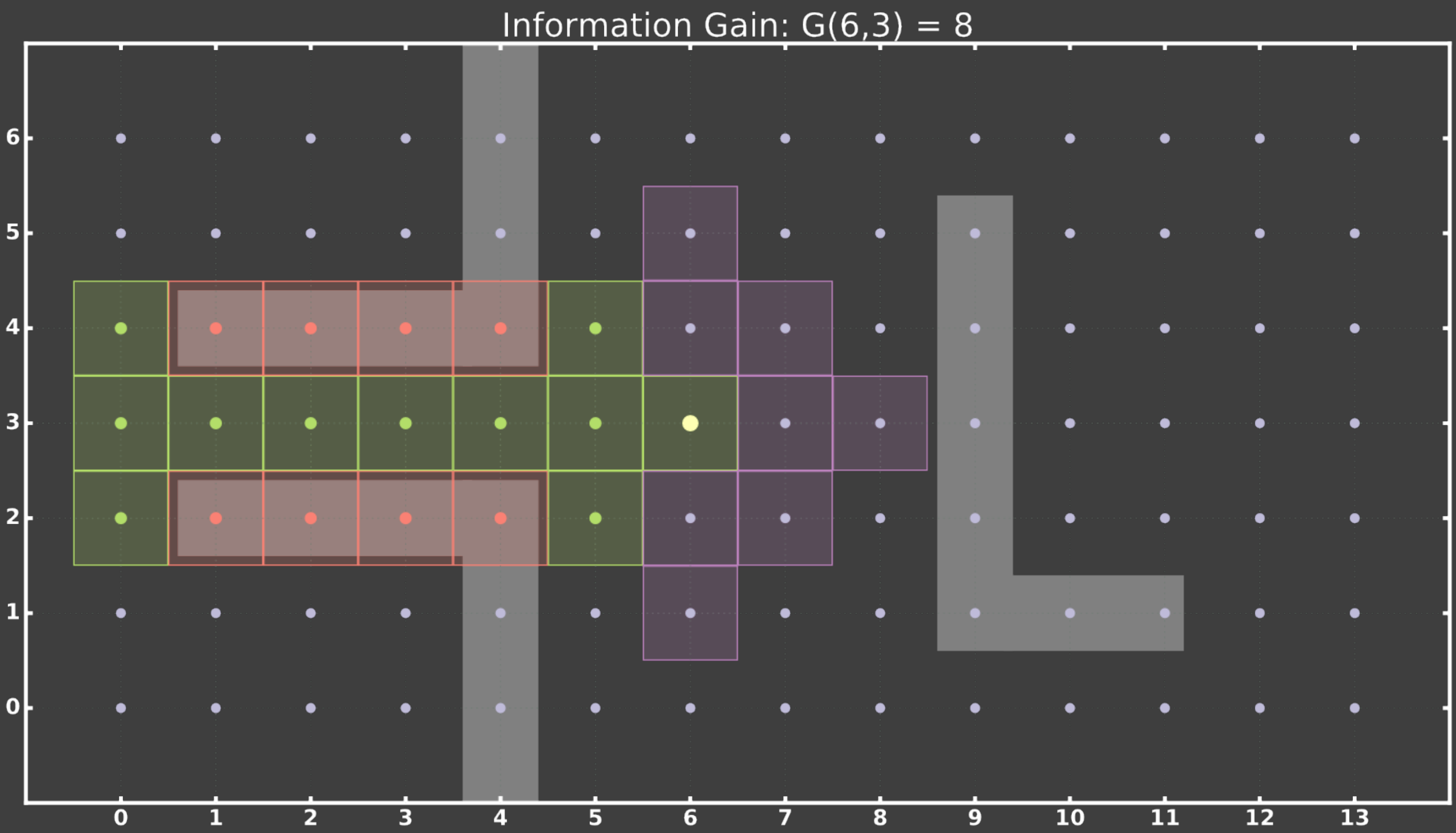
Frontier-based Planning



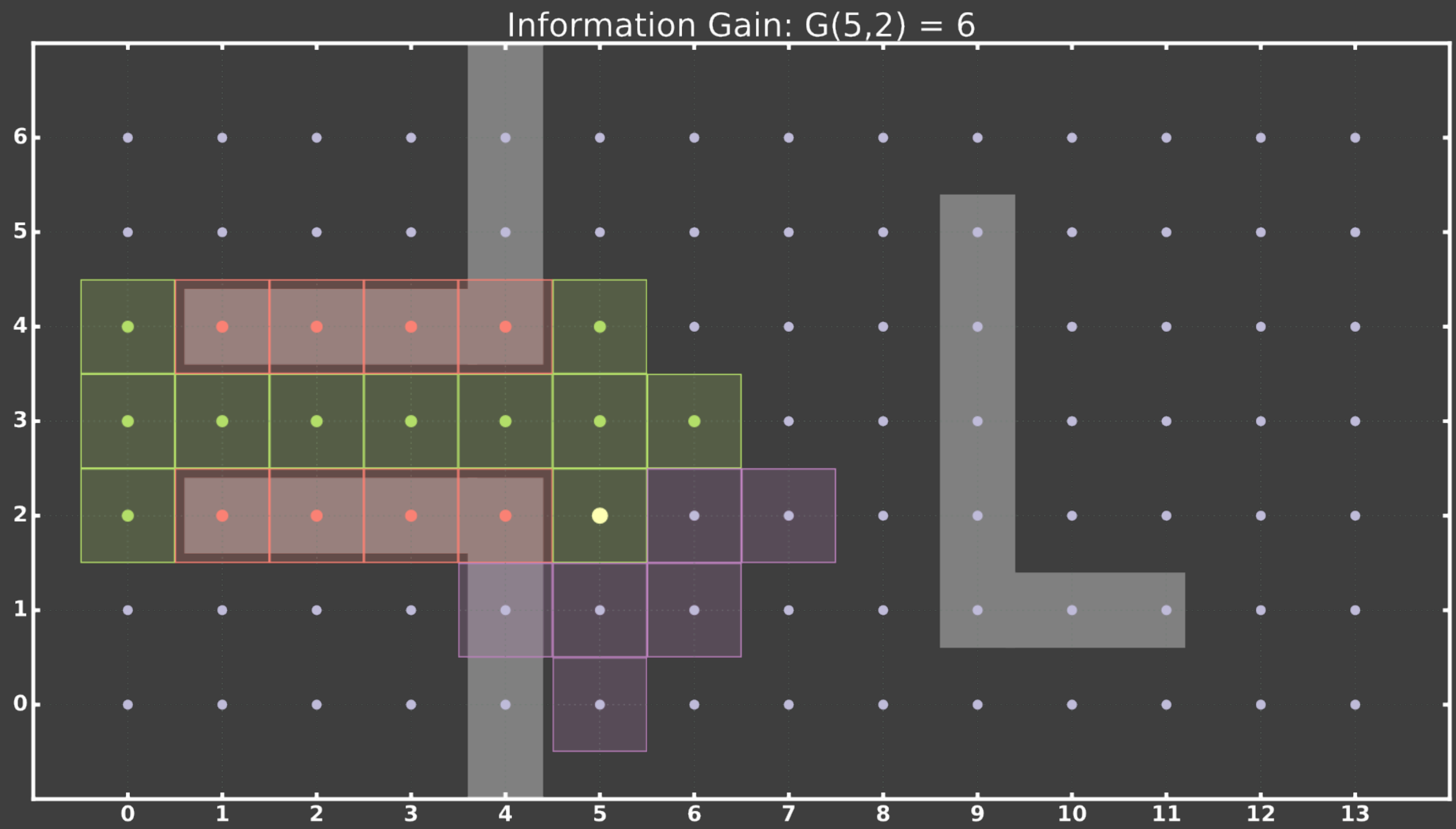
Frontier-based Planning



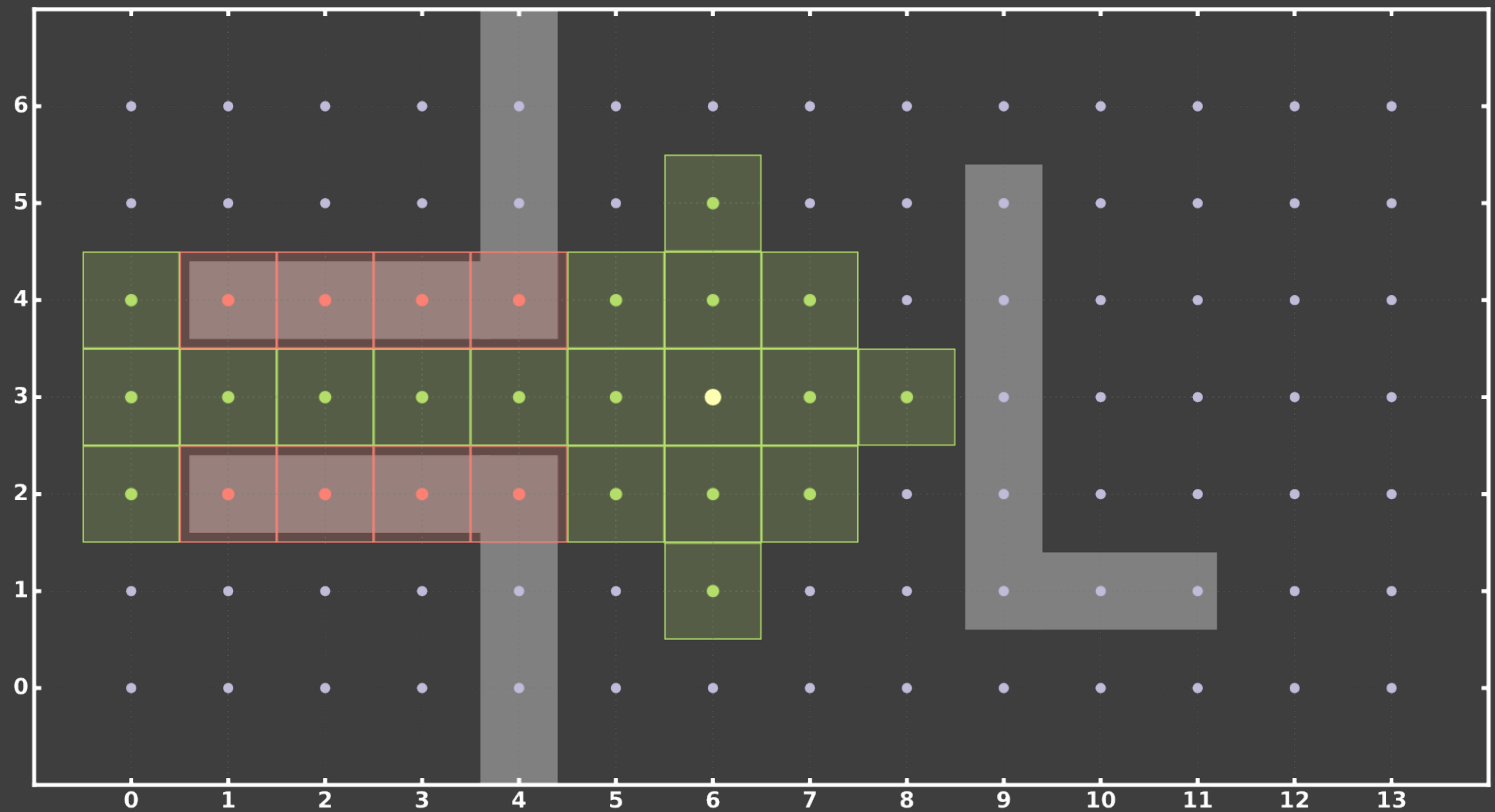
Frontier-based Planning



Frontier-based Planning



Frontier-based Planning



Frontier-based Planning

Pro

- Information Gain in each step is maximized
- Deterministic
- Low number of views

Contra

- Computationally expensive

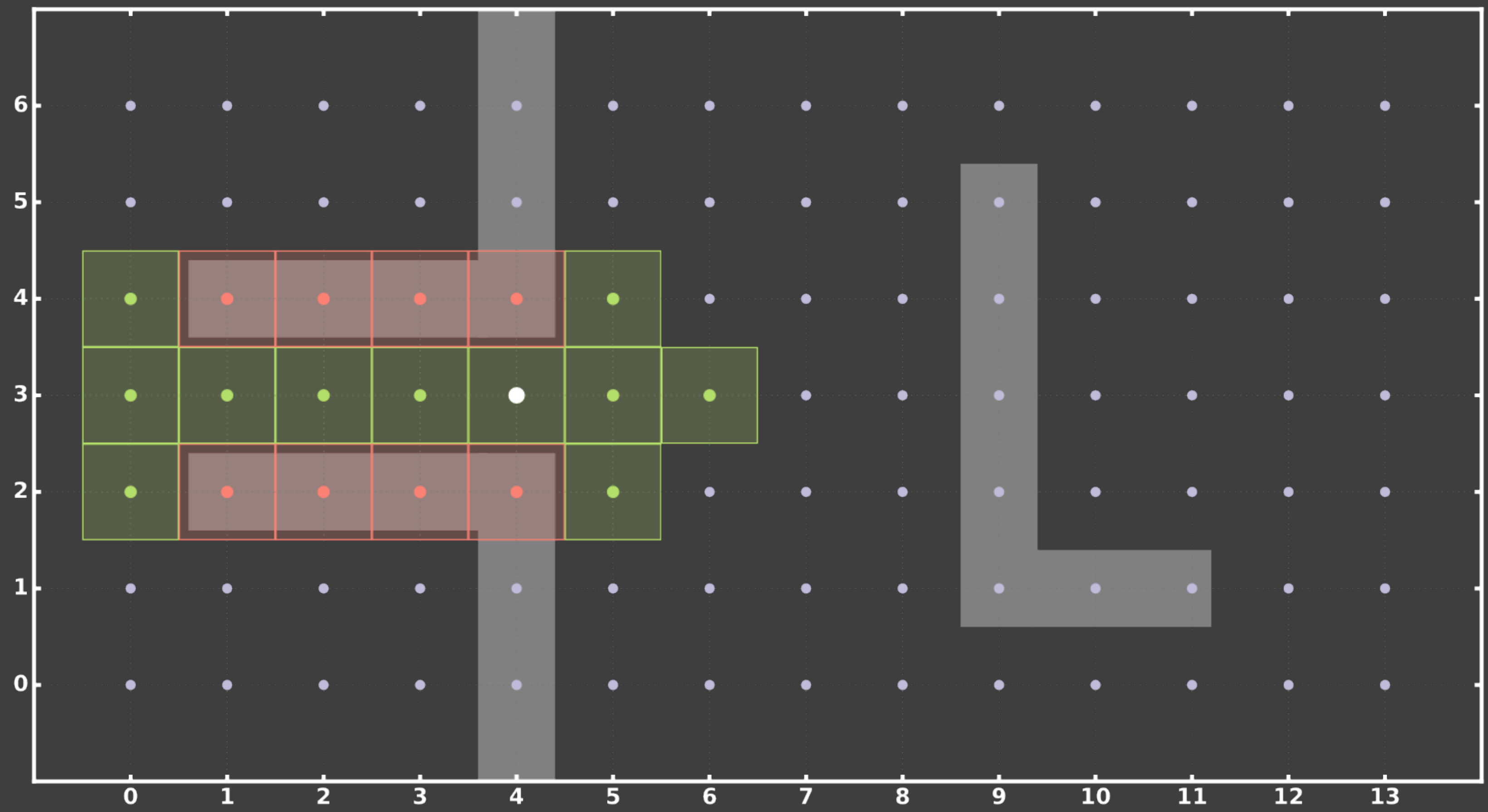
Receding Horizon Planning

Until map is explored:

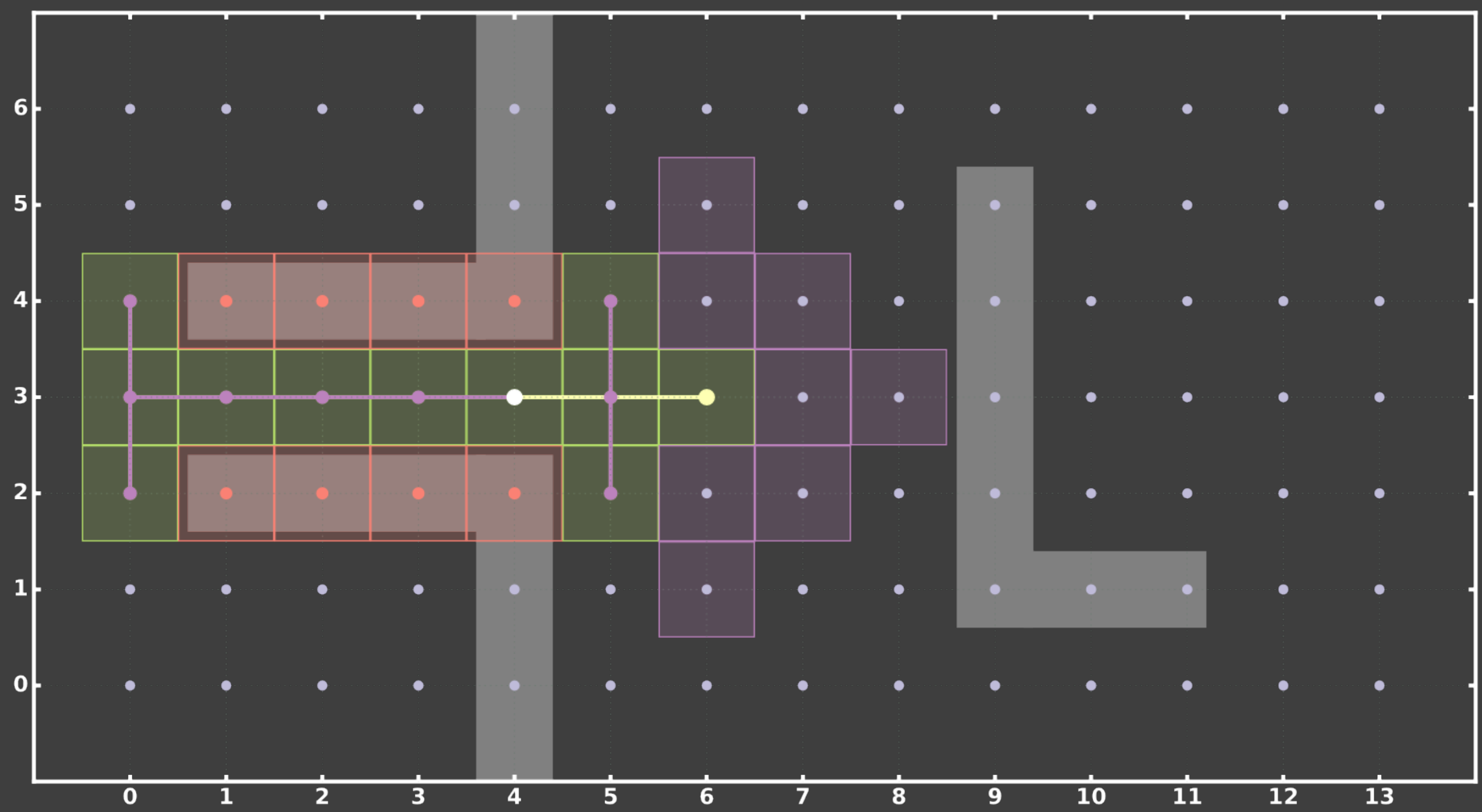
- Span RRT with k branches
- Evaluate information gain at each node
- Move one edge towards the best node
- Update map from this view
- Store best branch for next iteration

[A. Bircher et. al, ICRA, 2016]

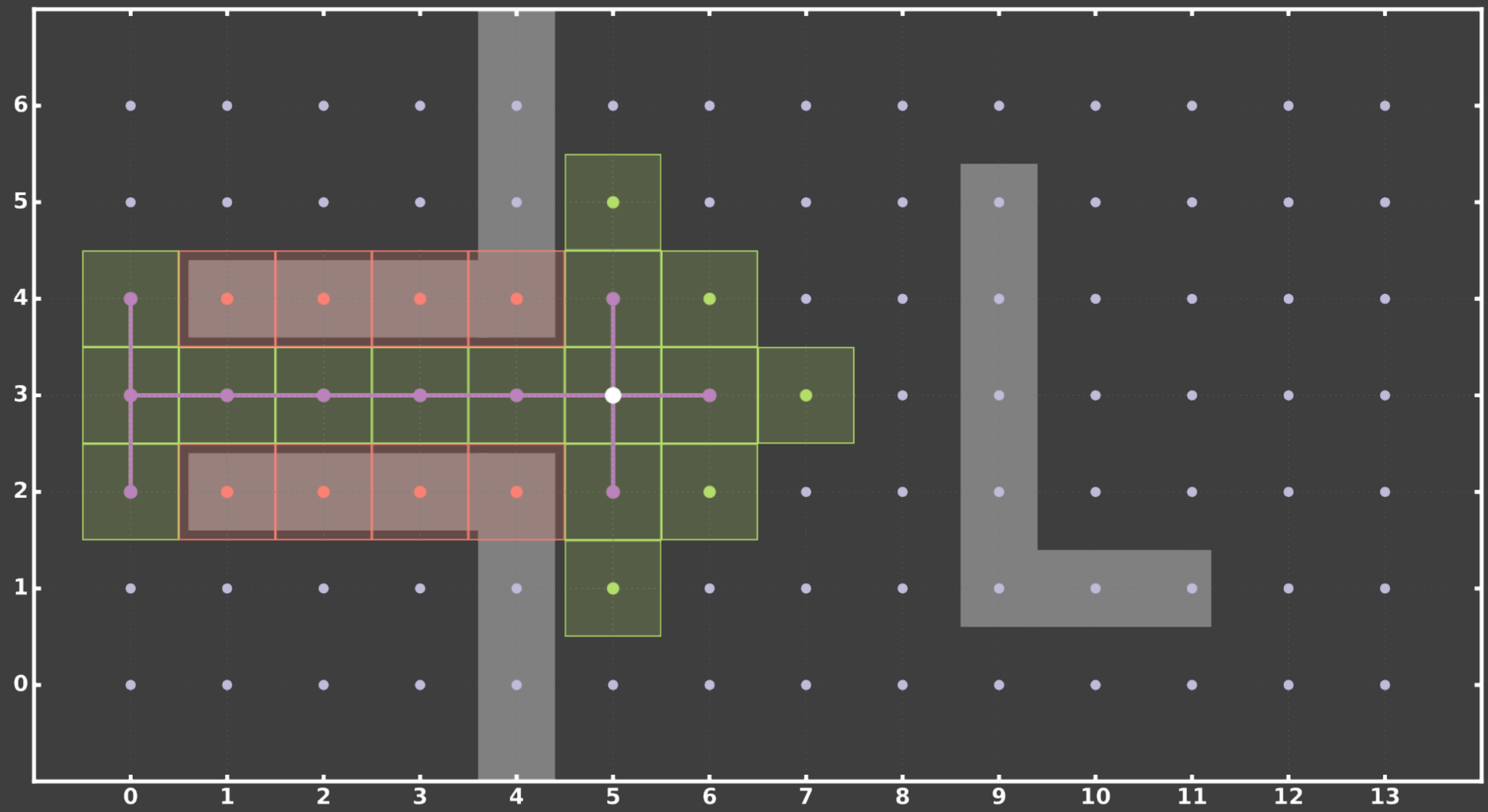
Receding Horizon Planning



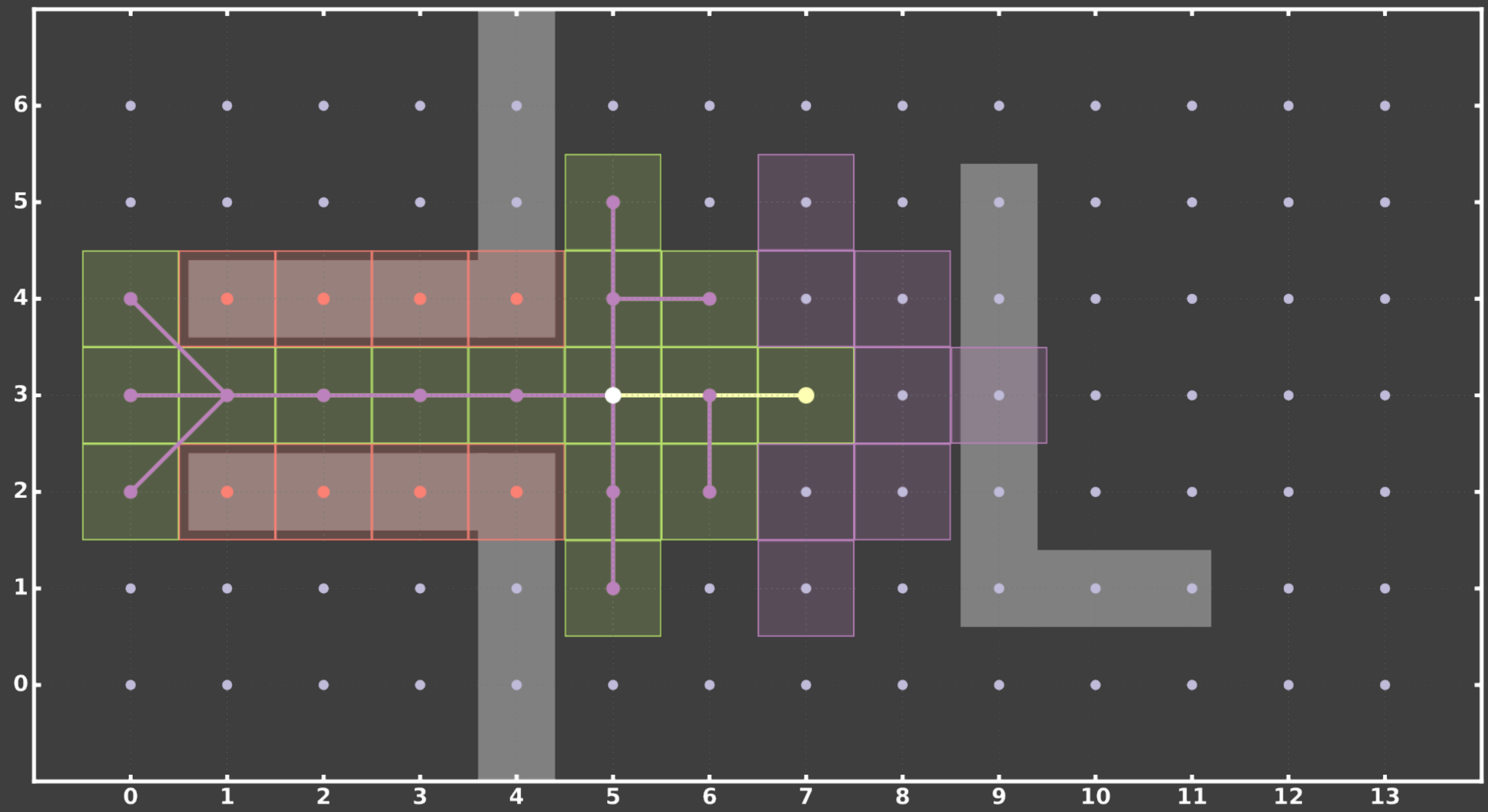
Receding Horizon Planning



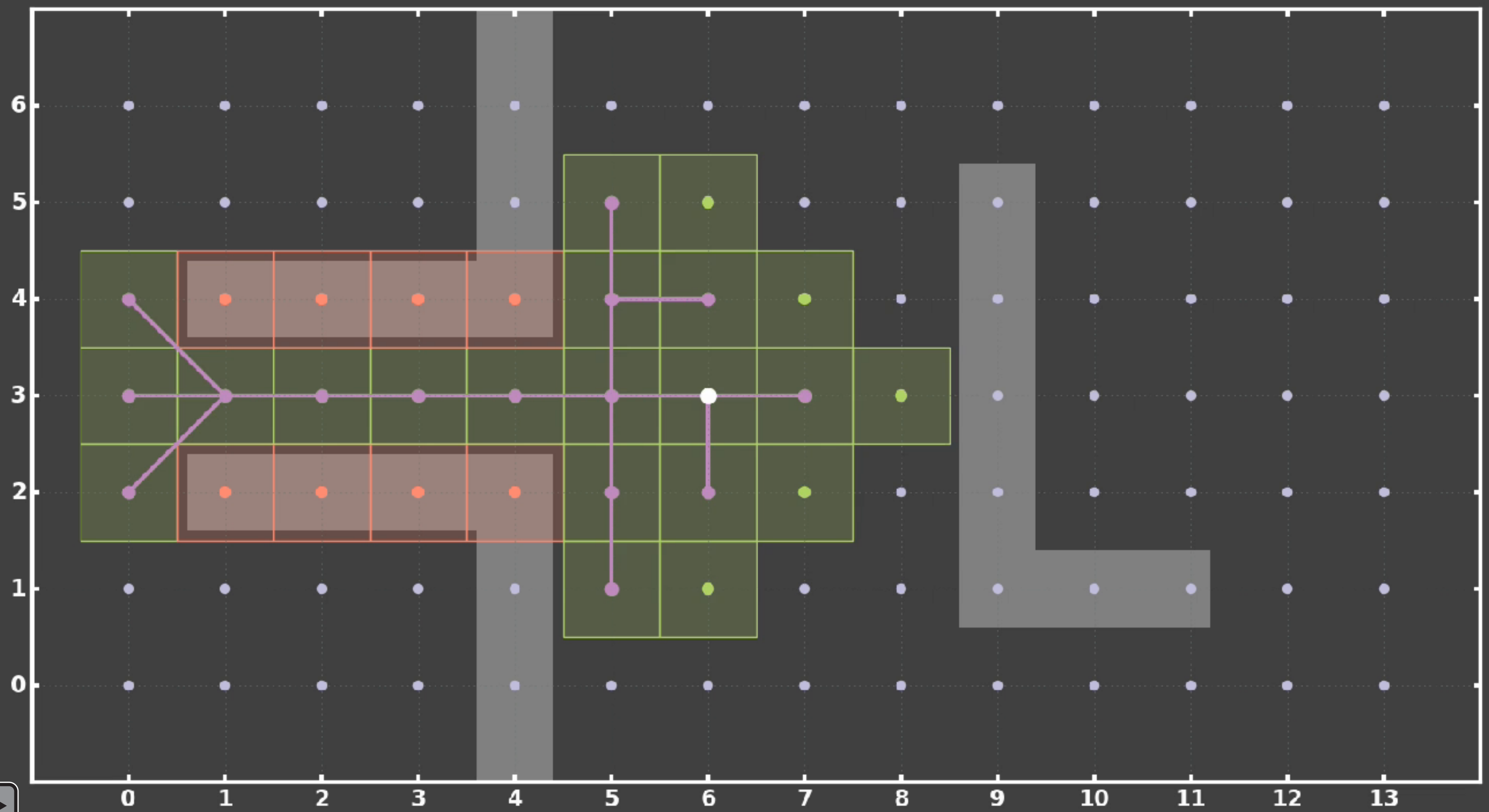
Receding Horizon Planning



Receding Horizon Planning



Receding Horizon Planning



Receding Horizon Planning

Pro

- Scales very good for large areas
- Fast computation

Contra

- Stable localization and mapping is needed
- Non-ideal paths

Comparison Frontier vs Receding Horizon

Gazebo based simulation benchmark with hexacopter MAV:

	Small Scale **Area: 20x10x3m, Resolution 0.4m**		**Large Scale** **Area: 50x26x14m, Resolution 0.25m**	
	Frontier	Receding Horizon	Frontier	Receding Horizon
t_{tot}	469.7s	501s	70% after 1670 min	43.8 min
t_{comp}^{tot}	83.8s	15.2s	1660 min	9.4 min
t_{comp}^{step}	5.7s	0.15s	25.9 min	1.6 s

[A. Bircher et. al, ICRA, 2016]

- Frontier-based planning results in better paths but is unfeasible in large environments

- Fast calculations in Receding Horizon planning compensates the non-optimal path

Summary

Conclusion

- Two different approaches for path planning and exploration were presented
- Accurate algorithms scale poor in large spaces → randomized approaches needed

Outlook

- There are many, many more algorithms
  - Have to find the one that matches your problem the best
- Important addition: Multi-Agent-Planning
  - Dedicated talk on this topic!