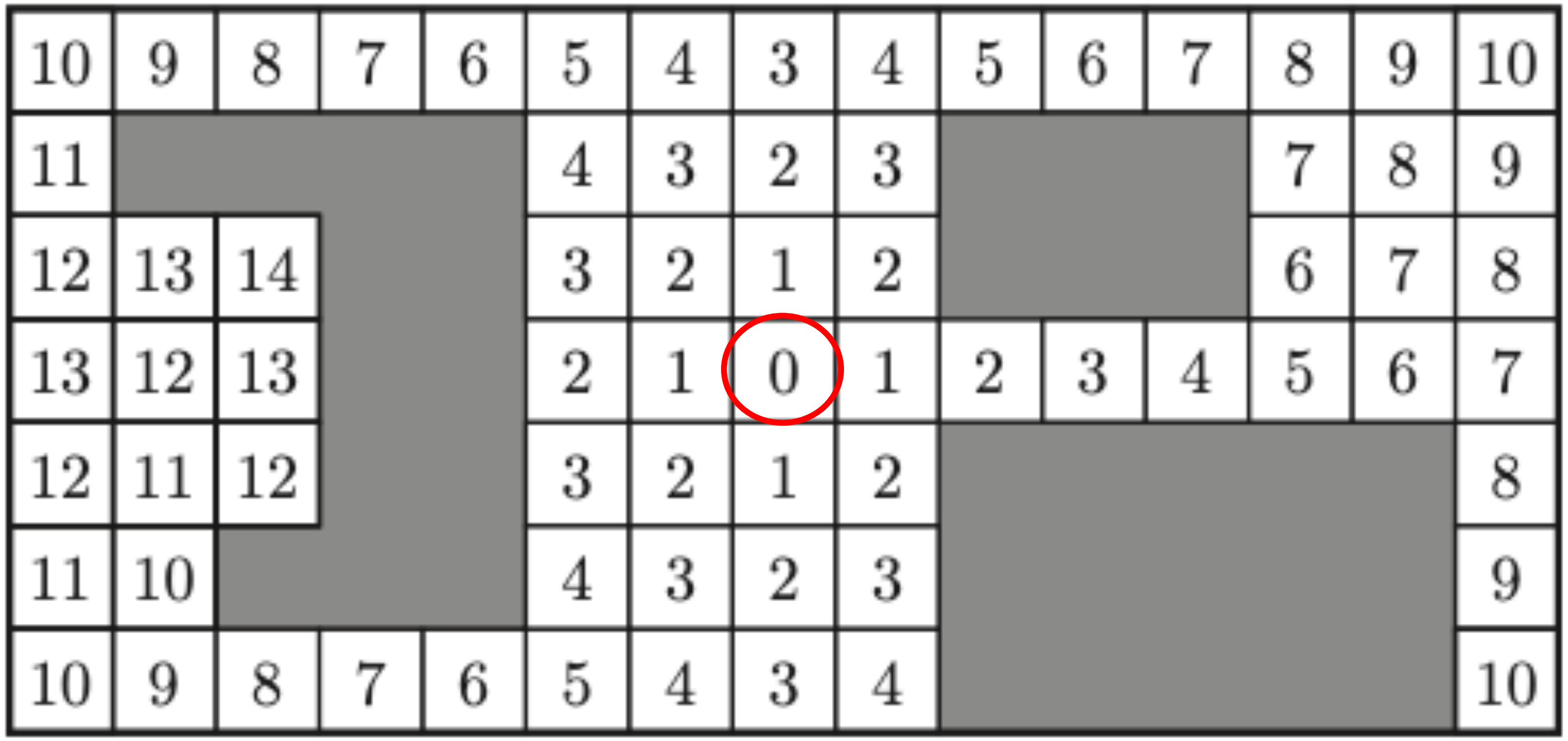
Wavefront planner on a two-dimensional grid.



10	9	8	7	6	5	4	3	4	5	6	7	8	9	10
11	▨	▨	▨	▨	4	3	2	3	▨	▨	▨	7	8	9
12	13	14	▨	▨	3	2	1	2	▨	▨	▨	6	7	8
13	12	13	▨	▨	2	1	0	1	2	3	4	5	6	7
12	11	12	▨	▨	3	2	1	2	▨	▨	▨	▨	▨	8
11	10	▨	▨	▨	4	3	2	3	▨	▨	▨	▨	▨	9
10	9	8	7	6	5	4	3	4	▨	▨	▨	▨	▨	10

D* Algorithm

- generalizes the occupancy grid to a cost map which represents the cost $c \in R$, $c > 0$ of traversing each cell in the horizontal or vertical direction. The cost of traversing the cell diagonally is c sqrt(2). For cells corresponding to obstacles $c = \inf$.
- supports incremental replanning.
- D* finds the path which minimizes the total cost of travel.

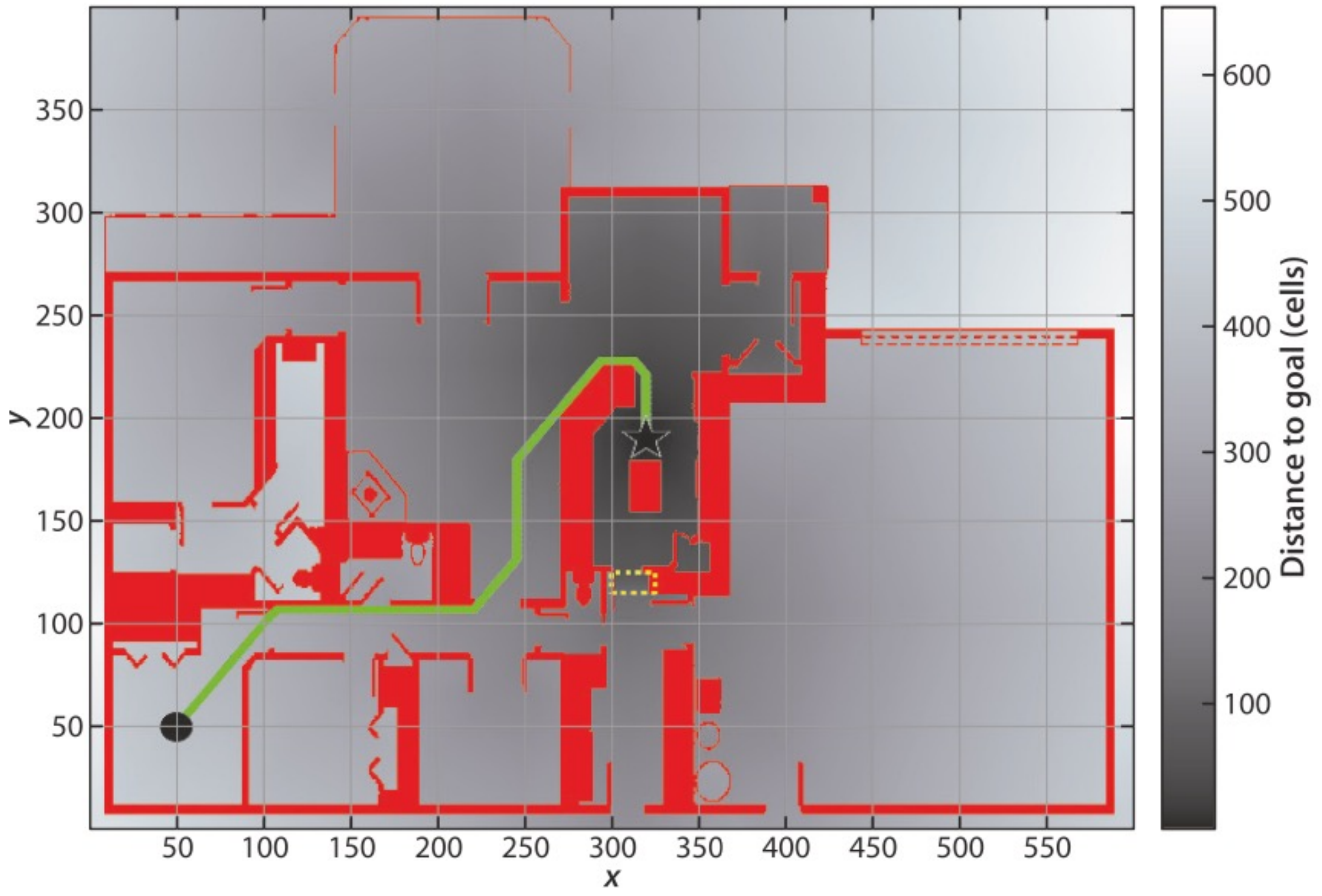
D*

- ds = Dstar(house);
- c = ds.costmap();
- ds.plan(place.kitchen);

creates a very dense directed graph. Every cell is a graph vertex and has a cost, a distance to the goal, and a link to the neighboring cell that is closest to the goal. Each cell also has a state t ∈ {NEW, OPEN, CLOSED}

- ds.query(place.br3,'animate');

Modify planning



Modify planning

- ds.modify_cost([300,325; 115,125], 5);
- ds.plan();
- ds.query(place.br3);

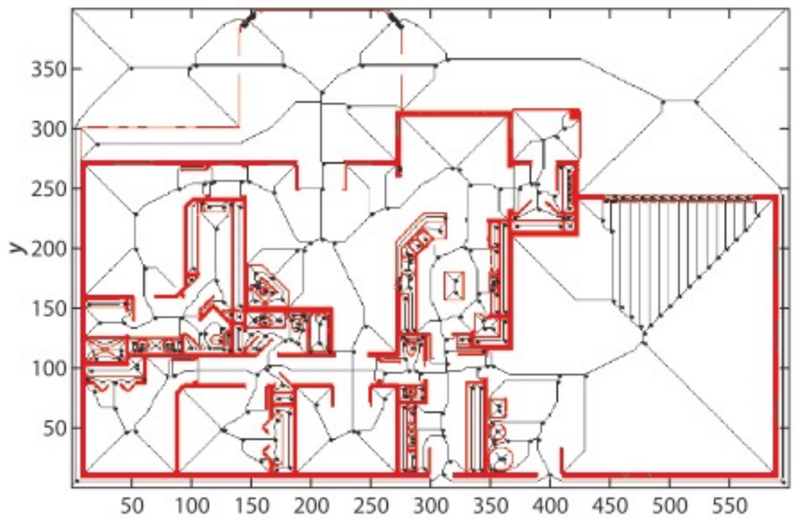
Roadmap methods

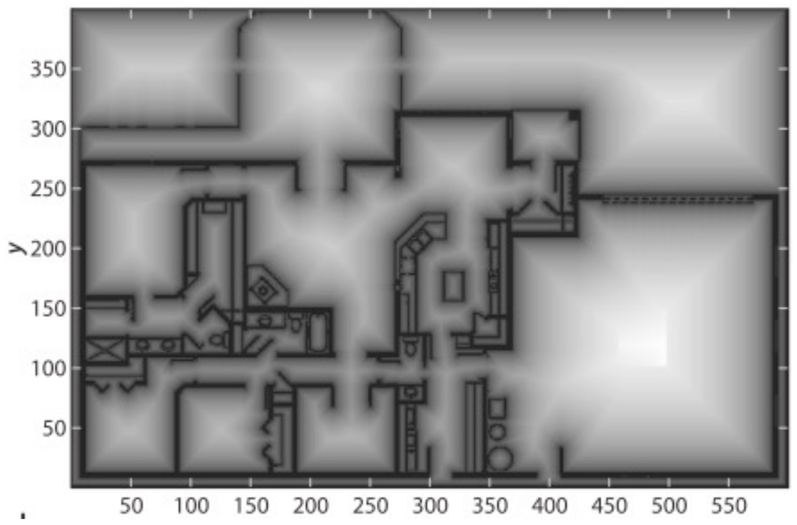
Problems with D*

- In robotic path planning the analysis of the map is referred to as the *planning phase*.
- The query phase uses the result of the planning phase to find a path from A to B. Distance transform and D*, require a significant amount of computation for the planning phase, but the query phase is very cheap.
- If the goal changes the expensive planning phase must be re-executed. Even though D* allows the path to be recomputed as the costmap changes it does not support a changing goal.

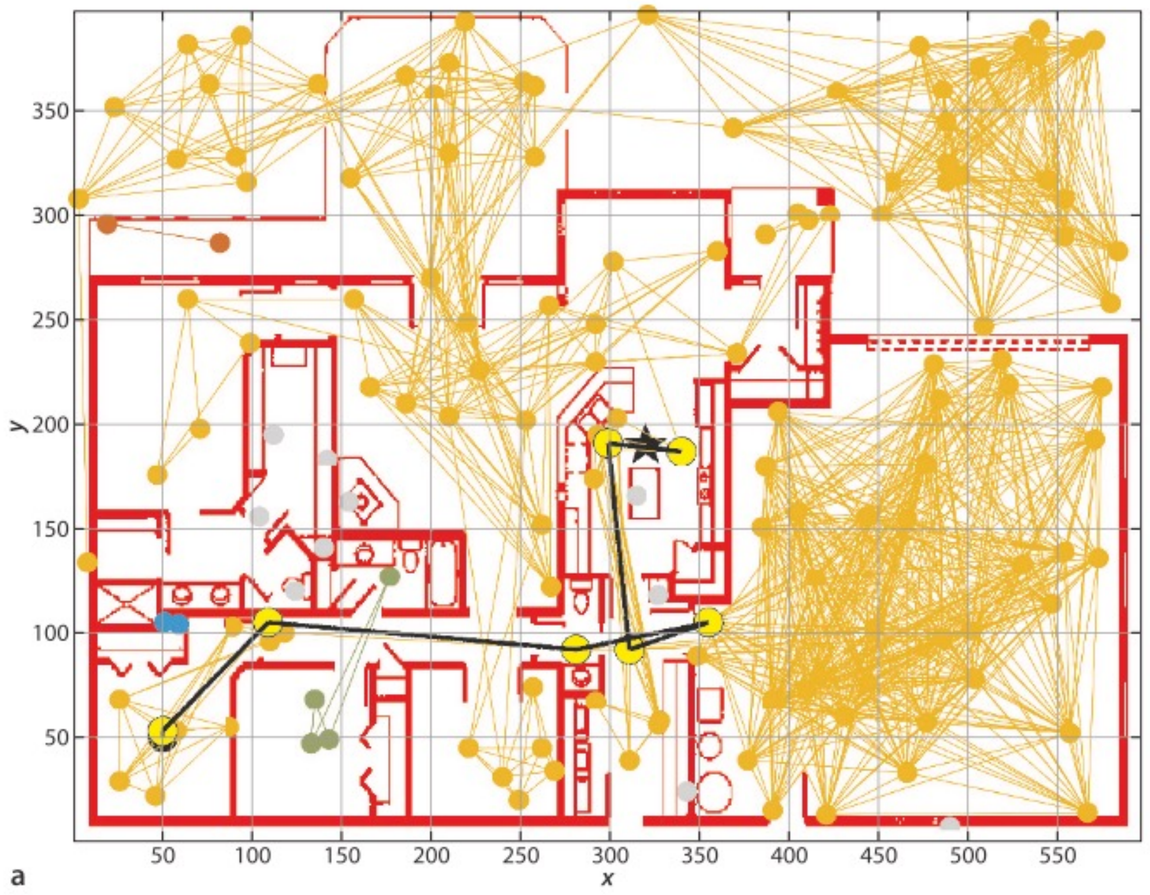
Roadmap definition

- The roadmap need only be computed once and can then be used to get us from any start location to any goal location.
- Split the space using Voronoi Diagrams



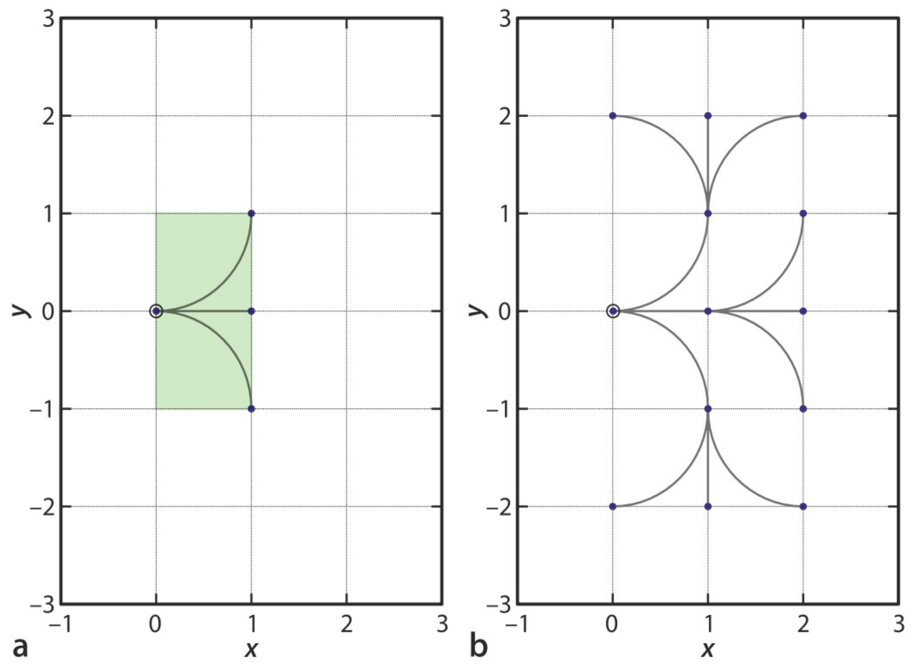


Probabilistic Roadmap Method (PRM)

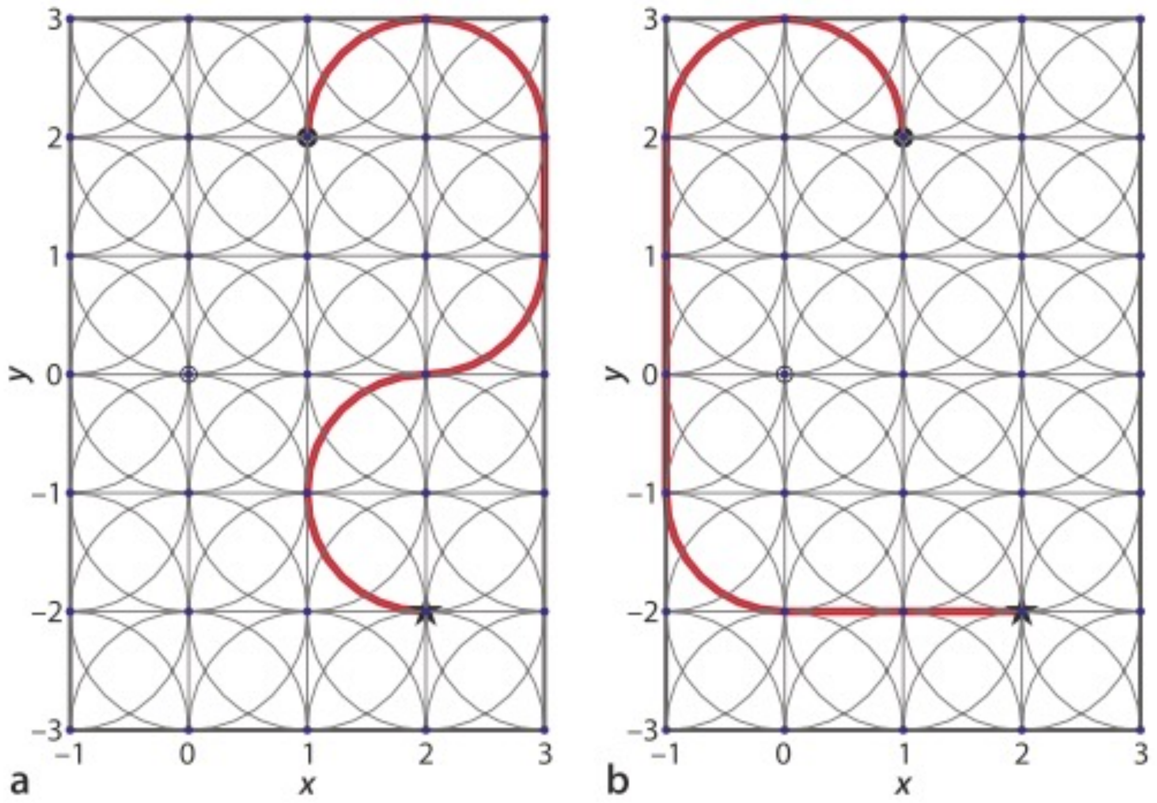


Lattice Methods

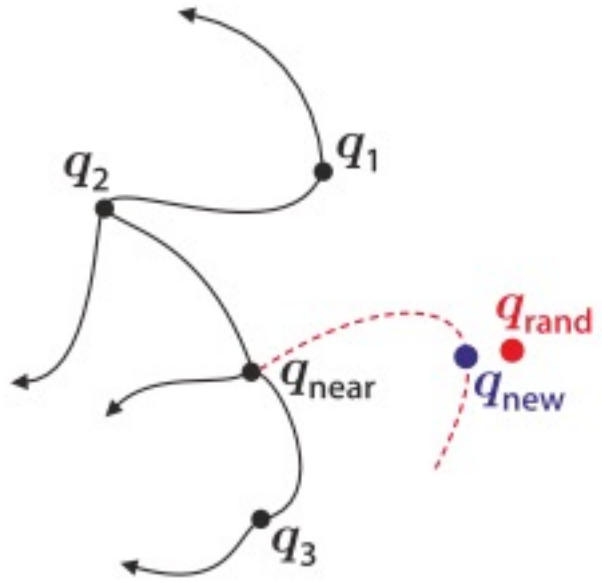
- Consider also the robot model

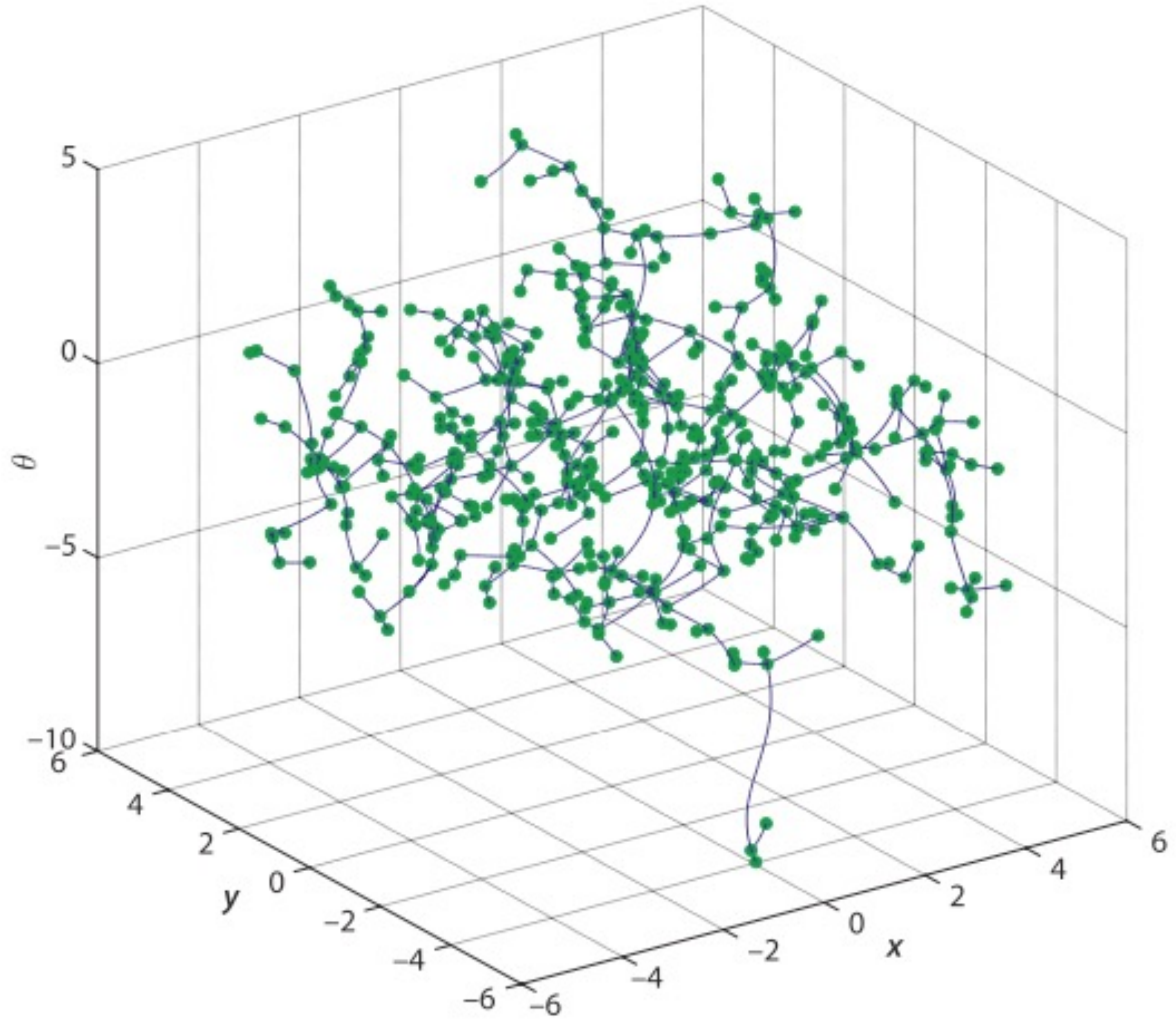


$\longrightarrow$



Rapidly-Exploring Random Tree (RRT)





Virtual Potential Fields

Virtual Potential Fields

- From physics we know that a potential field P(q) defined over C induces a force $\boldsymbol{F} = \frac{\partial \boldsymbol{P}}{\partial \boldsymbol{q}}$ that drives the object from a high to a low potential
- In robot motion control, the goal configuration q_{goal} is assigned a low virtual potential and obstacles are assigned a high virtual potential
- Applying a force to the robot proportional to the negative gradient of the virtual potential naturally pushes the robot toward the goal and away from the obstacles.
- Drawback: local minima

A Point in C-space

Potential Energy

$$P_{goal}(q) = \frac{1}{2}(q - q_{goal})^T K(q - q_{goal})$$

K symmetric positive-definite weighting matrix

Induced force

$$\mathbf{F_{goal}(q) = -\frac{\partial P_{goal}}{\partial q} = K(q_{goal} - q)}$$

Repulsive potential induced by a C-obstacle B

$$\mathcal{P}_{\mathcal{B}}(q) = \frac{k}{2d^2(q, \mathcal{B})}$$

d(q,B) is the distance from the obstacle

A Point in C-space

Force induced by the obstacle

$$F_{\mathcal{B}}(q) = -\frac{\partial \mathcal{P}_{\mathcal{B}}}{\partial q} = \frac{k}{d^3(q, \mathcal{B})} \frac{\partial d}{\partial q}$$

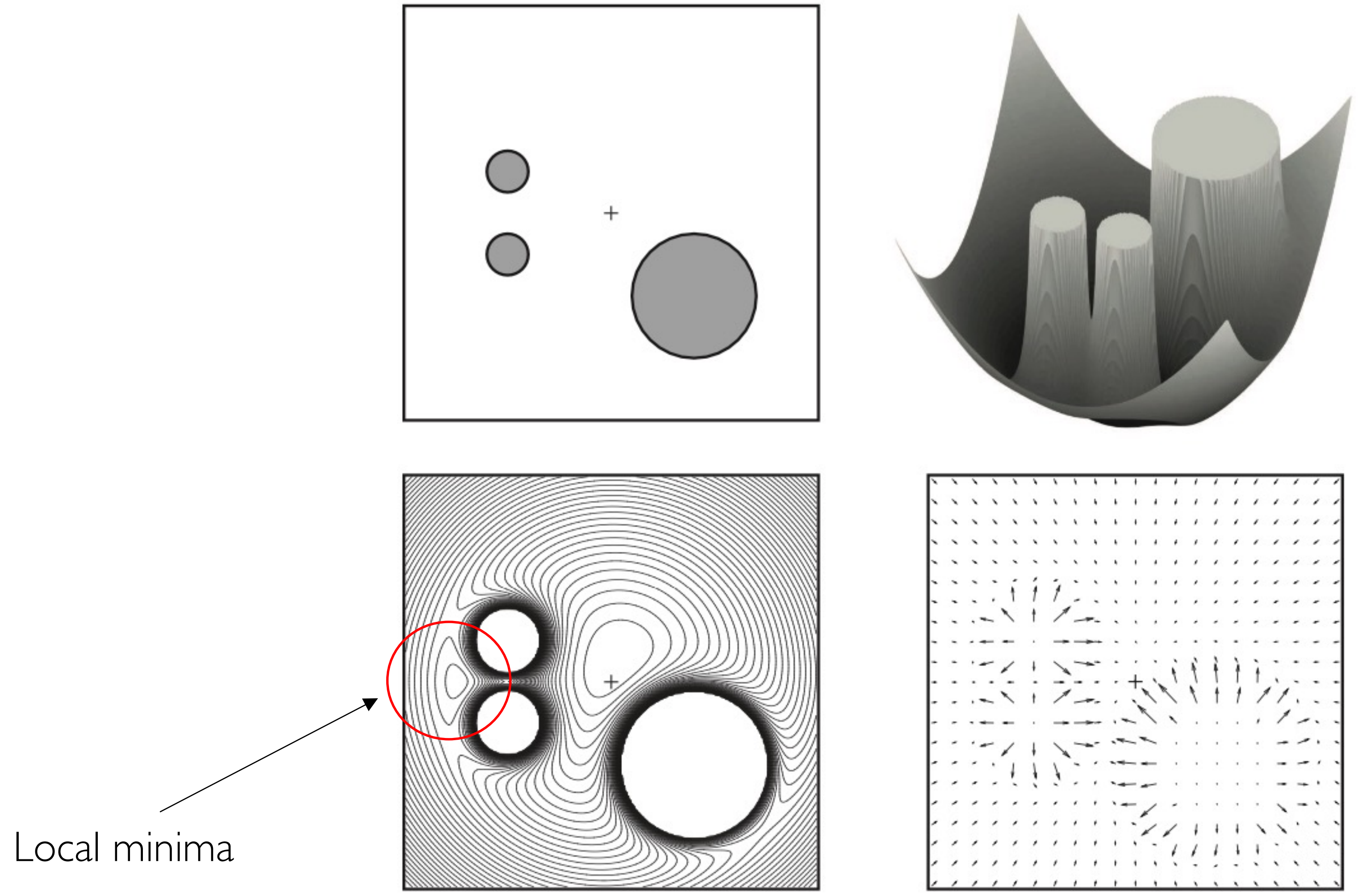
Total potential

$$\mathcal{P}(q) = \mathcal{P}_{\text{goal}}(q) + \sum_i \mathcal{P}_{\mathcal{B}_i}(q)$$

Total force

$$F(q) = F_{\text{goal}}(q) + \sum_i F_{\mathcal{B}_i}(q)$$

Representation of potential fields



Control the robot using the calculated F(q)

- Apply the calculated force plus damping

$$u = F(q) + B\dot{q}$$

- Treat the calculated force as a commanded velocity

$$\dot{q} = F(q)$$

- To remove effects of distant objects

$$U_{\mathcal{B}}(q) = \begin{cases} \frac{k}{2}\left(\frac{d_{\text{range}} - d(q,\mathcal{B})}{d_{\text{range}} d(q,\mathcal{B})}\right)^2 & \text{if } d(q,\mathcal{B}) < d_{\text{range}} \\ 0 & \text{otherwise.} \end{cases}$$

Wrapping up Navigation

- Robot navigation is the problem of guiding a robot towards a goal
- The simplest was the purely reactive Braitenberg-type vehicle. Then we added limited memory to create state machine based automata such as bug2 which can deal with obstacles, however the paths that it finds are far from optimal.
- Map-based planning algorithms. The distance transform is a computationally intense approach that finds an optimal path to the goal. D* also finds an optimal path, but supports a more nuanced travel cost – individual cells have a continuous traversability measure rather than being considered as only free space or obstacle. D* also supports computationally cheap incremental re-planning for small changes in the map.
- PRM reduces the computational burden significantly by probabilistic sampling but at the expense of somewhat less optimal paths. In particular it may not discover narrow routes between areas of free space.
- The lattice planner takes into account the motion constraints of a real vehicle to create paths which are feasible to drive, and can readily account for the orientation of the vehicle as well as its position.
- RRT is another random sampling method that also generates kinematically feasible paths. All the map-based approaches require a map and knowledge of the robot's location.
- Virtual potential fields are inspired by potential energy fields such as gravitational and electromagnetic fields. The goal point creates an attractive potential while obstacles create repulsive potentials. The robot is controlled by applying this force plus damping or by simulating first-order dynamics and driving the robot with F(q) as a velocity. Potential field methods are conceptually simple but may get the robot stuck in local minima away from the goal.