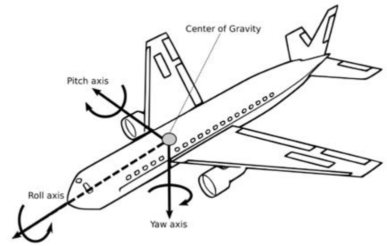
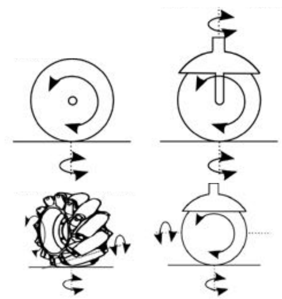
Path Planning

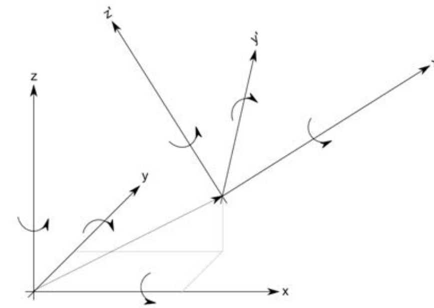
Chapter 4

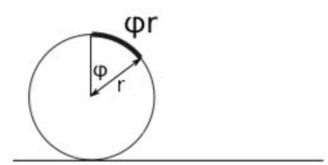
Recap: Kinematics

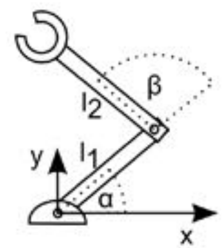
- Degrees of Freedom
- Coordinate Transforms
- Kinematic Constraints
- Forward Kinematics
- [Odometry]
- Inverse Kinematics
- Position Control

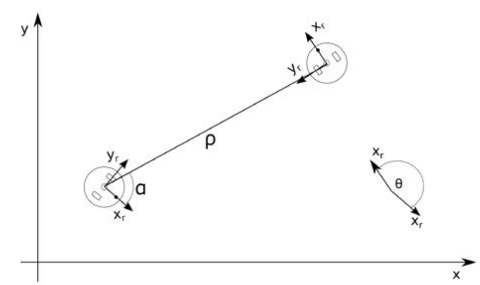












<u>Basics of how robots move in the world and know where they are</u>

(and a rough idea why this does not work)

Summary: Inverse Kinematics of a Mobile Robot

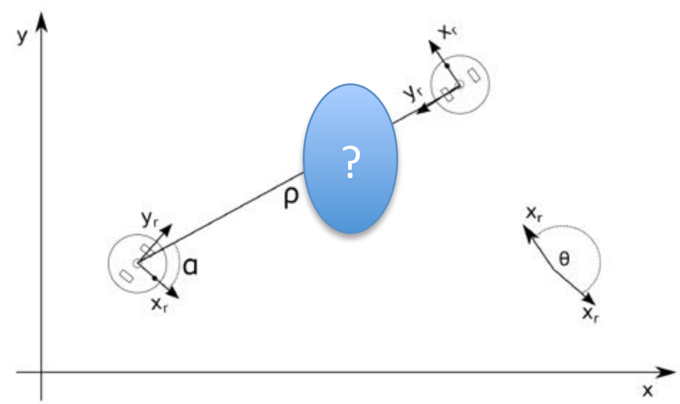
- Calculate suitable velocities that drive the robot toward your goal

$$
\begin{aligned}
\dot{x} &= p_1\rho \\
\dot{\theta} &= p_2\alpha + p_3\eta
\end{aligned}
$$

- Calculate the necessary wheel-speed

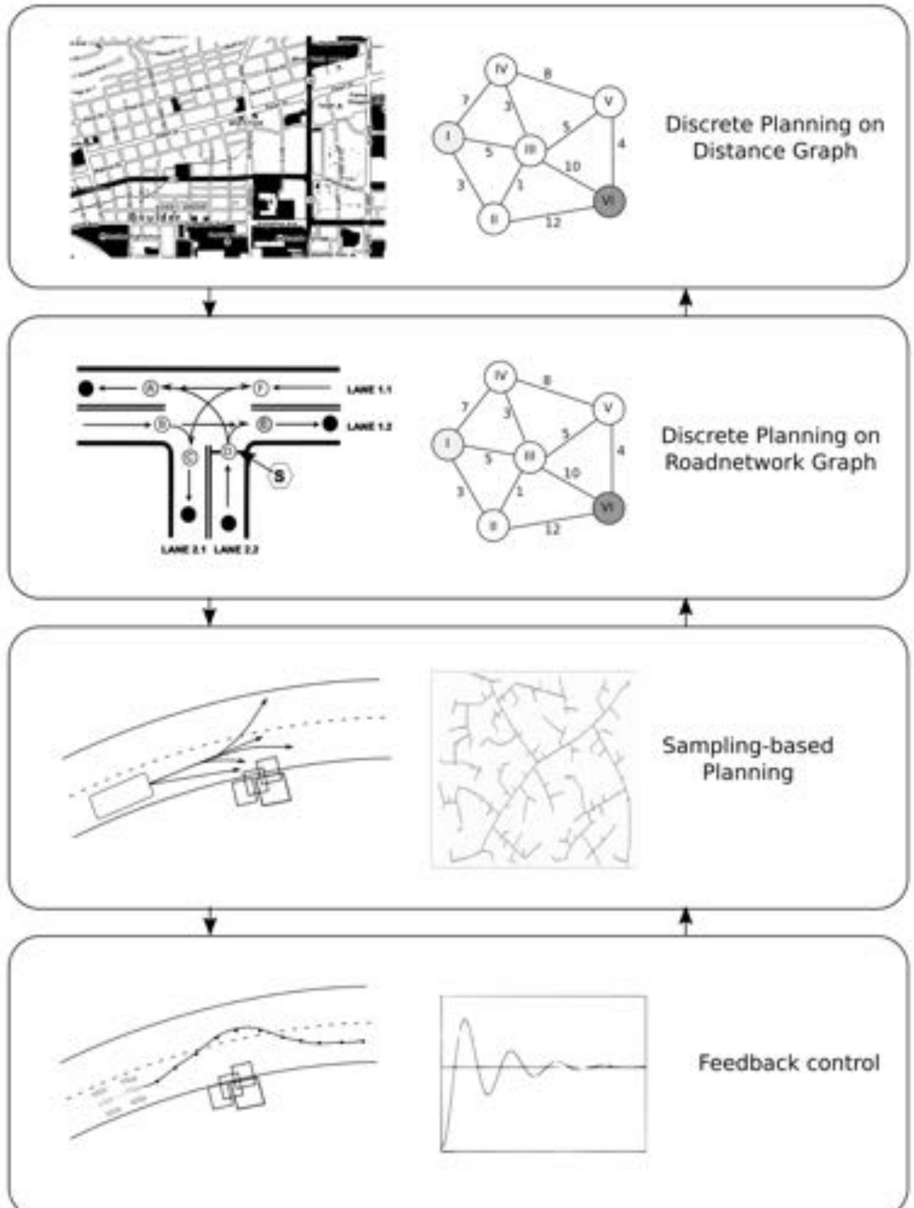
$$
\begin{aligned}
\dot{\phi}_l &= (2\dot{x}_R/r - \dot{\theta}d)/2 \\
\dot{\phi}_r &= (2\dot{x}_R/r + \dot{\theta}d)/2
\end{aligned}
$$

- Problem
  – How to deal with obstacles?
  – How to find short(est) paths?
- Solution: Path Planning

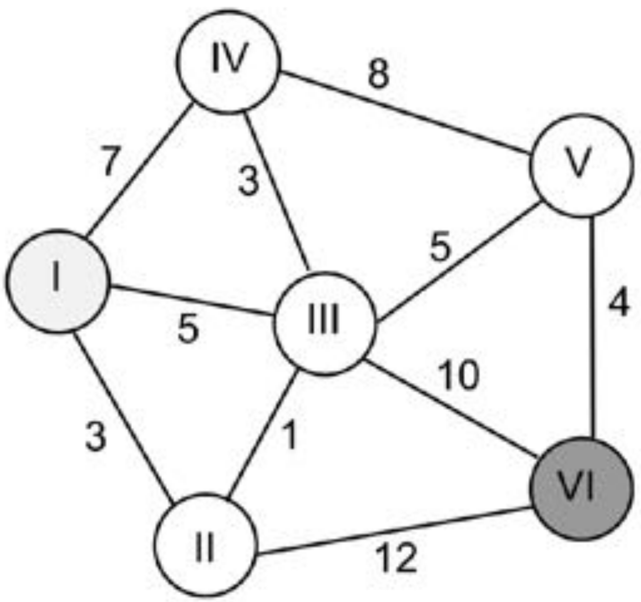


Exercise: Plan for a robotic car

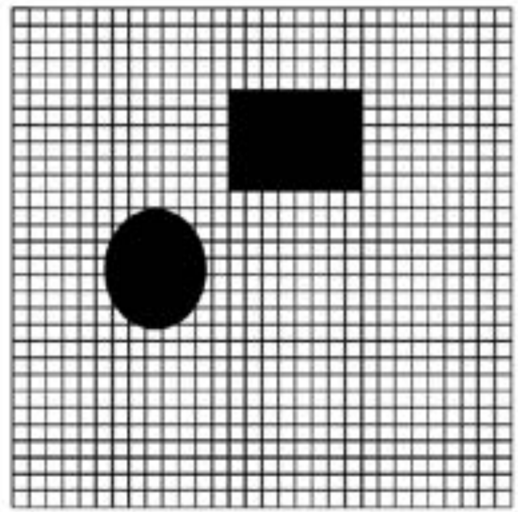
Planning across length scales



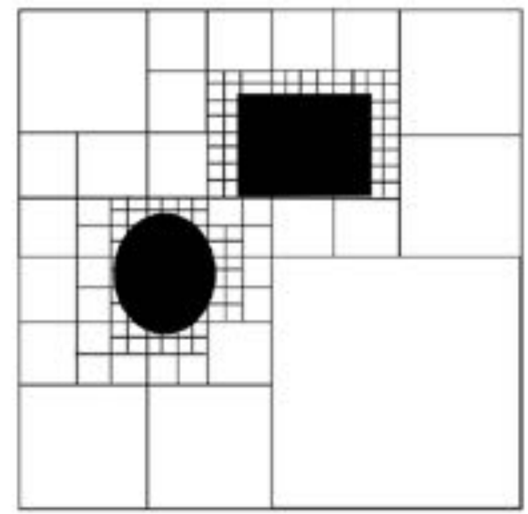
Map Representations



Topological Map
(Continuous Coordinates)

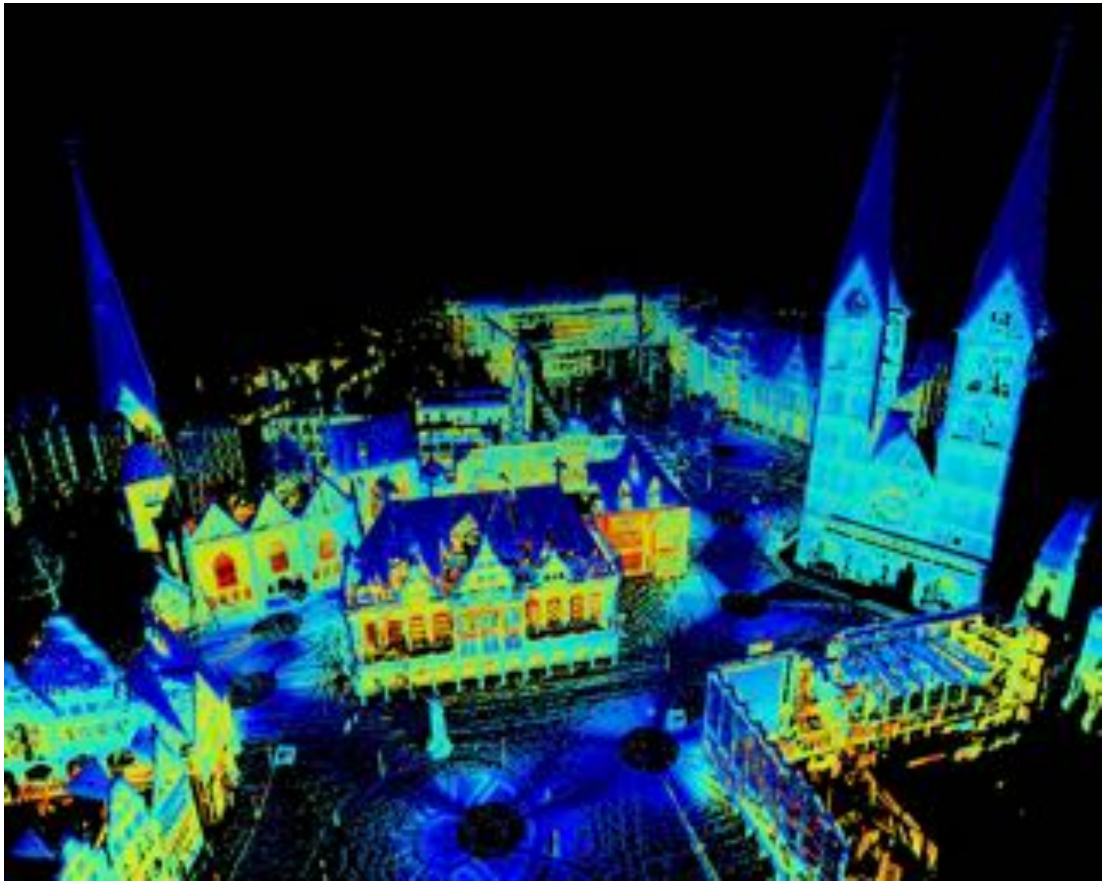


Grid Map
(Discrete Coordinates)



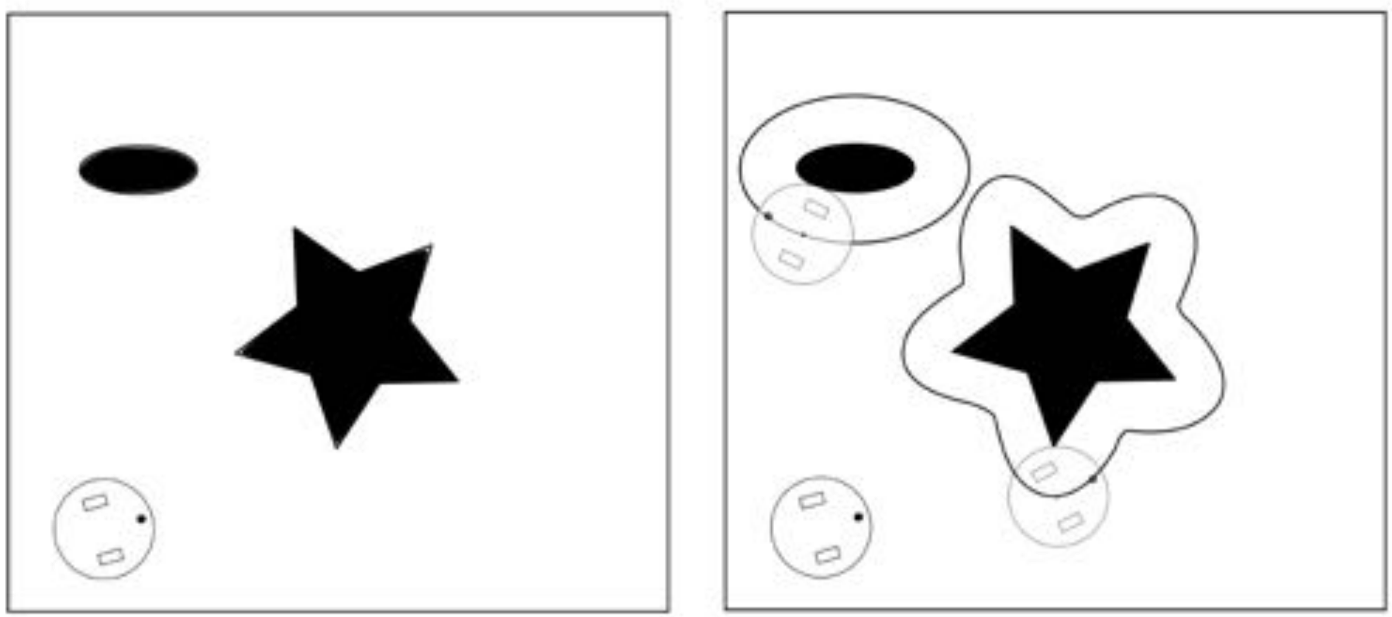
K-d Tree Map (Quadtree)

Octree



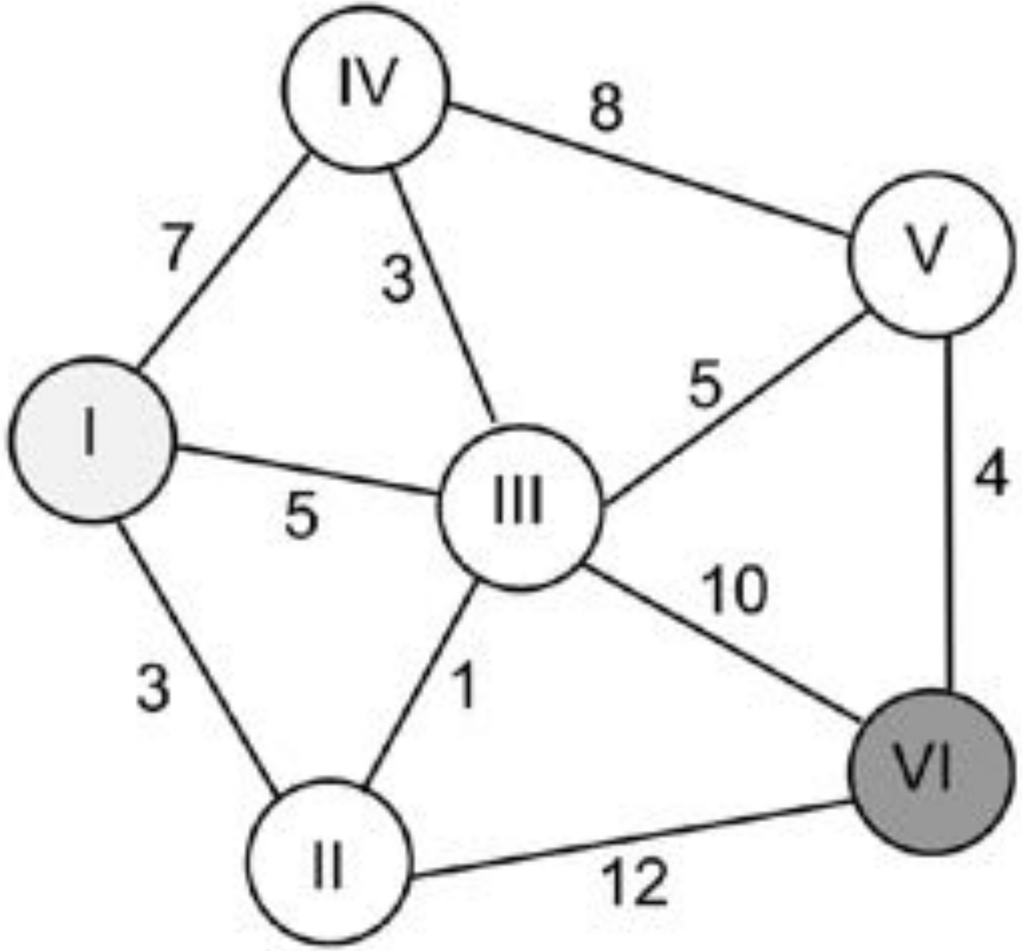
https://www.youtube.com/watch?v=7ZsxJzR14rc

Configuration Space

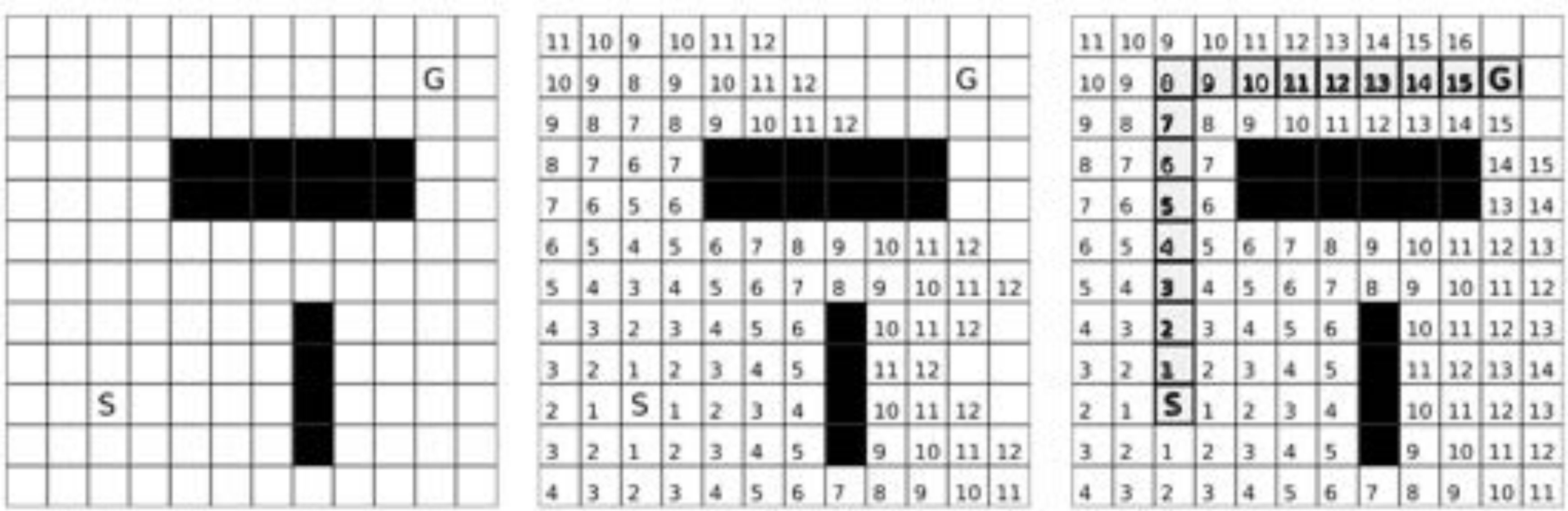


Grow obstacles by robot radius (only really works in 2D)

Dijkstra's Algorithm

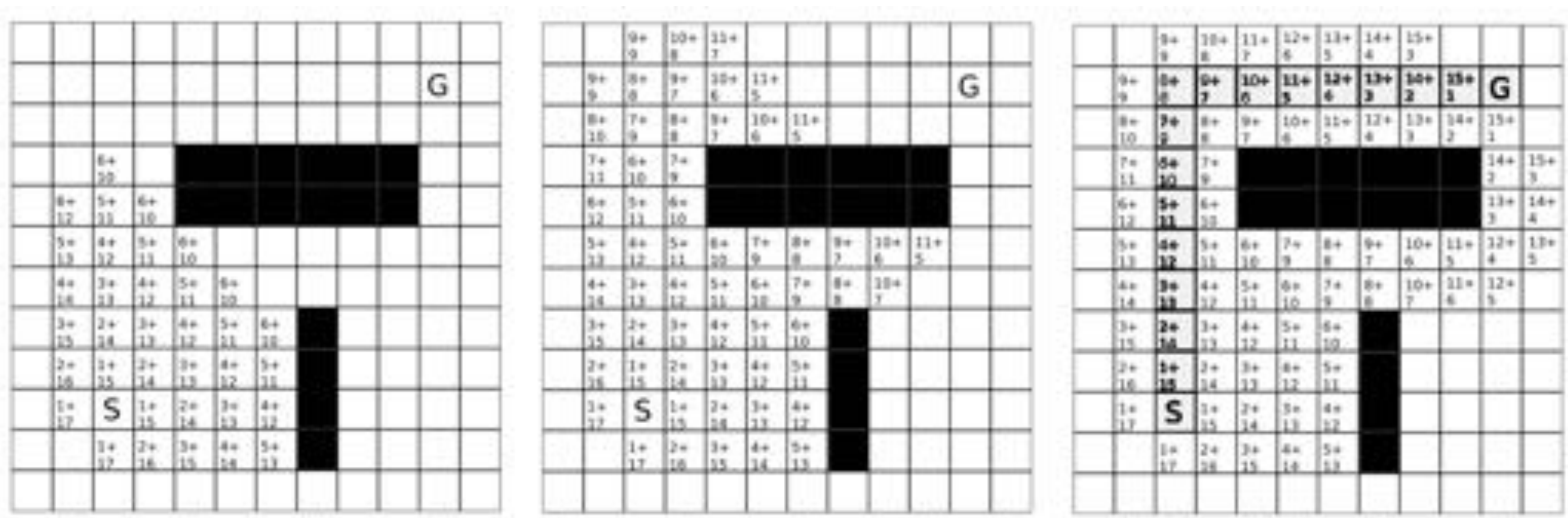


Dijkstra’s Algorithm on a Grid

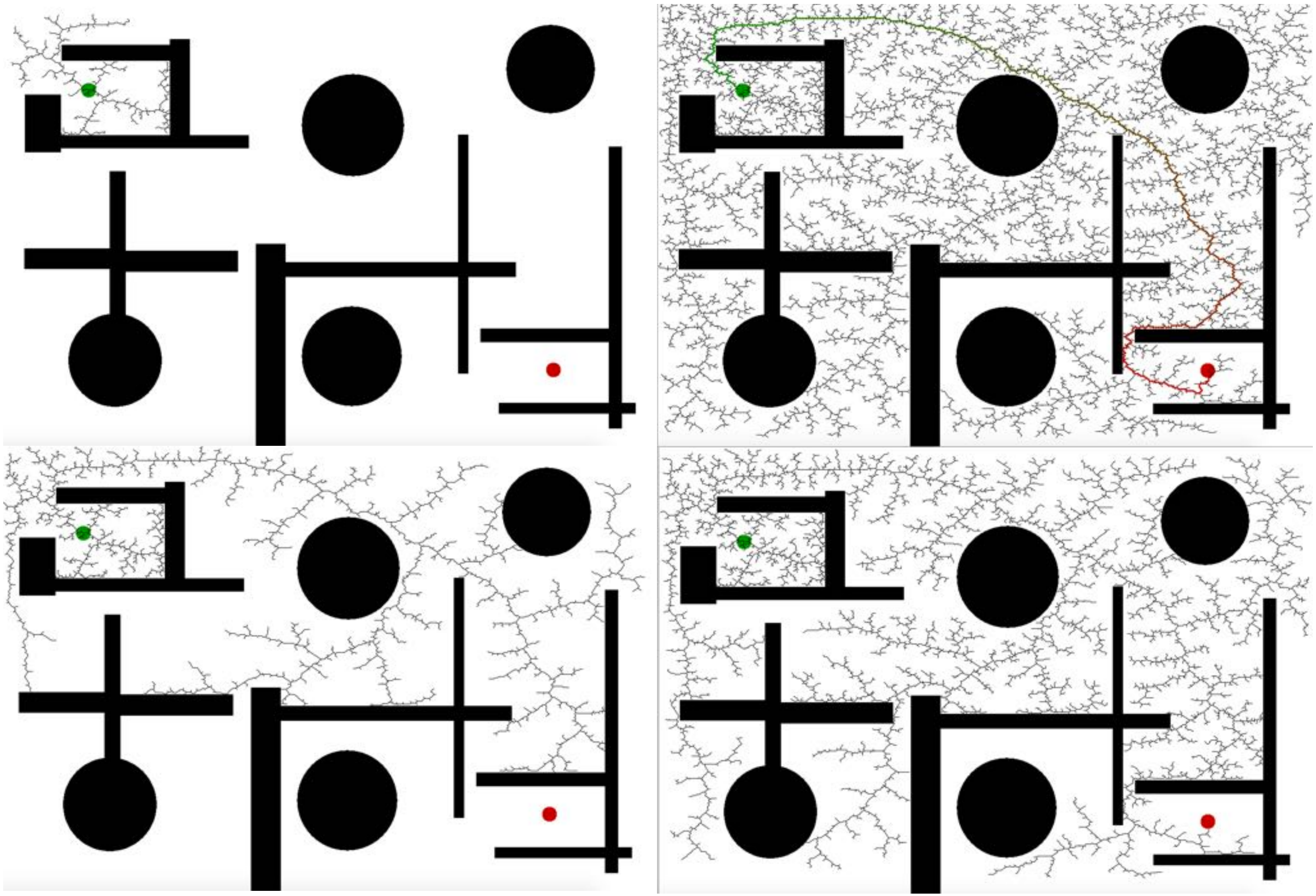


Problem: A lot of useless exploration

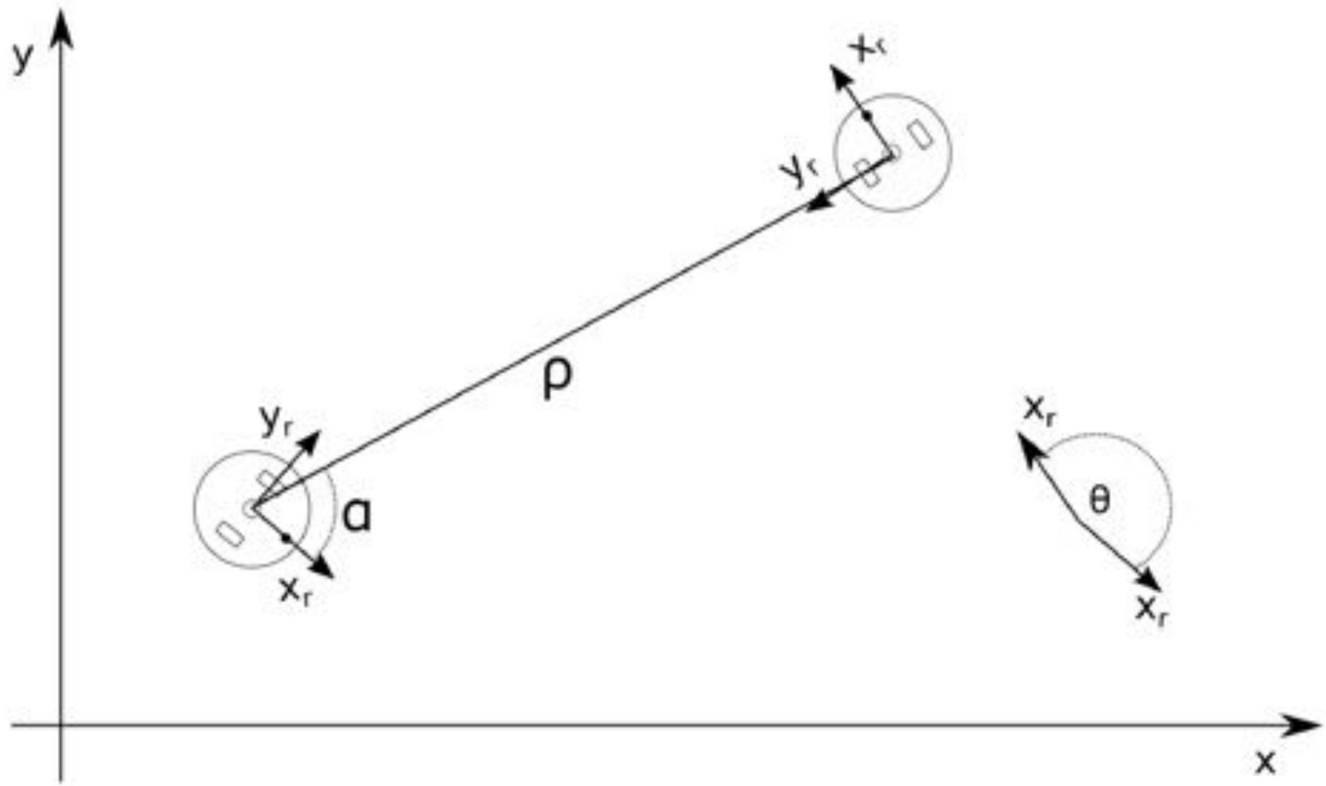
Dijkstra plus directional heuristic: A*



Rapidly Exploring Random Trees



Turning waypoints into trajectories



Take-home lessons

- First step in addressing a planning problem is choosing a suitable map representation
- Reduce robot to a point-mass by inflating obstacles
- Grid-based algorithms are complete, sampling-based ones probabilistically complete, but usually faster
- Most real planning problems require combination of multiple algorithms