Motion Planning, Part II

Howie Choset

Motion Planning Statement

If **W** denotes the robot's workspace,

And $\mathbf{C_i}$ denotes the i'th obstacle,

Then the robot's free space, **FS,** is defined as:

$$\mathbf{FS = W \setminus (\bigcup C_i)}$$

And a path **c** is $\mathbf{c : [0,1] \rightarrow FS}$

where **c(0)** is $\mathbf{q_{start}}$ and **c(1)** is $\mathbf{q_{goal}}$

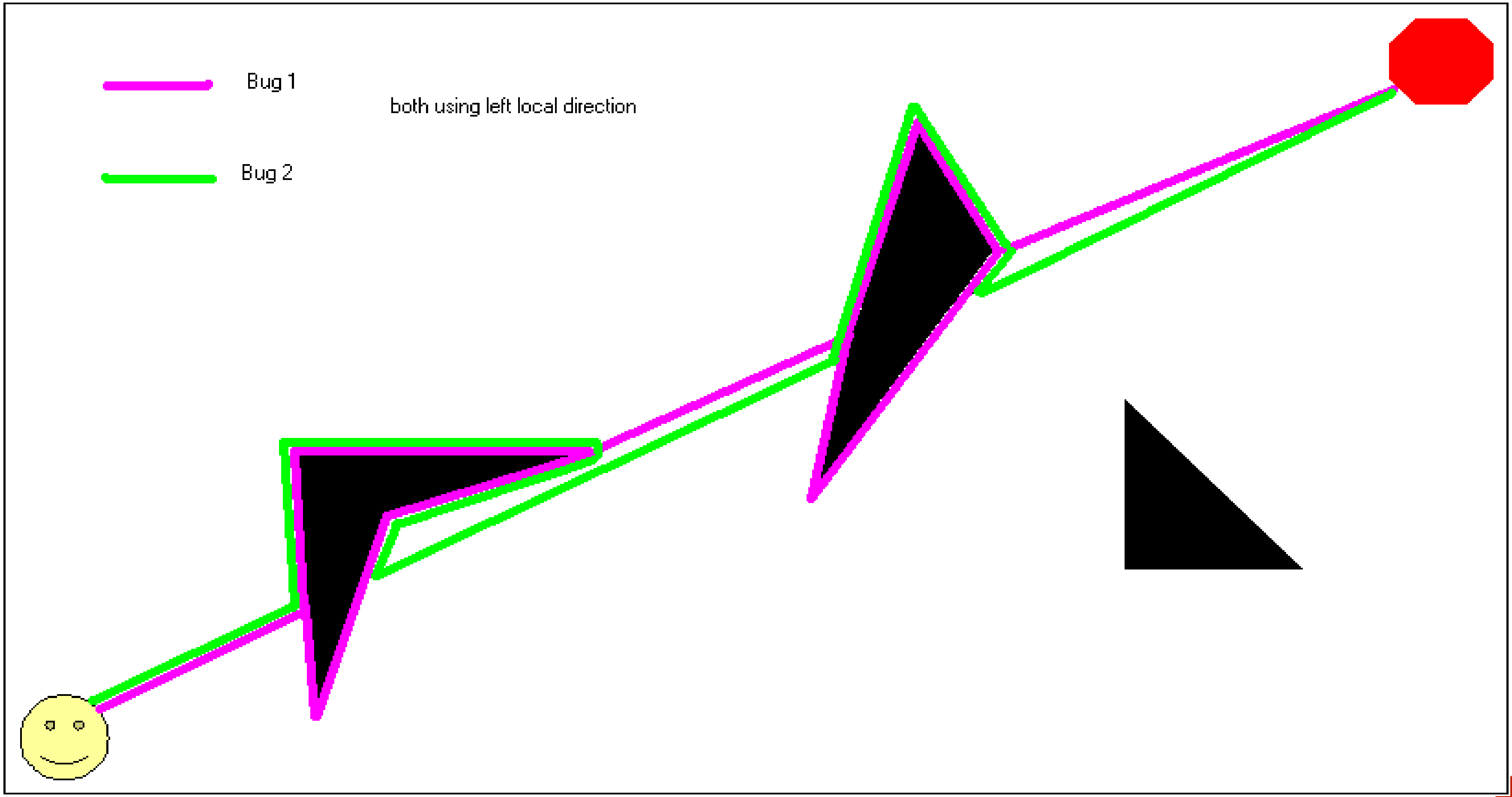
What is a good path?

Completeness

Basics: Metrics

- There are many different ways to measure a path:
  - Time
  - Distance traveled
  - Expense
  - Distance from obstacles
  - Etc…

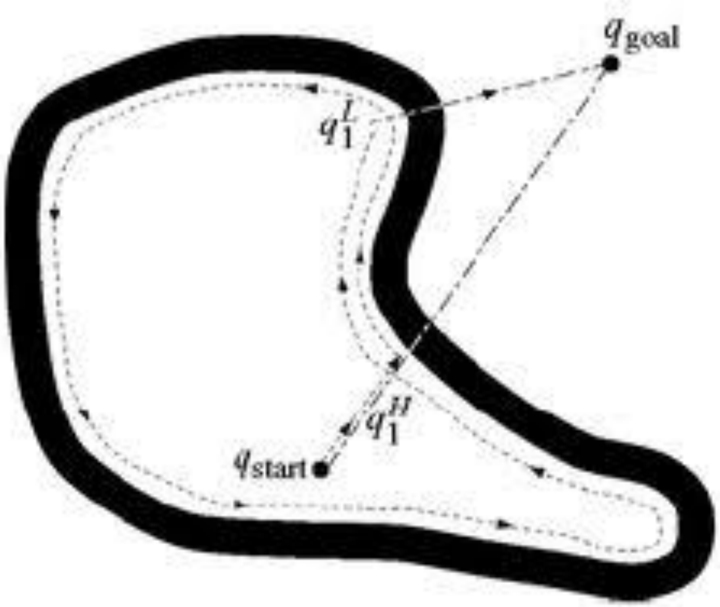
Start-Goal Algorithm: Lumelsky Bug Algorithms



Bug 1

But some computing power!
- known direction to goal
- otherwise local sensing

walls/obstacles & **encoders**



"Bug 1" algorithm

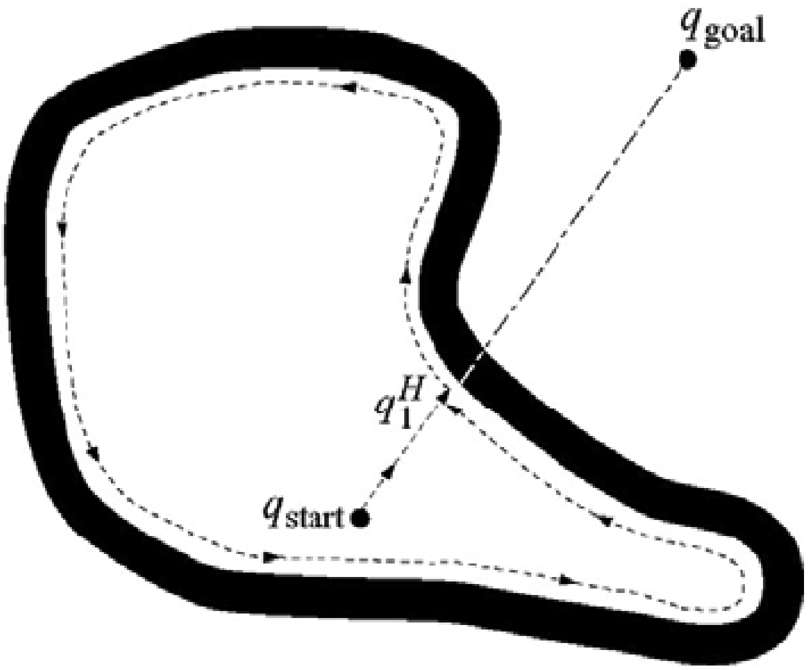
1) head toward goal

2) if an obstacle is encountered, circumnavigate it *and* remember how close you get to the goal

3) return to that closest point (by wall-following) and continue

Vladimir Lumelsky & Alexander Stepanov: Algorithmica 1987

A better bug?



"Bug 2" Algorithm

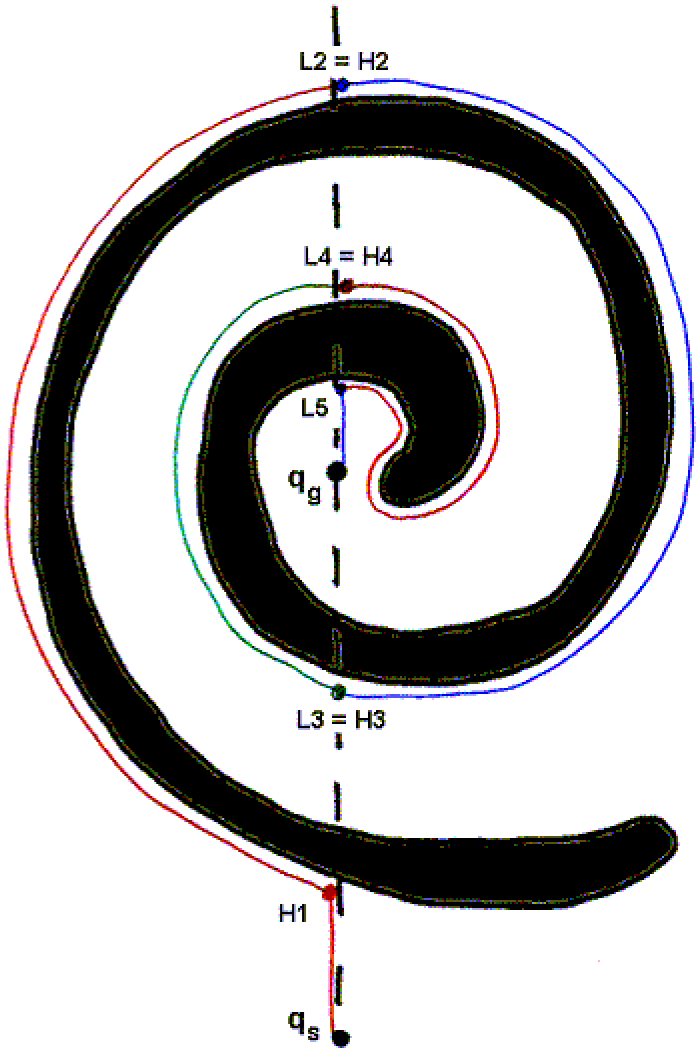
1) head toward goal on the *m-line*

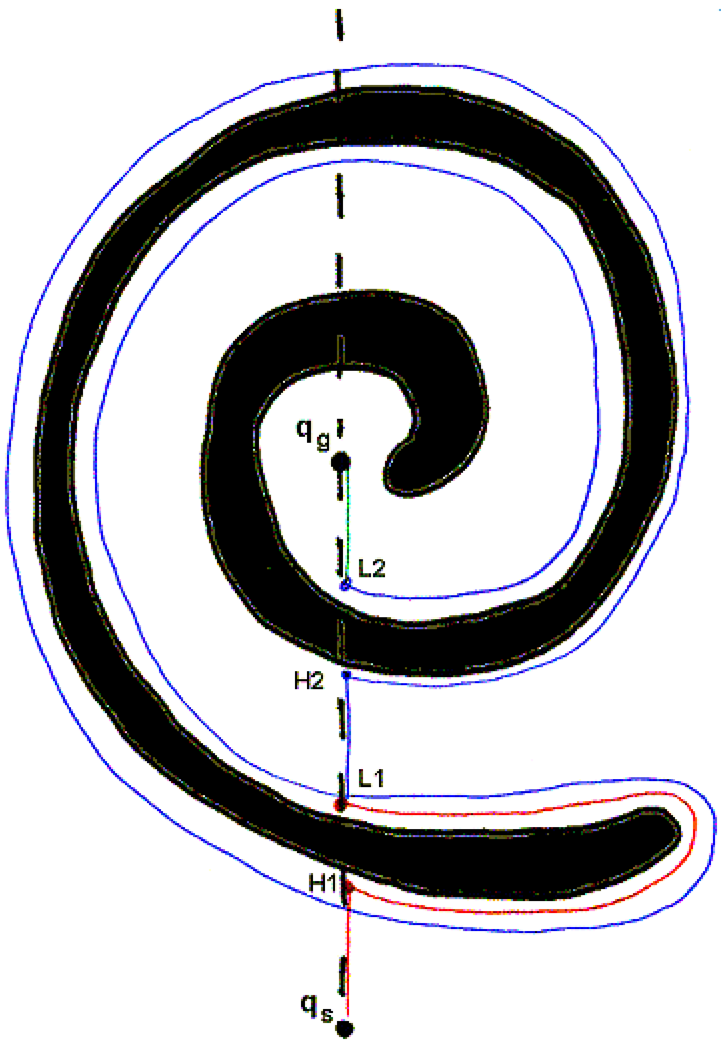
2) if an obstacle is in the way, follow it until you encounter the m-line again ***closer to the goal***.

3) Leave the obstacle and continue toward the goal

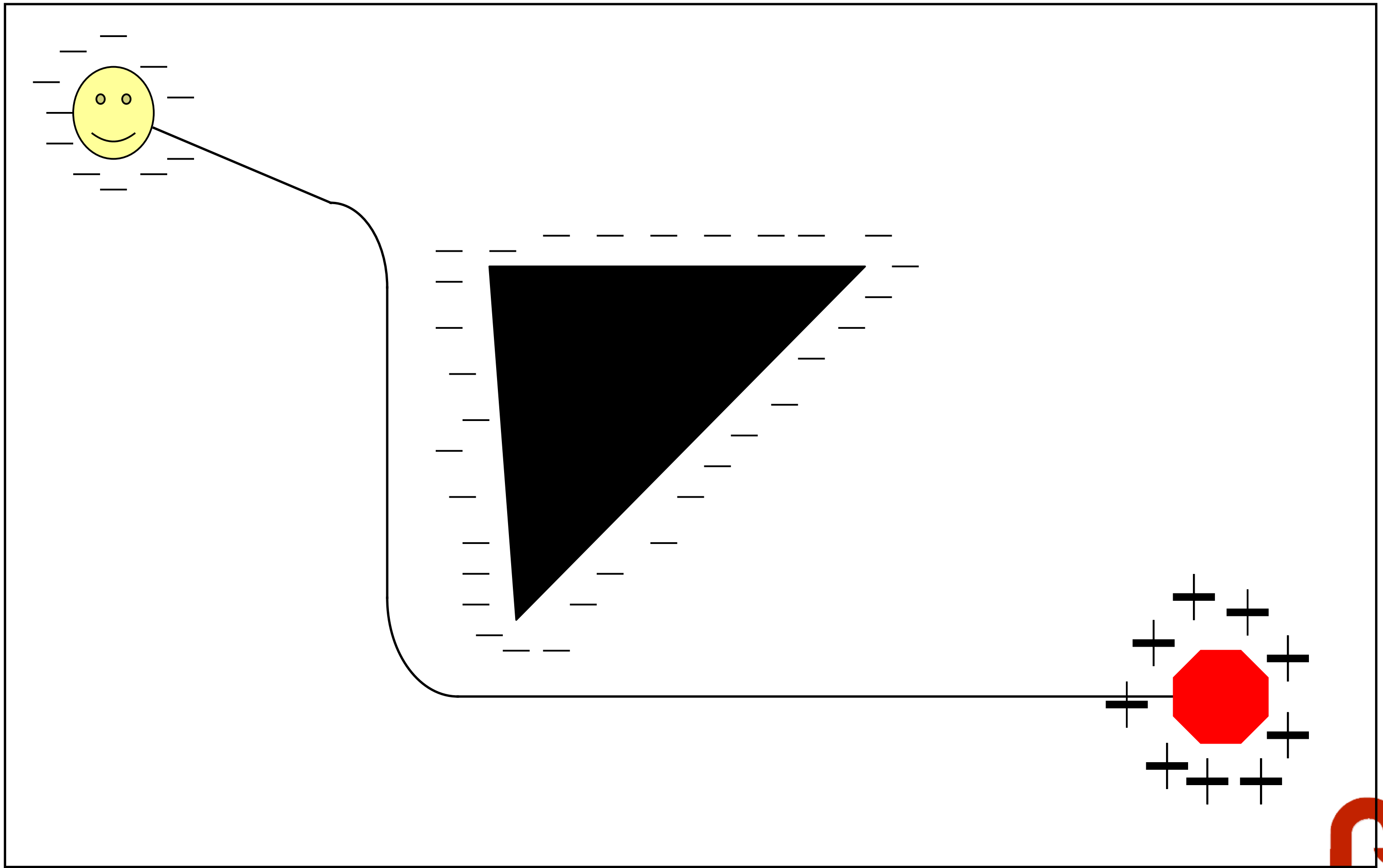
Better or worse than Bug1?

Bug 2 Spiral





Start-Goal Algorithm: Potential Functions



Attractive/Repulsive Potential Field

$$U(q) = U_{\mathrm{att}}(q) + U_{\mathrm{rep}}(q)$$

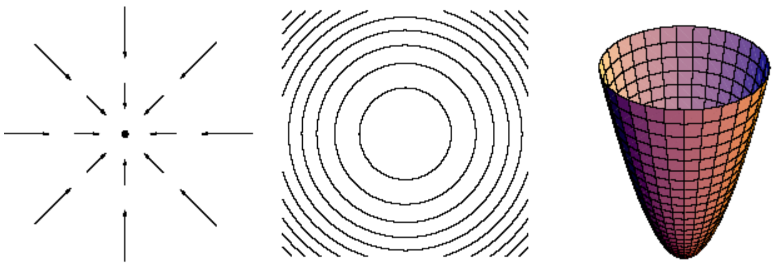
– U_{att} is the “attractive” potential --- move to the goal

– U_{rep} is the “repulsive” potential --- avoid obstacles

Artificial Potential Field Methods: Attractive Potential

Quadratic Potential

$$U_{\text{att}}(q) = \frac{1}{2}\zeta d^2(q, q_{\text{goal}}),$$



(a) (b) (c)

$$F_{\text{att}}(q) = \nabla U_{\text{att}}(q) = \nabla\left(\frac{1}{2}\zeta d^2\left(q, q_{\text{goal}}\right)\right),$$

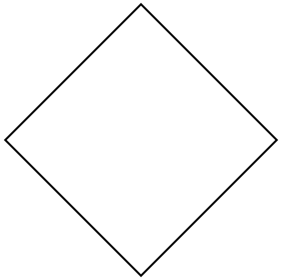
$$= \frac{1}{2}\zeta \nabla d^2(q, q_{\text{goal}}),$$

$$= \zeta(q - q_{\text{goal}}),$$

Distance

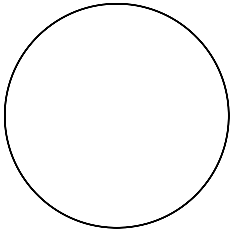
$$d : R^2 \times R^2 \rightarrow R$$

L1 Metric



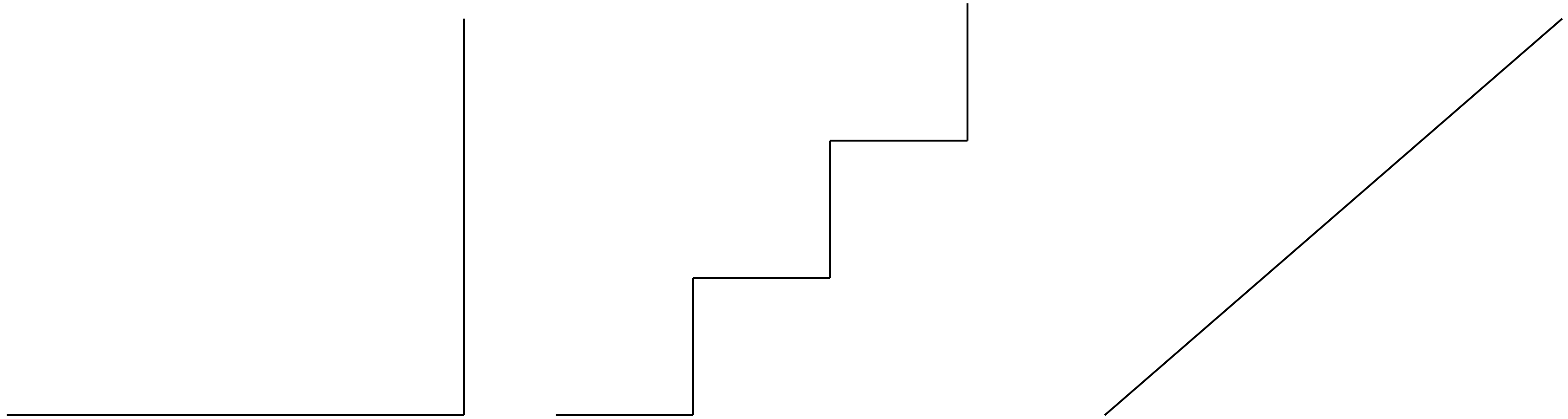
$$d(a,b) = |a_x - b_x| + |a_y - b_y|$$

L2 Metric

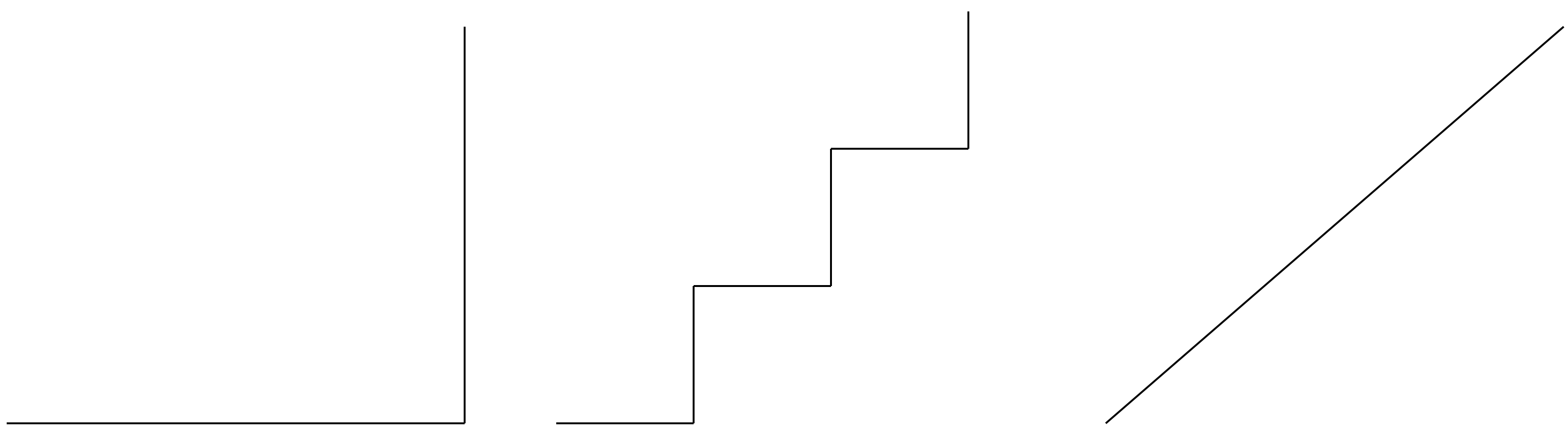


$$d(a,b) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}$$

Path Length
Which is shortest?



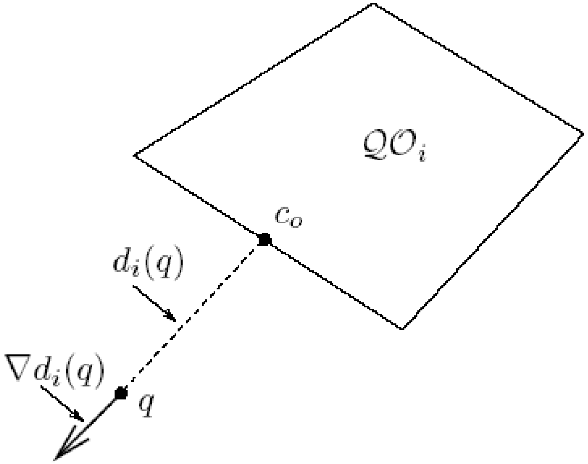
Path Length
Depends on metric



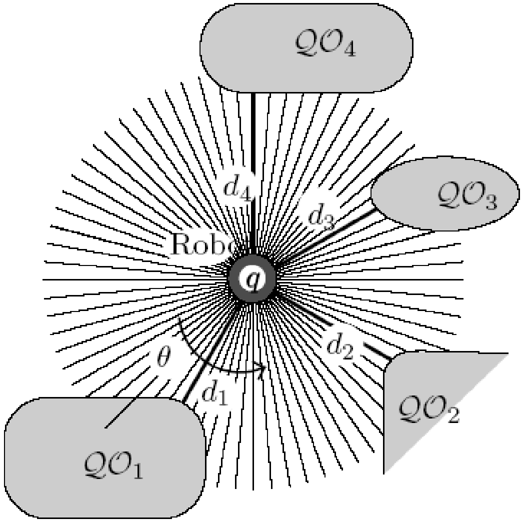
Distance to Obstacle(s)

$$d_i(q) = \min_{c \in \mathcal{QO}_i} d(q, c)$$

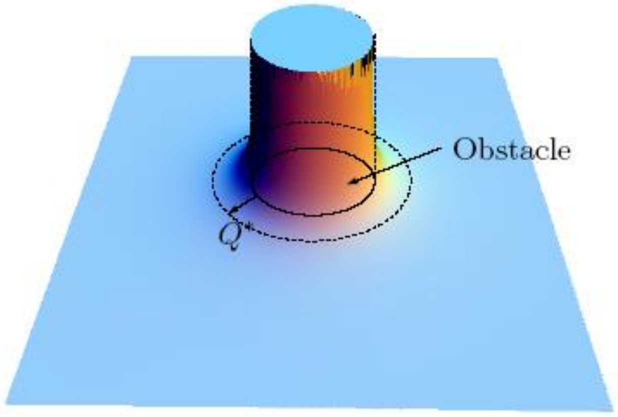
$$\nabla d_i(q) = \frac{q - c}{d(q, c)}$$



$$D(q) = \min d_i(q)$$



The Repulsive Potential



$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{D(q)} - \frac{1}{Q^*}\right)^2, & D(q) \leq Q^*, \\ 0, & D(q) > Q^*, \end{cases}$$

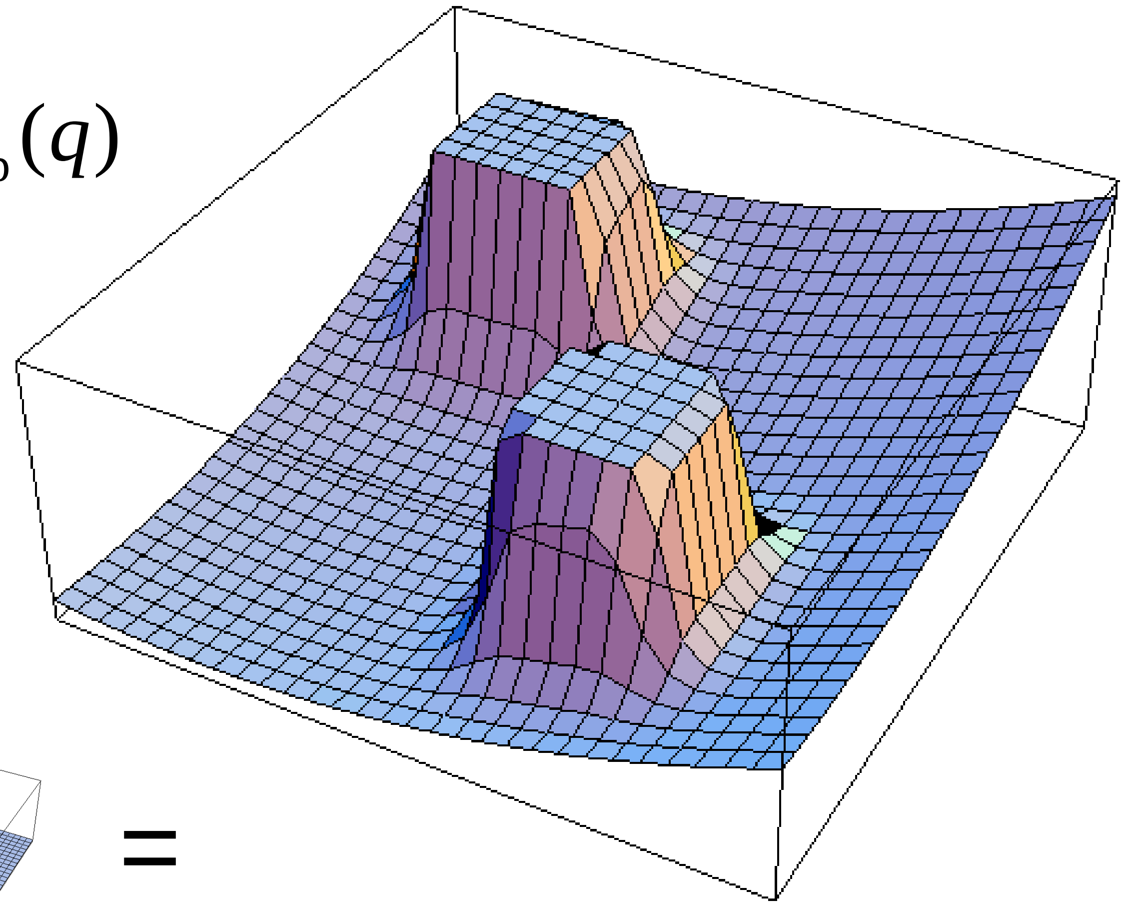
whose gradient is

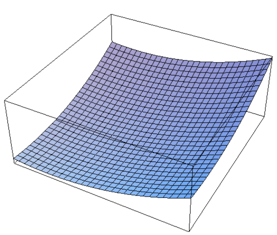
$$\nabla U_{\text{rep}}(q) = \begin{cases} \eta\left(\frac{1}{Q^*} - \frac{1}{D(q)}\right)\frac{1}{D^2(q)}\nabla D(q), & D(q) \leq Q^*, \\ 0, & D(q) > Q^*, \end{cases}$$

Total Potential Function

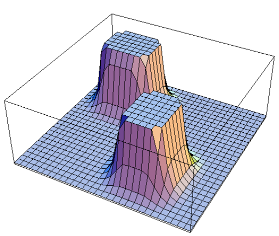
$$U(q) = U_{att}(q) + U_{rep}(q)$$

$$F(q) = -\nabla U(q)$$



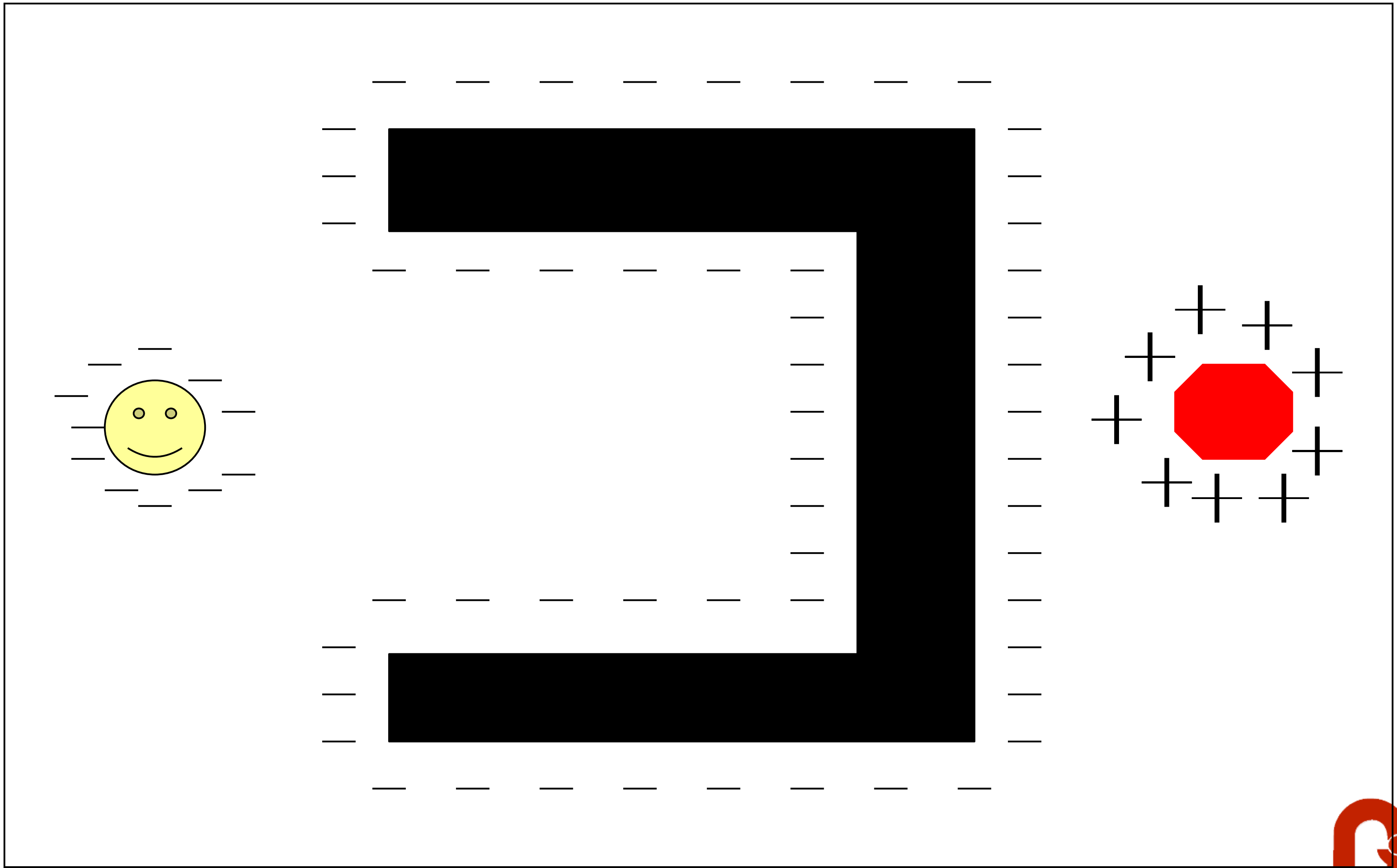


+



=

Local Minimum Problem with the Charge Analogy

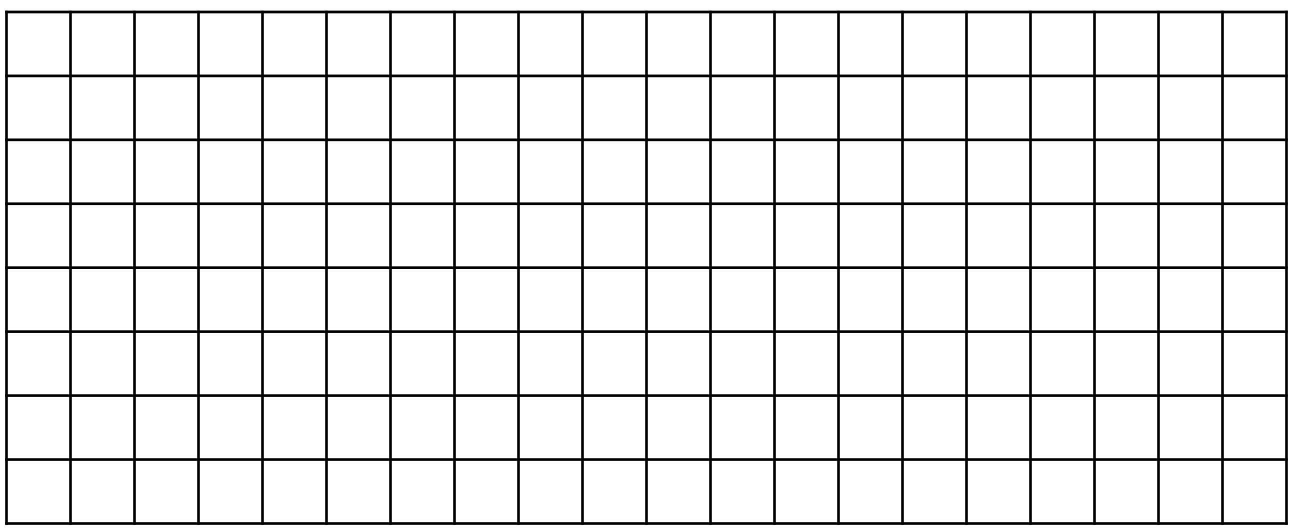


The Wavefront Planner

- A common algorithm used to determine the shortest paths between two points
  - In essence, a breadth first search of a graph
- For simplification, we'll present the world as a two-dimensional grid
- Setup:
  - Label free space with 0
  - Label start as START
  - Label the destination as 2

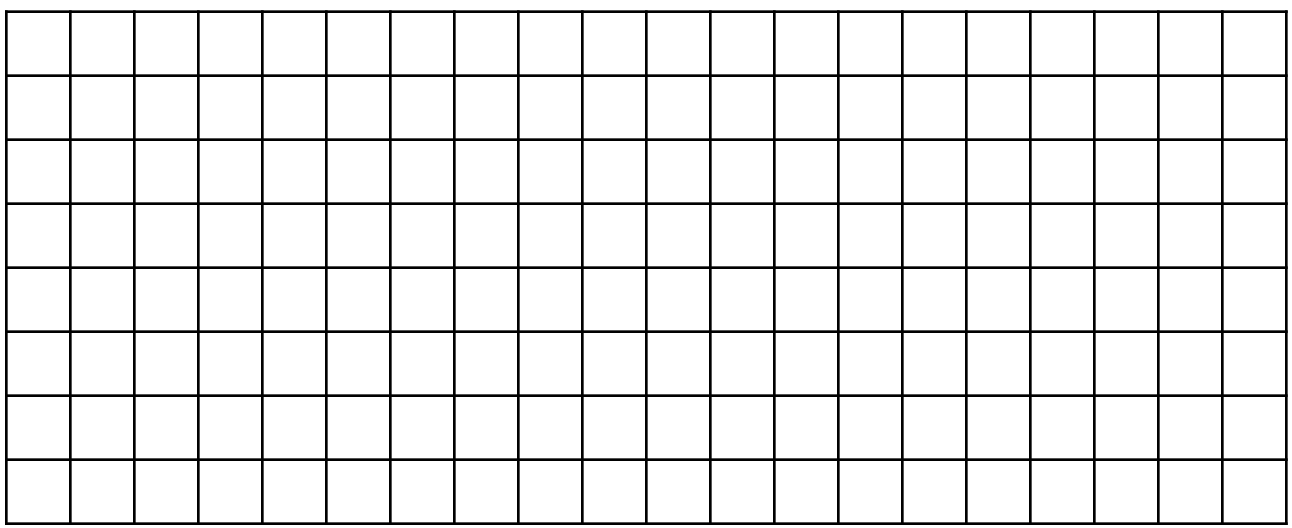
Representations

- World Representation
  - You could always use a large region and distances
  - However, a grid can be used for simplicity



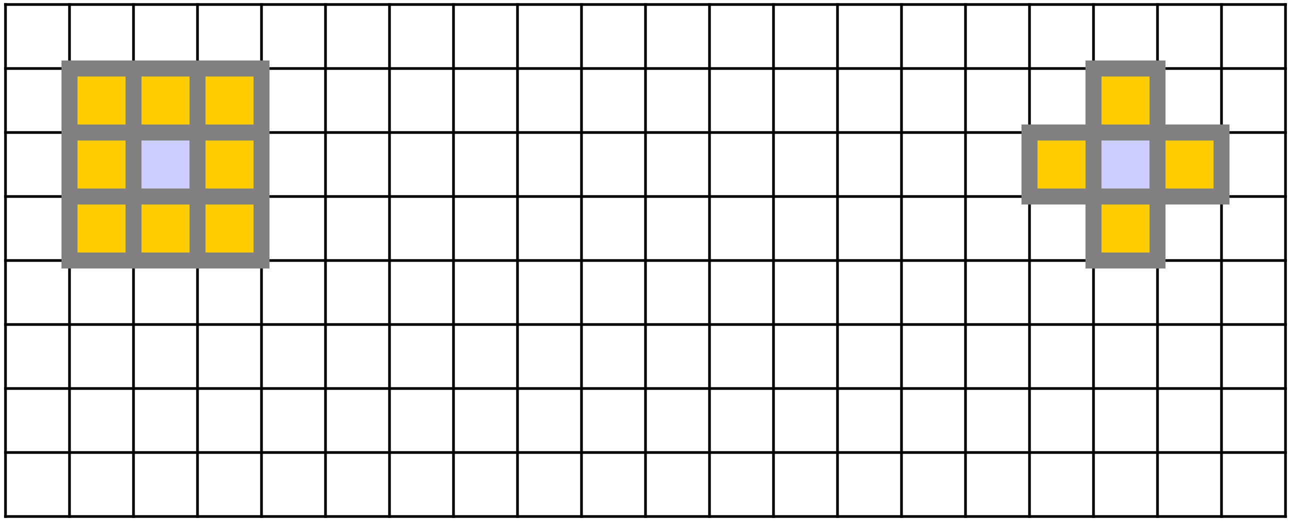
Representations: A Grid

- Distance is reduced to discrete steps
  - For simplicity, we'll assume distance is uniform
- Direction is now limited from one adjacent cell to another
  - Time to revisit Connectivity (Remember Vision?)



Representations: Connectivity

- 8-Point Connectivity
- 4-Point Connectivity
  - *(approximation of the L1 metric)*



The Wavefront Planner: Setup

7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
3	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront in Action (Part 1)

- Starting with the goal, set all adjacent cells with “0” to the current cell + 1
  - 4-Point Connectivity or 8-Point Connectivity?
  - Your Choice. We’ll use 8-Point Connectivity in our example

7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront in Action (Part 2)

- Now repeat with the modified cells
  - This will be repeated until no 0's are adjacent to cells with values >= 2
    - 0's will only remain when regions are unreachable

7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
3	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4
1	0	0	0	0	0	0	0	0	0	0	0	0	0	4	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	3	2
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront in Action (Part 3)

- Repeat again...

7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	0	0	0	0	0	0	0	0	0	0	0	0	5	4	4	4
1	0	0	0	0	0	0	0	0	0	0	0	0	5	4	3	3
0	0	0	0	0	0	0	0	0	0	0	0	0	5	4	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront in Action (Part 4)

- And again...

7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	6	6	6	6
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	0	0	0	0	0	0	0	0	0	0	0	6	5	4	4	4
1	0	0	0	0	0	0	0	0	0	0	0	6	5	4	3	3
0	0	0	0	0	0	0	0	0	0	0	0	6	5	4	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront in Action (Part 5)

- And again until...

7	**0**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	7	7	7	7	7
4	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	6	6	6	6
3	0	0	0	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	0	0	0	0	0	0	0	0	0	0	7	6	5	4	4	4
1	0	0	0	0	0	0	0	0	0	0	7	6	5	4	3	3
0	0	0	0	0	0	0	0	0	0	0	7	6	5	4	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The Wavefront in Action (Done)

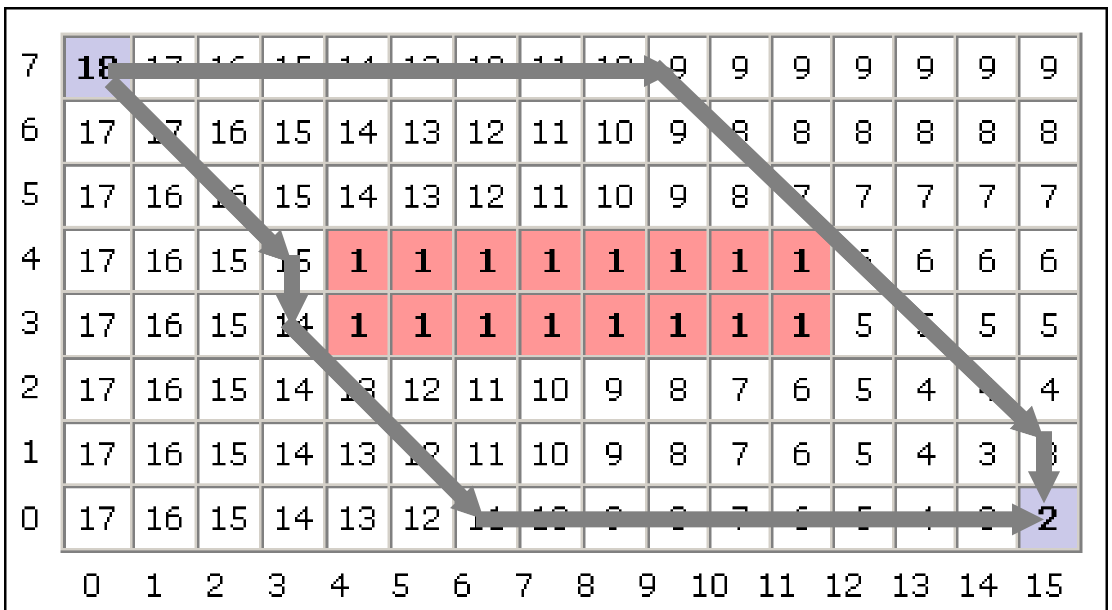
- You're done
  - Remember, 0's should only remain if unreachable regions exist

7	**18**	17	16	15	14	13	12	11	10	9	9	9	9	9	9	9
6	17	17	16	15	14	13	12	11	10	9	8	8	8	8	8	8
5	17	16	16	15	14	13	12	11	10	9	8	7	7	7	7	7
4	17	16	15	15	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	6	6	6	6
3	17	16	15	14	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	5	5	5	5
2	17	16	15	14	13	12	11	10	9	8	7	6	5	4	4	4
1	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	3
0	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	**2**
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

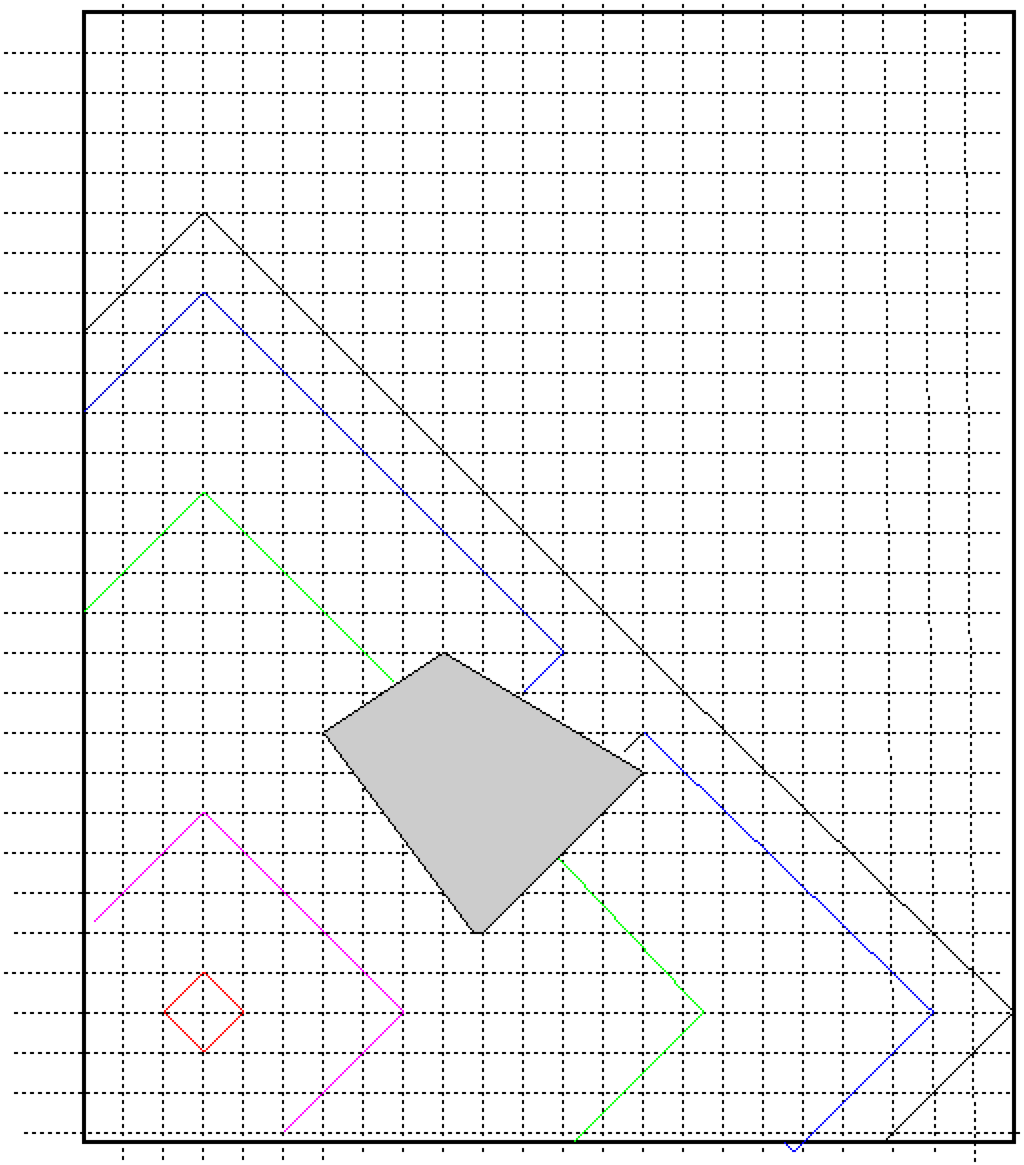
The Wavefront, Now What?

- To find the shortest path, according to your metric, simply always move toward a cell with a lower number
  - The numbers generated by the Wavefront planner are roughly proportional to their distance from the goal

Two possible shortest paths shown



This is really a Continuous Solution

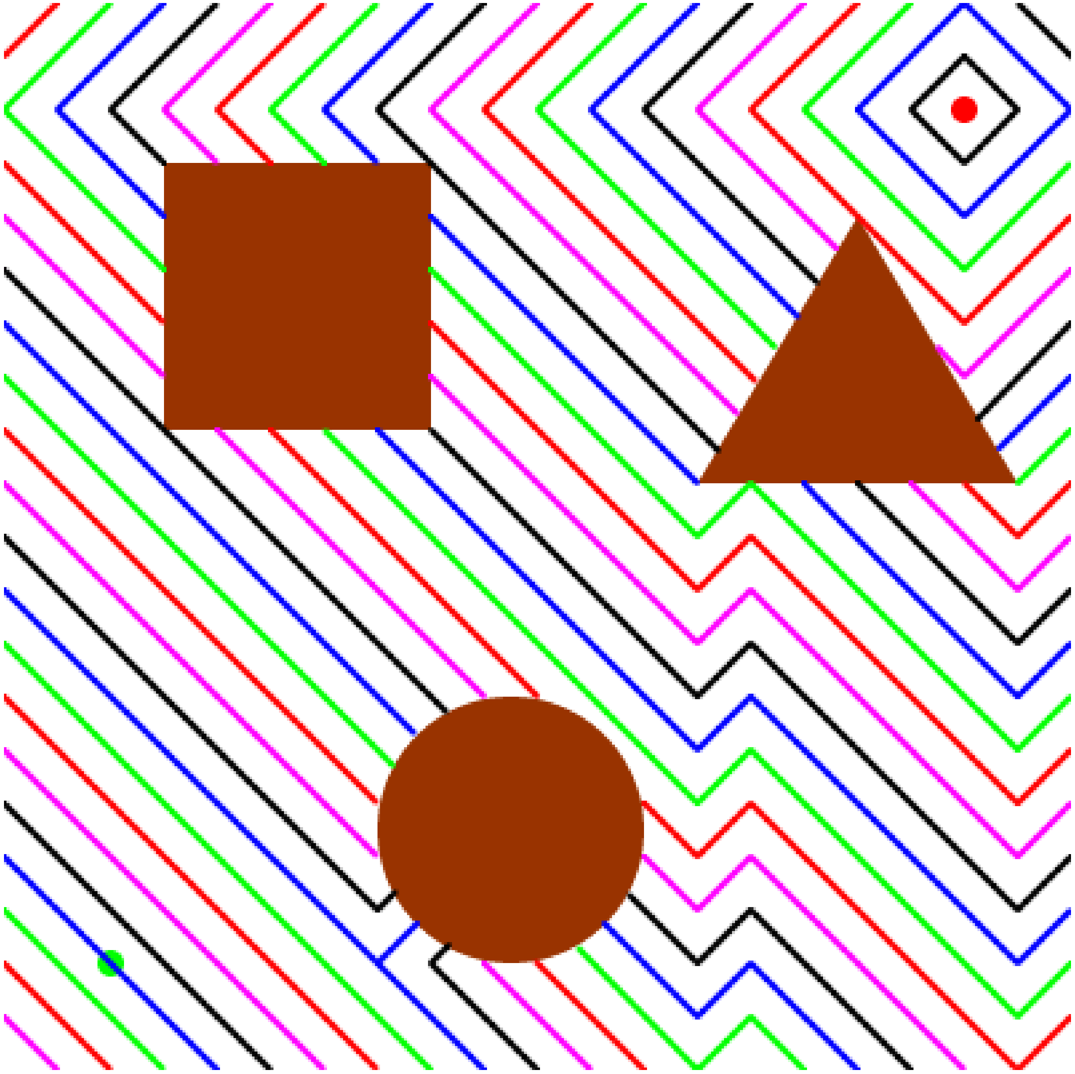


Not pixels

Waves bend

L1 distance

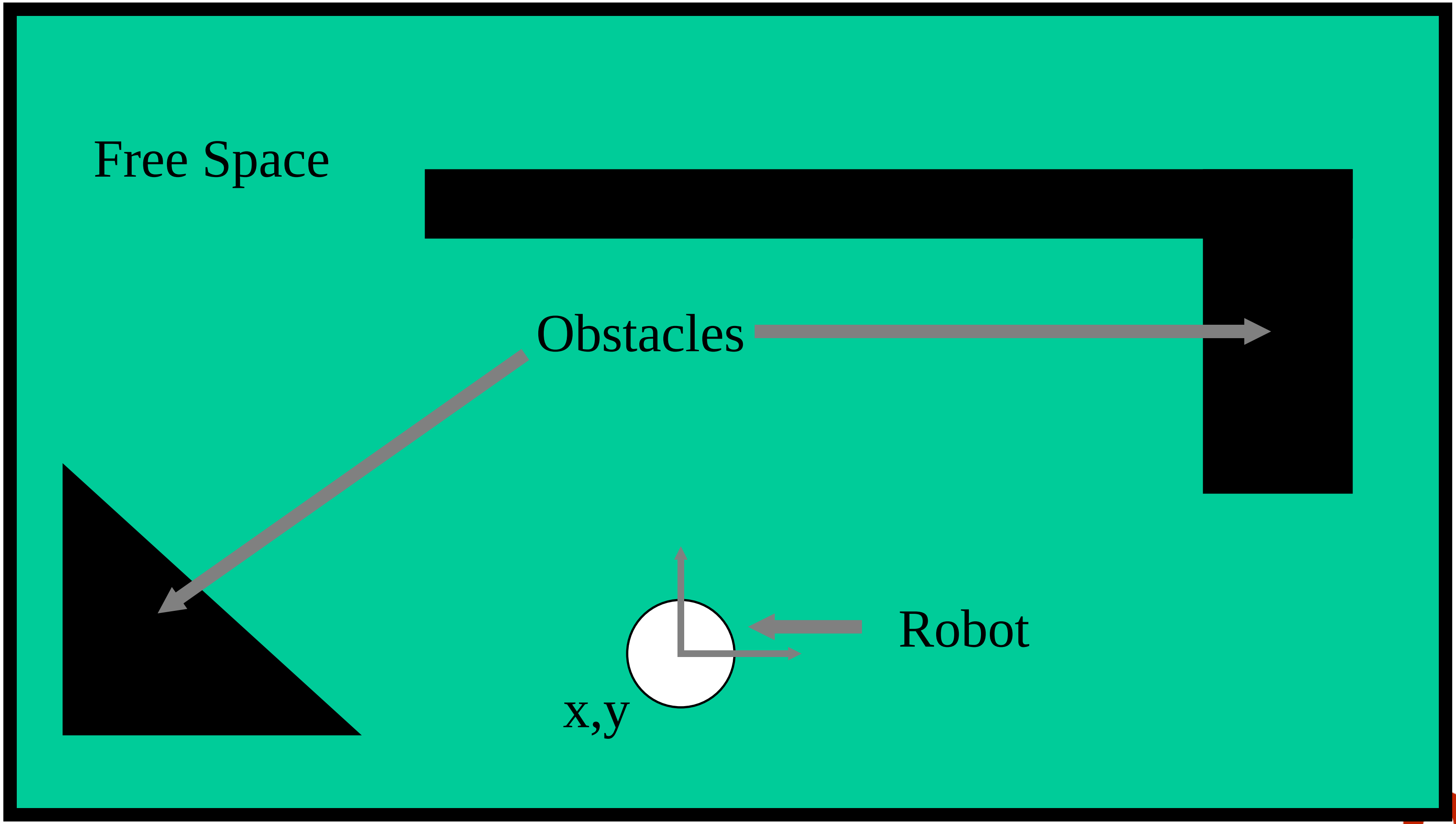
Pretty Wavefront



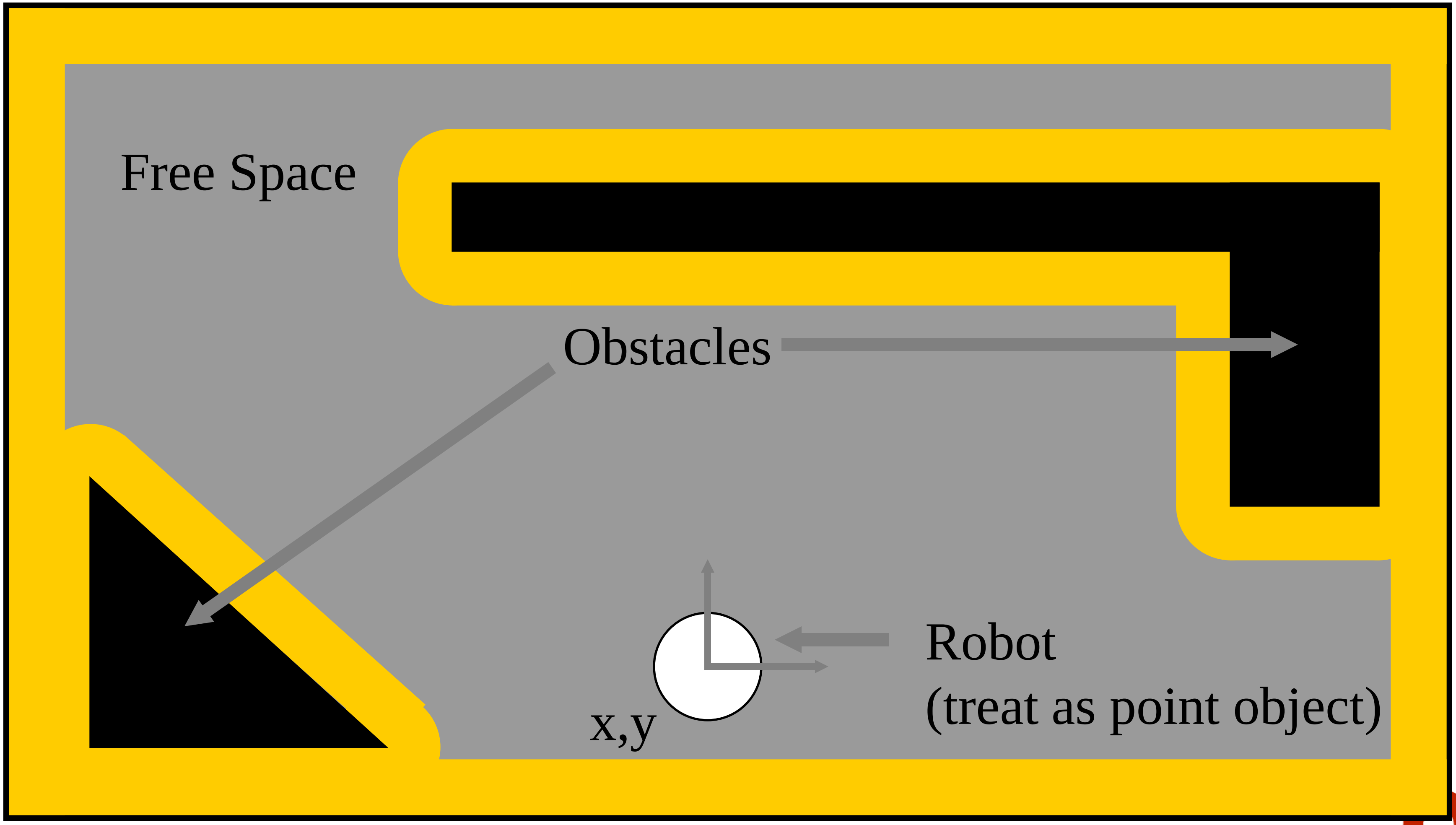
The Configuration Space

- What it is
  - A set of “reachable” areas constructed from knowledge of both the robot and the world
- How to create it
  - First abstract the robot as a point object. Then, enlarge the obstacles to account for the robot’s footprint and degrees of freedom
  - In our example, the robot was circular, so we simply enlarged our obstacles by the robot’s radius (*note the curved vertices*)

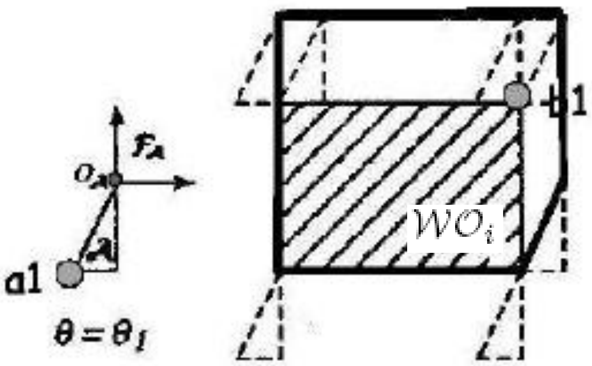
Example of a World (and Robot)

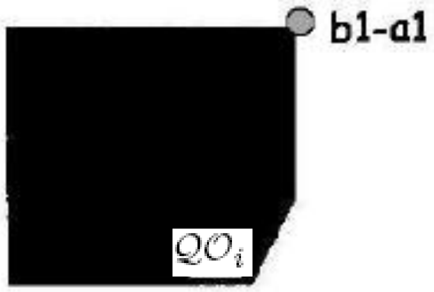


Configuration Space: Accommodate Robot Size



Translate-only, non-circularly

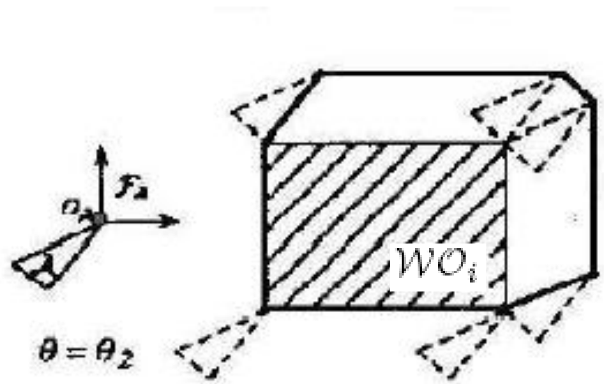


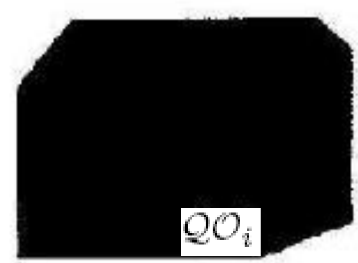


$$\mathcal{QO}_i = \{q \in \mathcal{Q} \mid R(q) \bigcap \mathcal{WO}_i \neq \emptyset\}.$$

Pick a reference point…

Translate-only, non-circularly symmetric

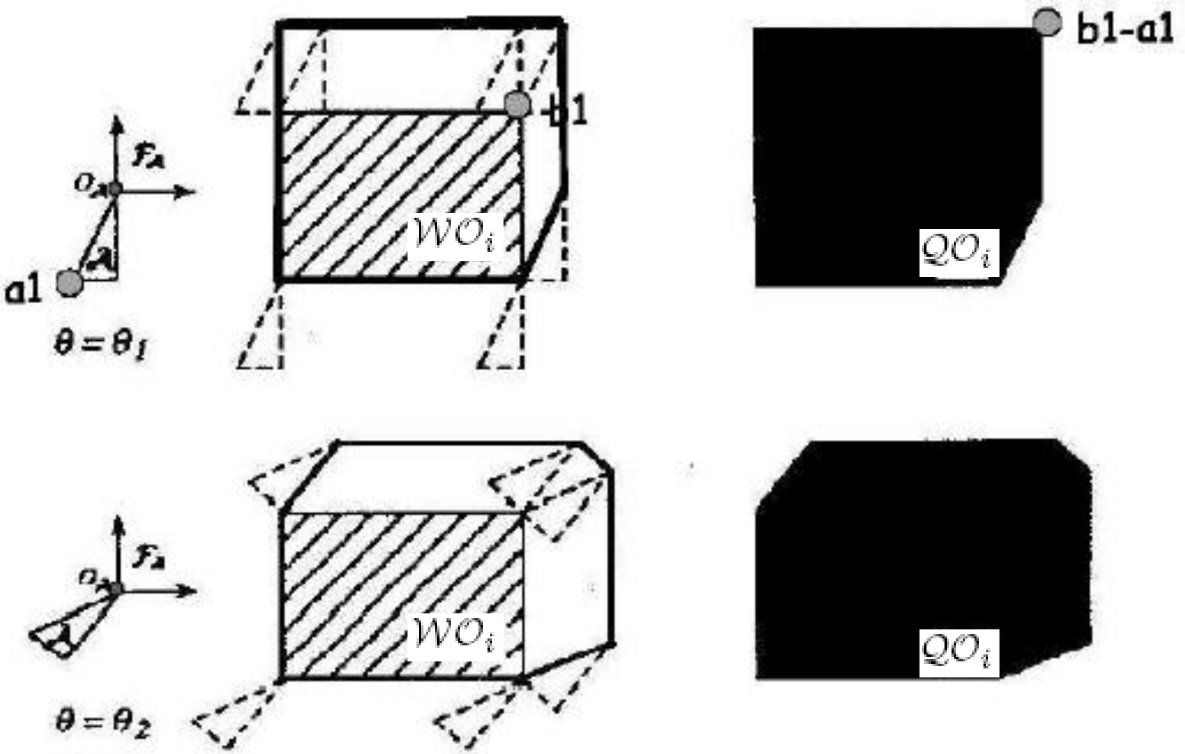




$$\mathcal{QO}_i = \{q \in \mathcal{Q} \mid R(q) \bigcap \mathcal{WO}_i \neq \emptyset\}.$$

Pick a reference point…

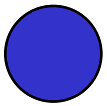
With Rotation: how much distance to rotate

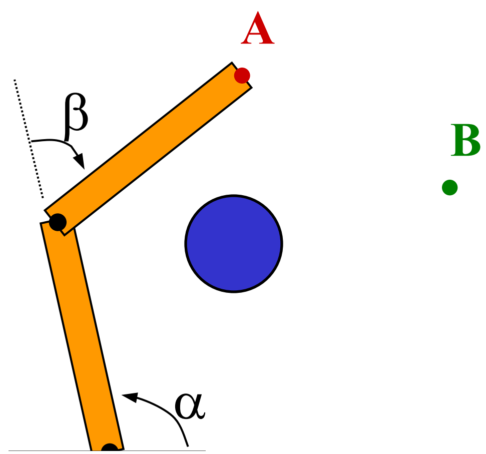


$$\mathcal{QO}_i = \{q \in \mathcal{Q} \mid R(q) \bigcap \mathcal{WO}_i \neq \emptyset\}.$$

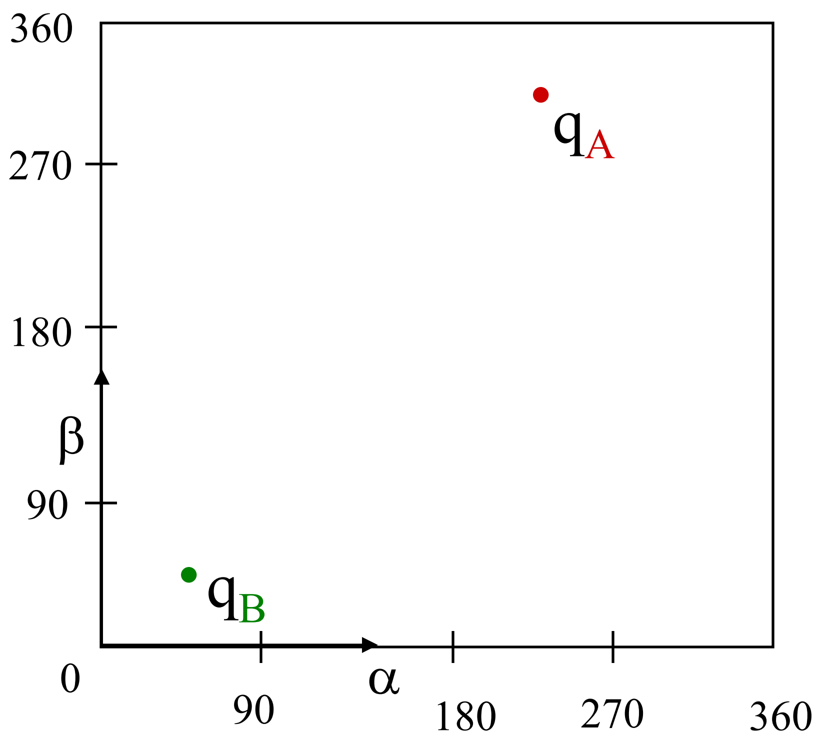
Pick a reference point…

Configuration Space "Quiz"

Where do we put  ?



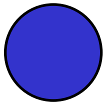
An obstacle in the robot's workspace

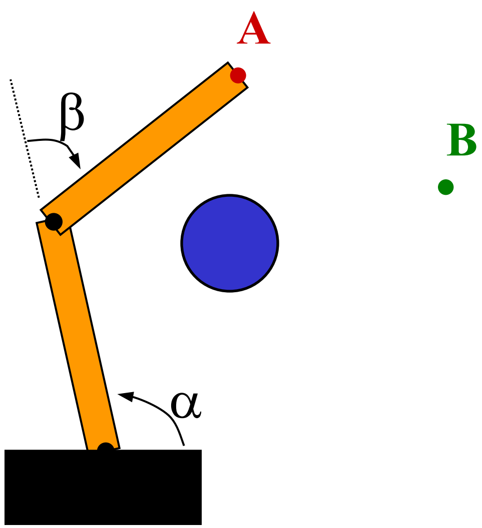


Torus

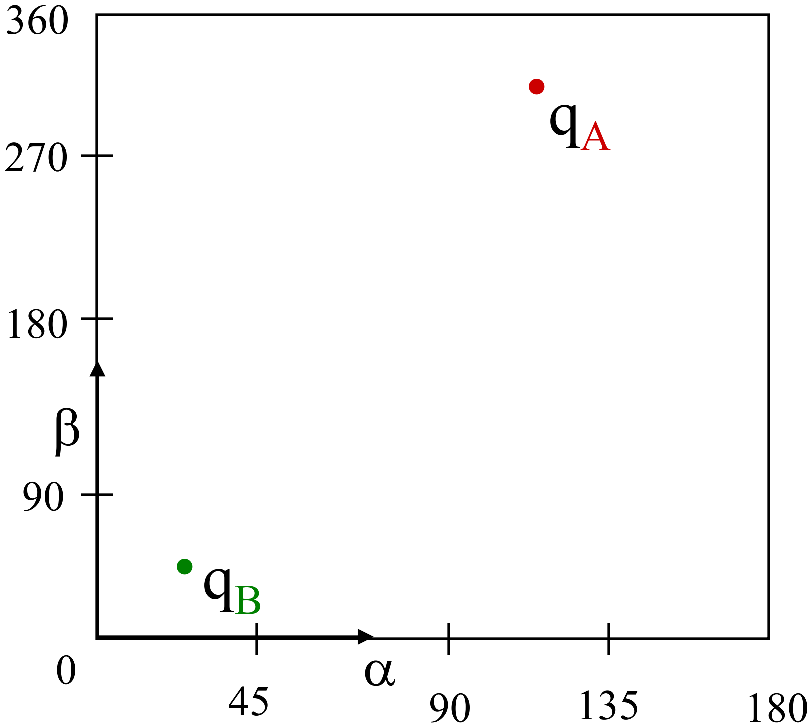
(wraps horizontally and vertically)

Configuration Space "Quiz"

Where do we put  ?



An obstacle in the robot's workspace



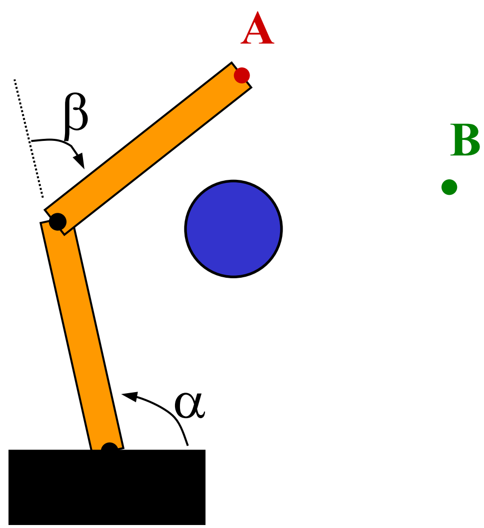
Torus

(wraps vertically)

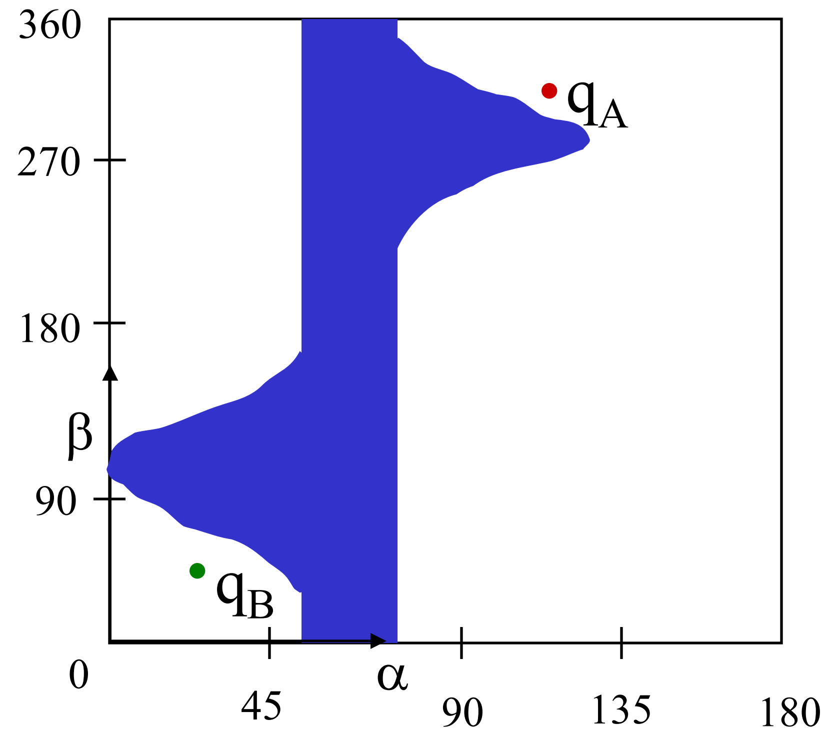
Configuration Space Obstacle

Reference *configuration*

How do we get from A to B ?

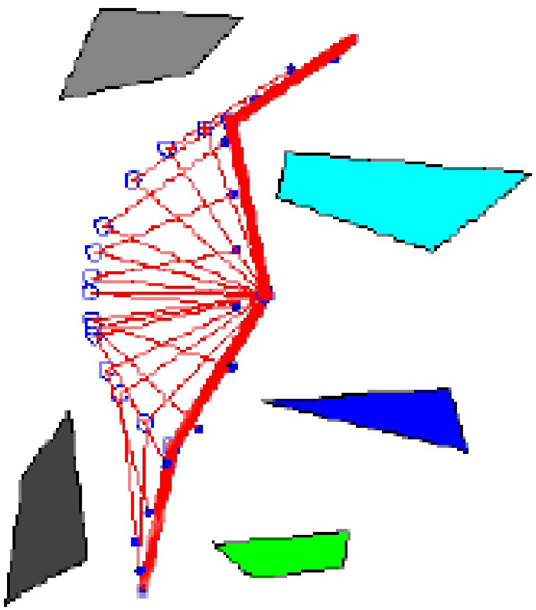


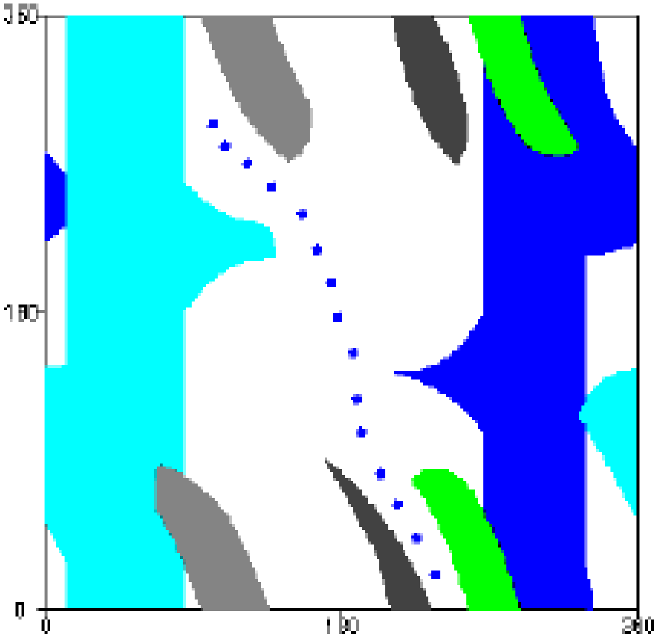
An obstacle in the robot's workspace



The C-space representation of this obstacle…

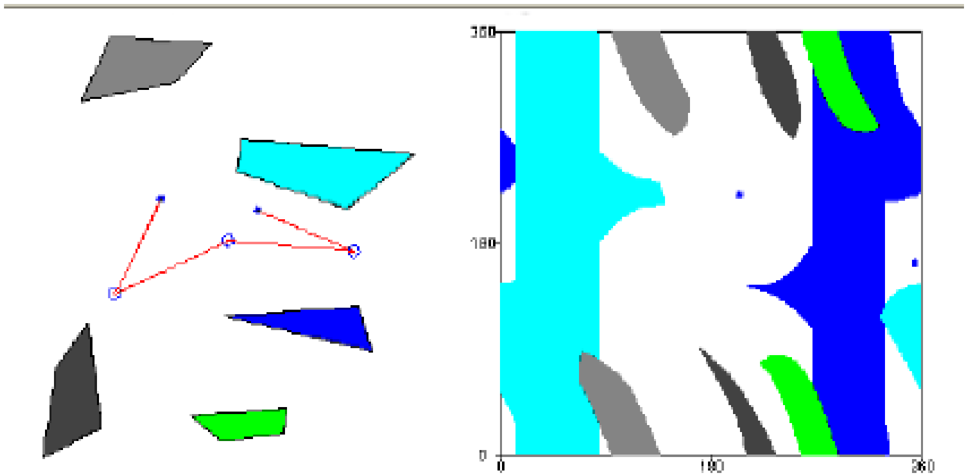
Two Link Path

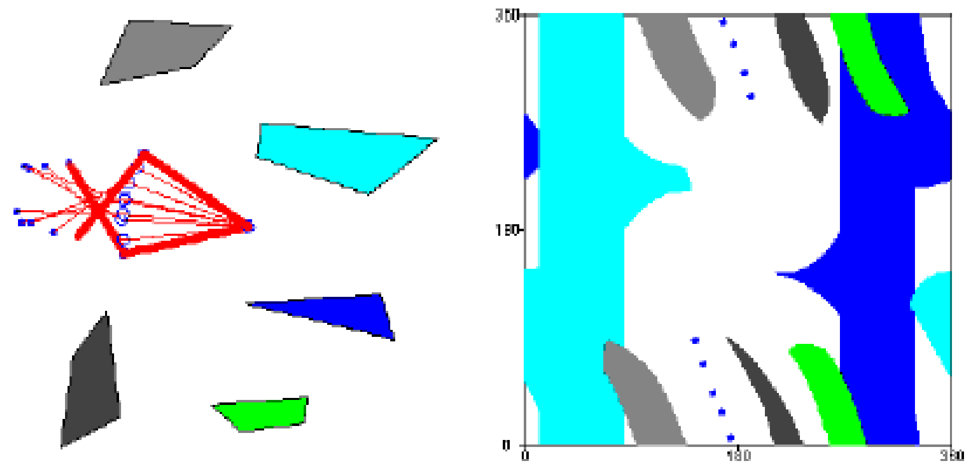




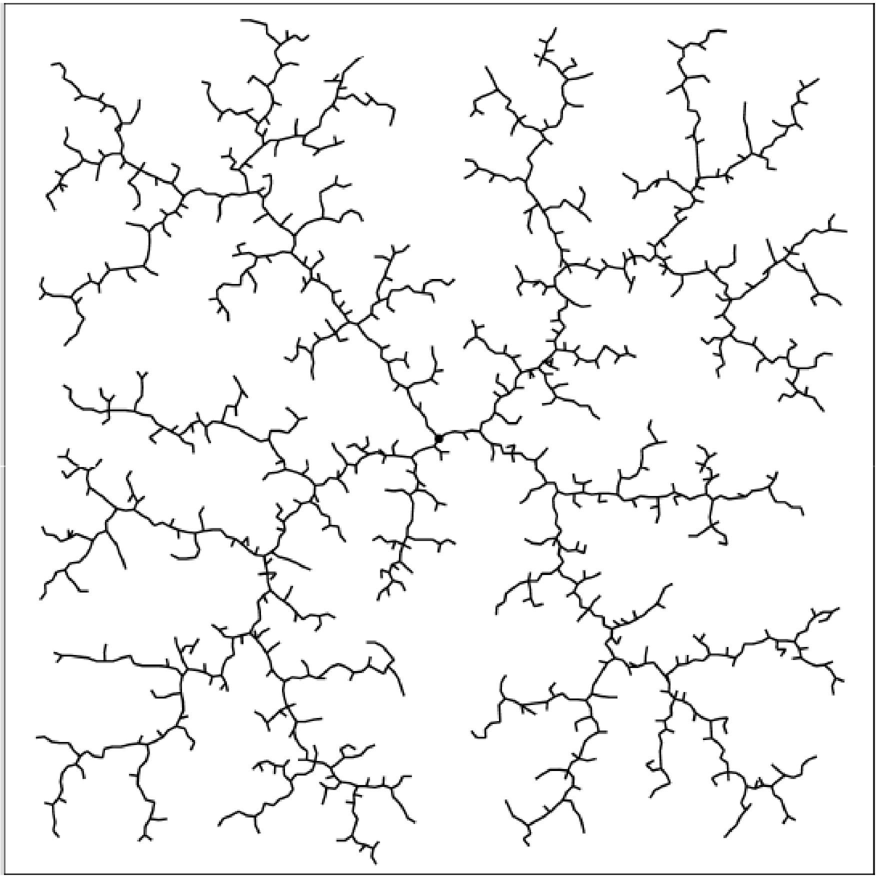
Thanks to Ken Goldberg

Two Link Path





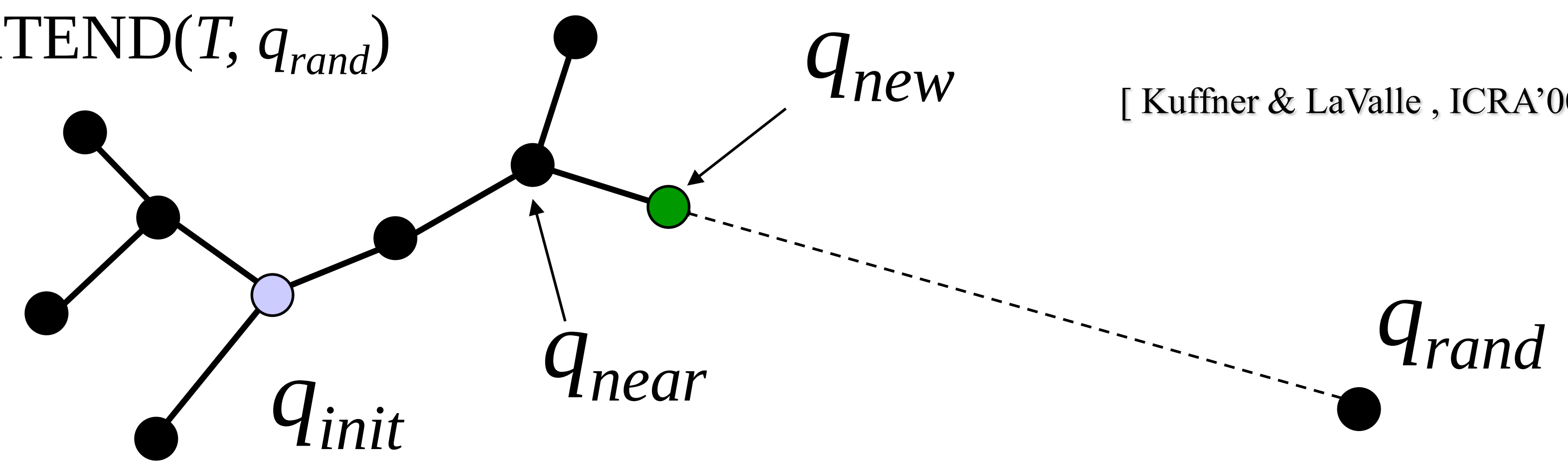
Rapidly-Exploring Random Tree



Path Planning with RRTs (Rapidly-Exploring Random Trees)

BUILD_RRT (q_{init}) {
    $T.init(q_{init})$;
    for k = 1 to K do
        q_{rand} = RANDOM_CONFIG();
        EXTEND(T, q_{rand})
}

EXTEND(T, q_{rand})



[Kuffner & LaValle , ICRA'00]

Path Planning with RRTs
(Some Details)

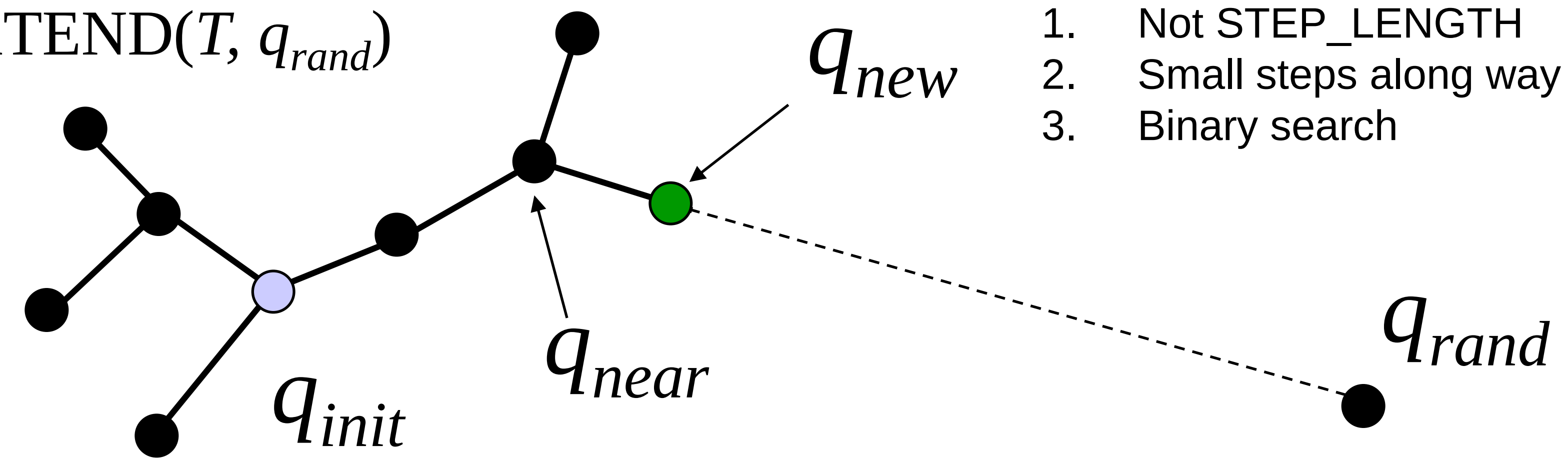
BUILD_RRT (q_{init}) {
  $T.init(q_{init})$;
  for k = 1 to K do
    q_{rand} = RANDOM_CONFIG();
    EXTEND(T, q_{rand})
}

STEP_LENGTH: How far to sample
1. Sample just at end point
2. Sample all along
3. Small Step

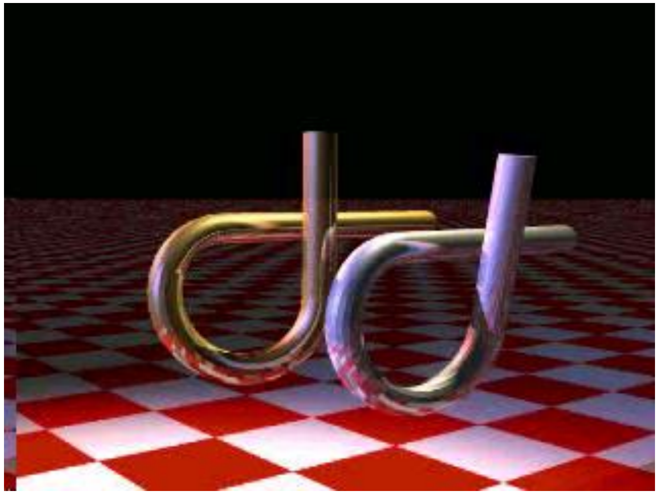
Extend returns
1. Trapped, cant make it
2. Extended, steps toward node
3. Reached, connects to node

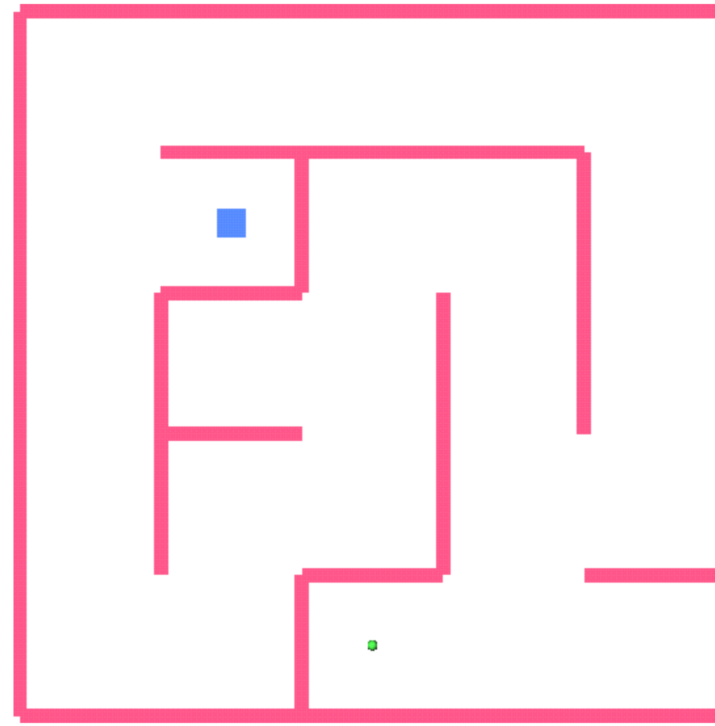
STEP_SIZE
1. Not STEP_LENGTH
2. Small steps along way
3. Binary search

EXTEND(T, q_{rand})

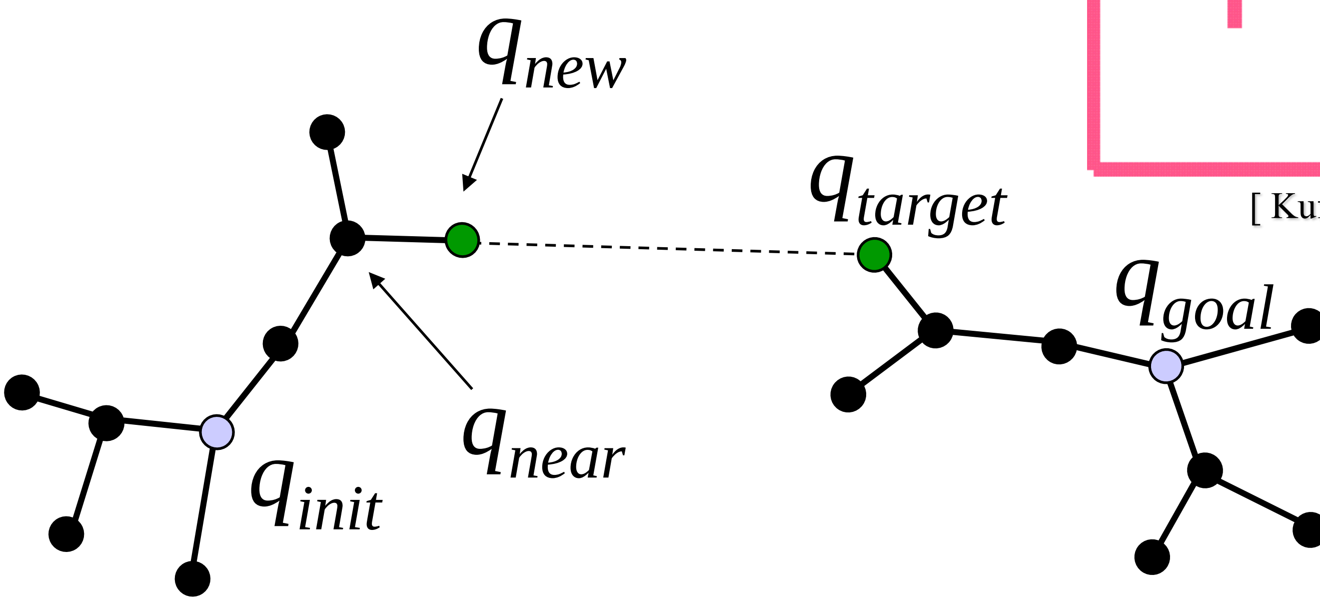


Grow two RRTs towards each other





[Kuffner, LaValle ICRA ‘00]



Map-Based Approaches: Roadmap Theory

- Properties of a roadmap:
  - Accessibility: there exists a collision-free path from the start to the road map
  - Departability: there exists a collision-free path from the roadmap to the goal.
  - Connectivity: there exists a collision-free path from the start to the goal (on the roadmap).

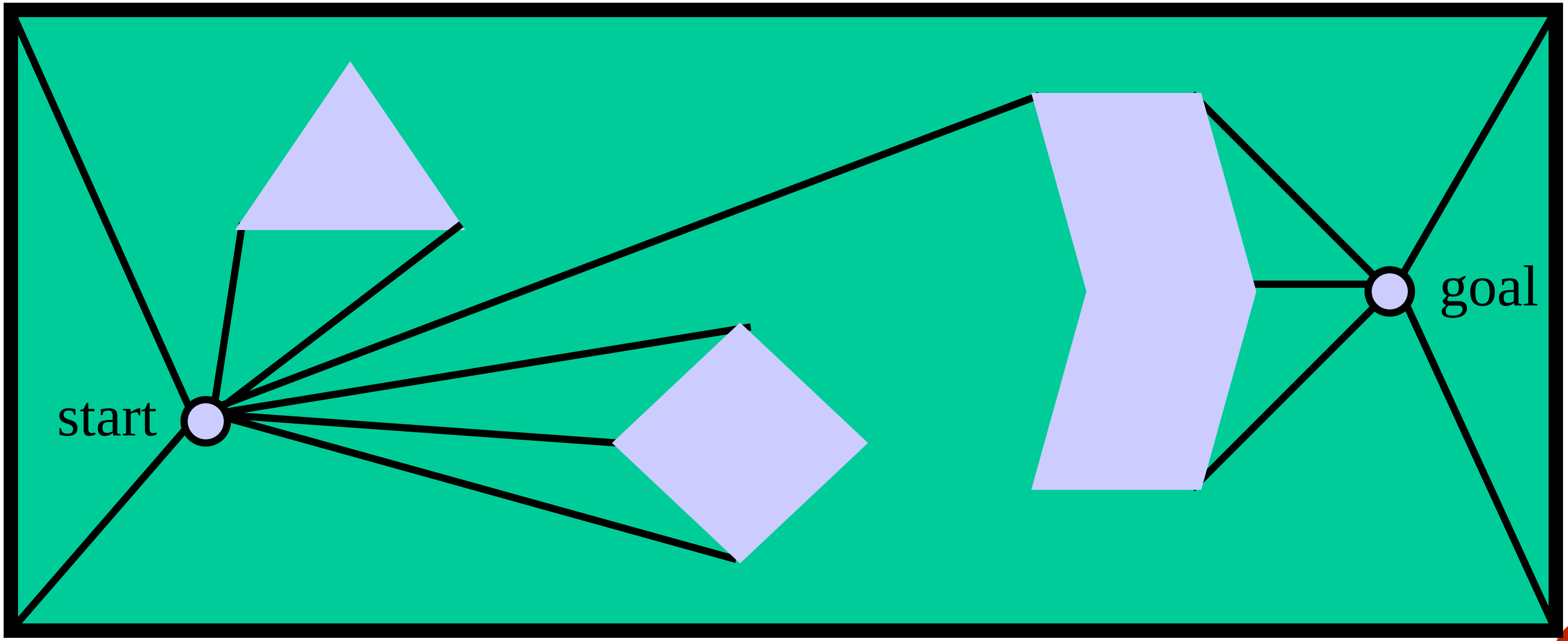
- **a roadmap exists ⇔ a path exists**
- **Examples of Roadmaps**
  - **Generalized Voronoi Graph (GVG)**
  - **Visibility Graph**

Roadmap: Visibility Graph

- Formed by connecting all "visible" vertices, the start point and the end point, to each other
- For two points to be "visible" no obstacle can exist between them
  - Paths exist on the perimeter of obstacles
- In our example, this produces the shortest path with respect to the L2 metric. However, the close proximity of paths to obstacles makes it dangerous

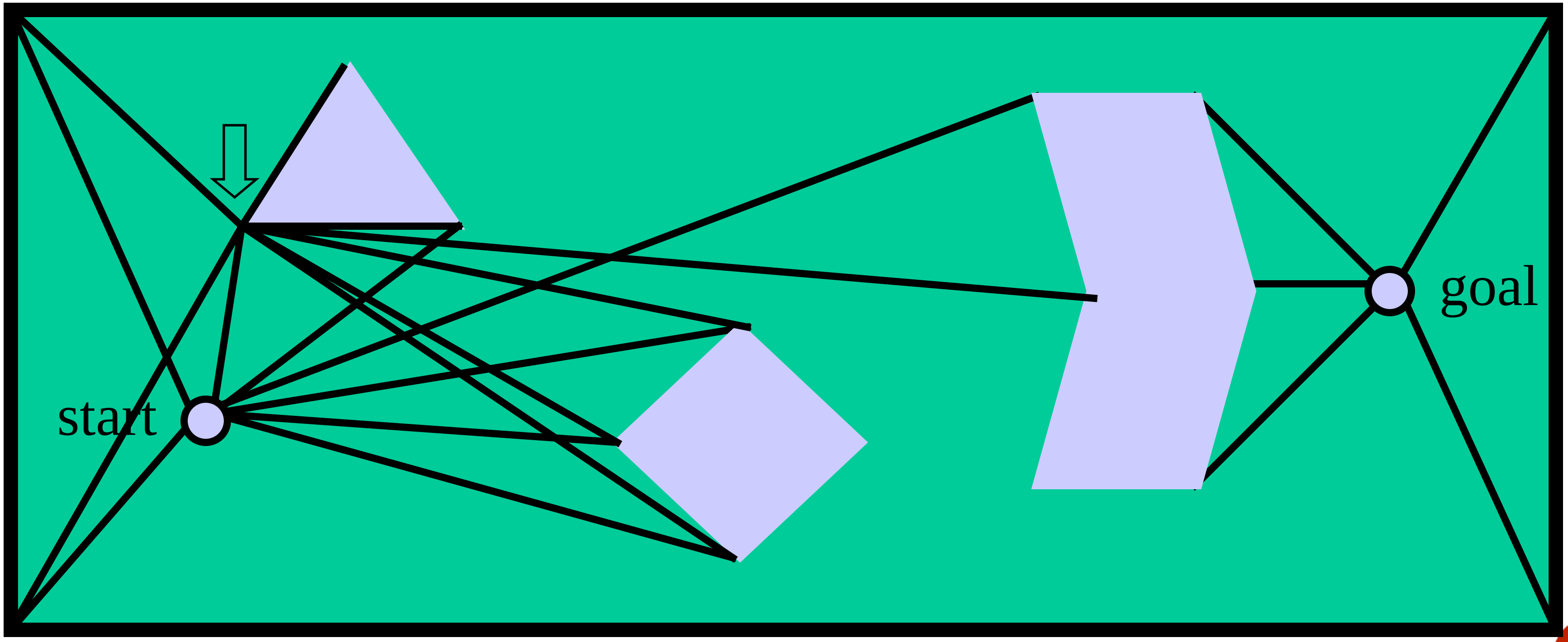
The Visibility Graph in Action (Part 1)

- First, draw lines of sight from the start and goal to all "visible" vertices and corners of the world.



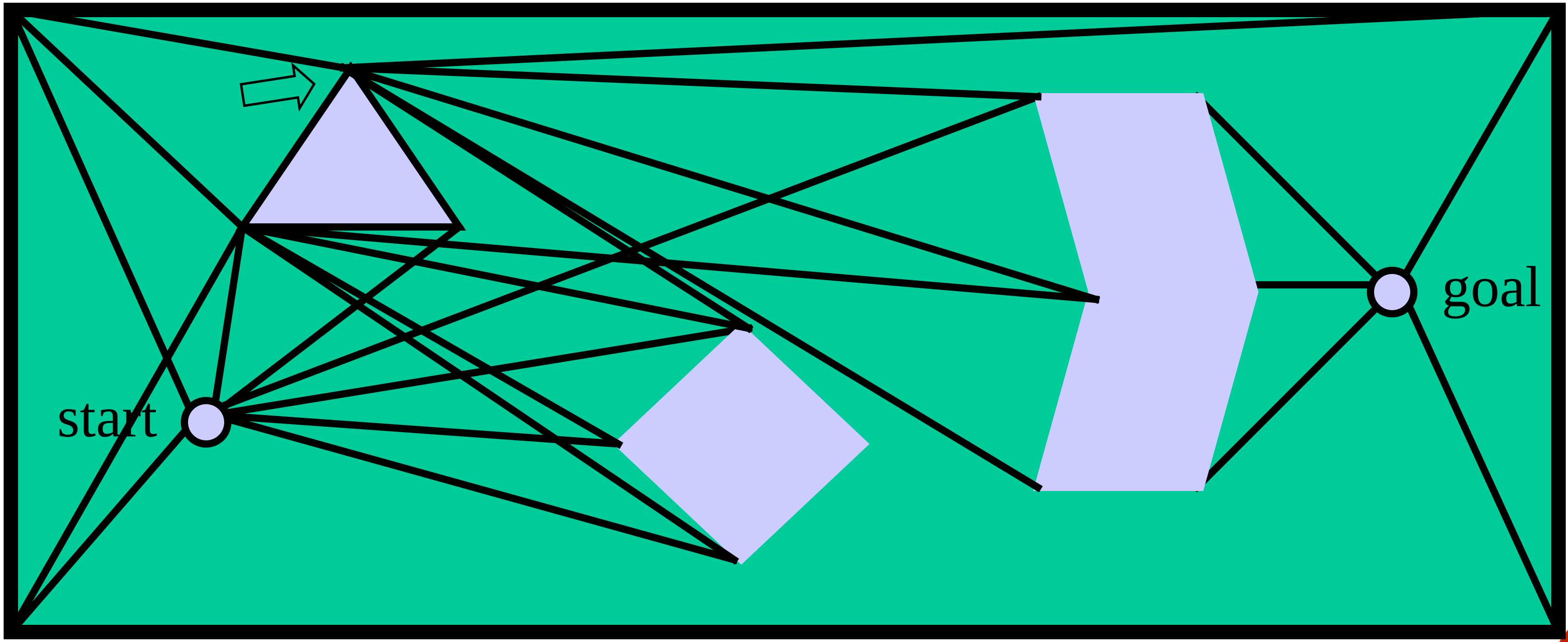
The Visibility Graph in Action (Part 2)

- Second, draw lines of sight from every vertex of every obstacle like before. Remember lines along edges are also lines of sight.



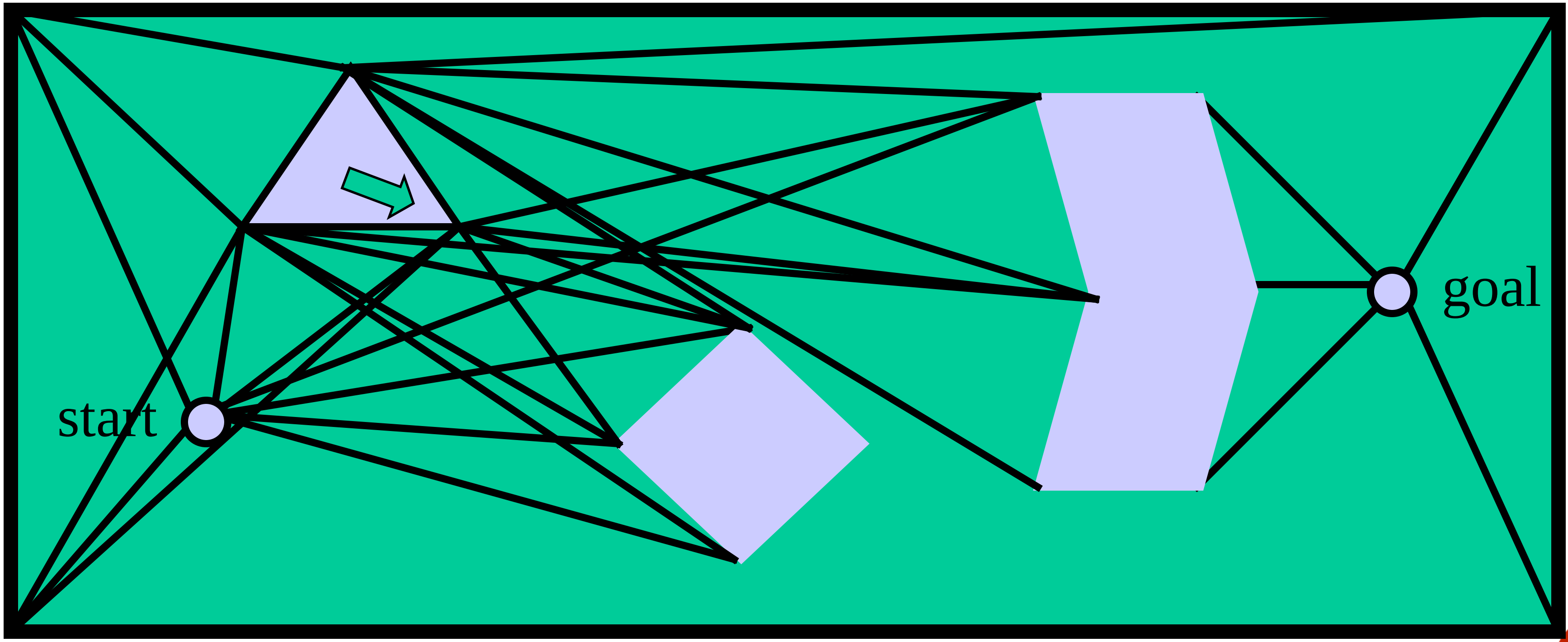
The Visibility Graph in Action (Part 3)

- Second, draw lines of sight from every vertex of every obstacle like before. Remember lines along edges are also lines of sight.



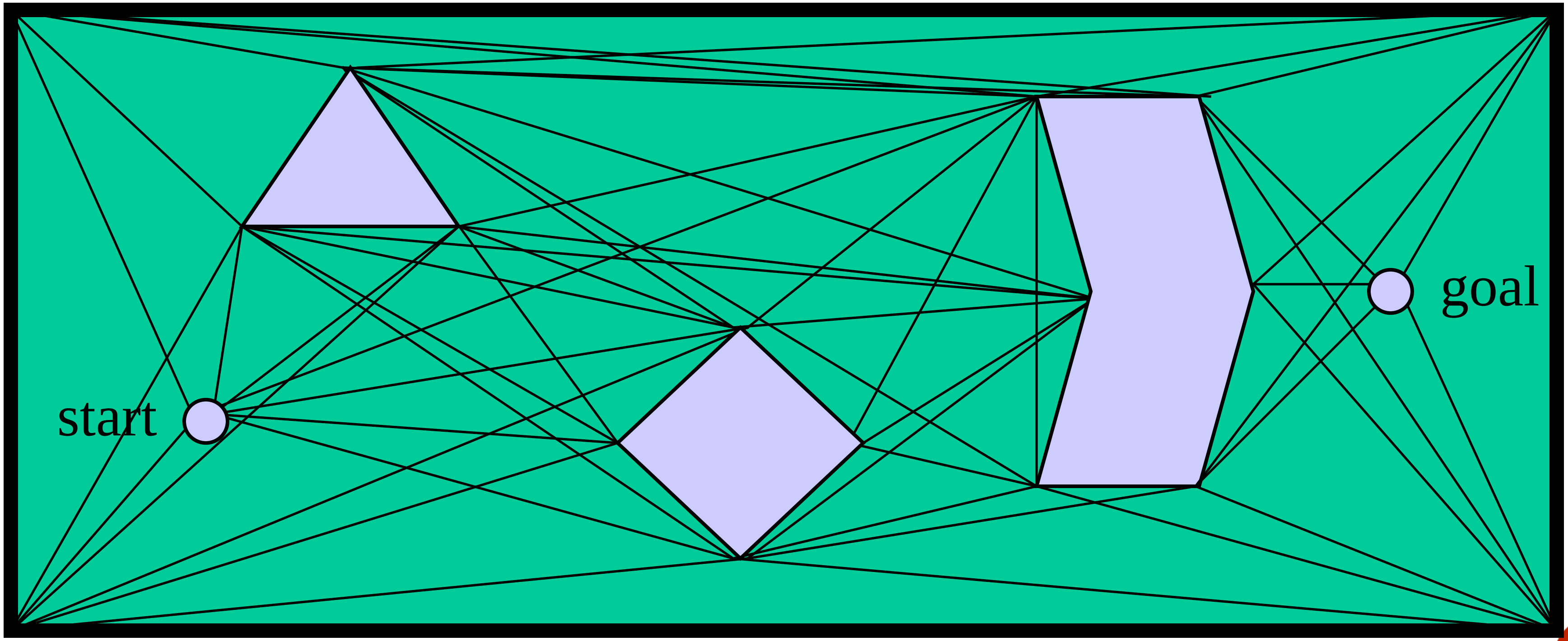
The Visibility Graph in Action (Part 4)

- Second, draw lines of sight from every vertex of every obstacle like before. Remember lines along edges are also lines of sight.



The Visibility Graph (Done)

- Repeat until you're done.



Visibility Graph Overview

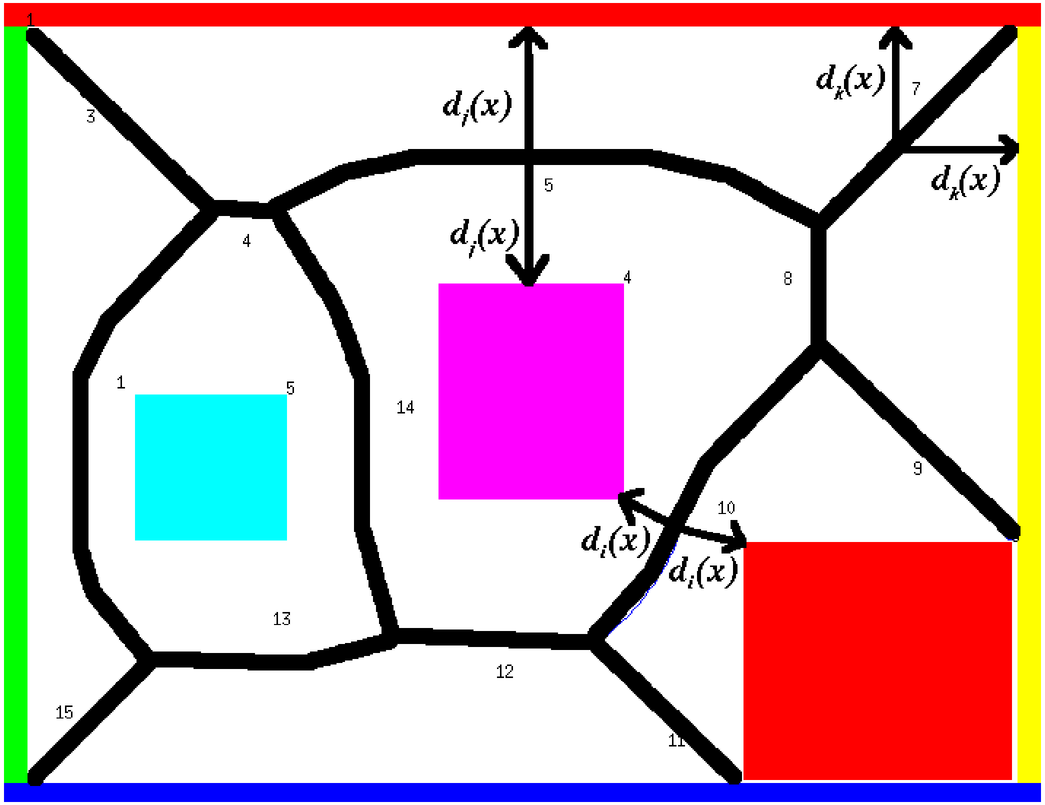
- Start with a map of the world, draw lines of sight from the start and goal to every "corner" of the world and vertex of the obstacles, not cutting through any obstacles.
- Draw lines of sight from every vertex of every obstacle like above. Lines along edges of obstacles are lines of sight too, since they don't pass through the obstacles.
- If the map was in Configuration space, each line potentially represents part of a path from the start to the goal.

Graph Search

- Who knows it?

Roadmap: GVG

- A GVG is formed by paths equidistant from the two closest objects
- *Remember "spokes", start and goal*



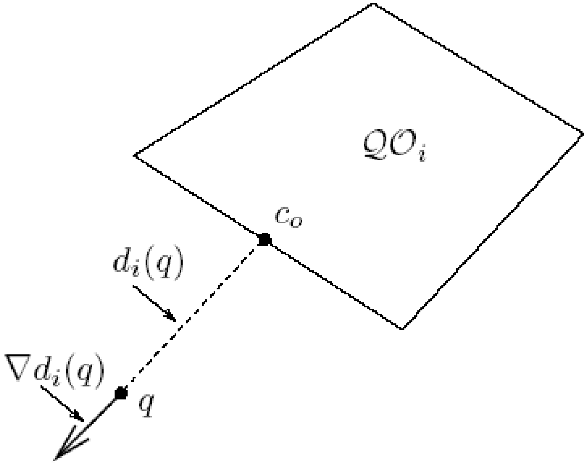
- This generates a very safe roadmap which avoids obstacles as much as possible

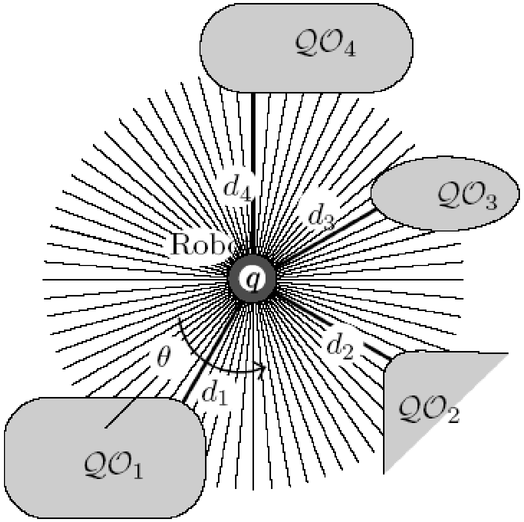
Distance to Obstacle(s)

$$d_i(q) = \min_{c \in \mathcal{QO}_i} d(q, c)$$

$$\nabla d_i(q) = \frac{q - c}{d(q, c)}$$

$$D(q) = \min d_i(q)$$

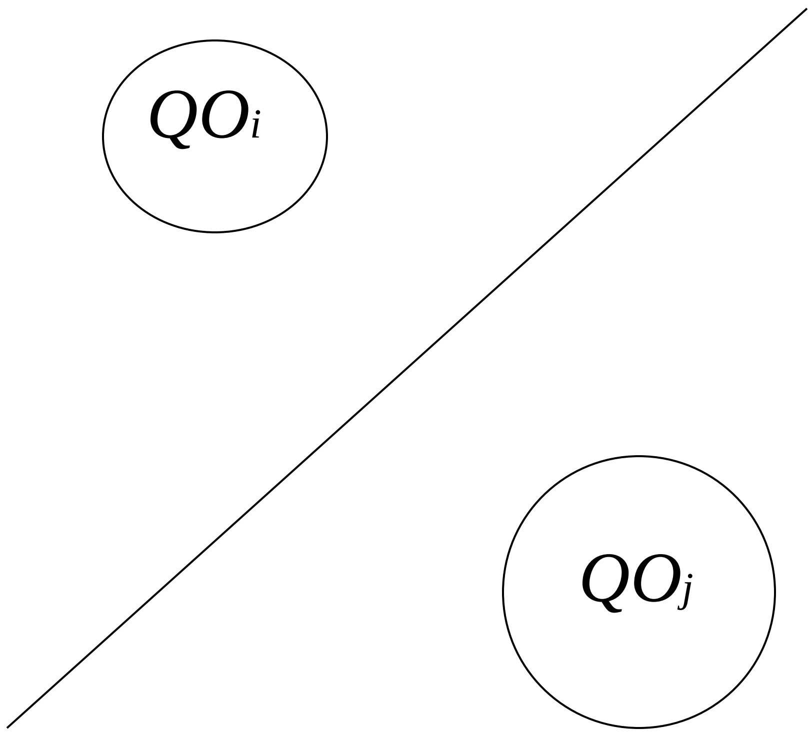




Two-Equidistant

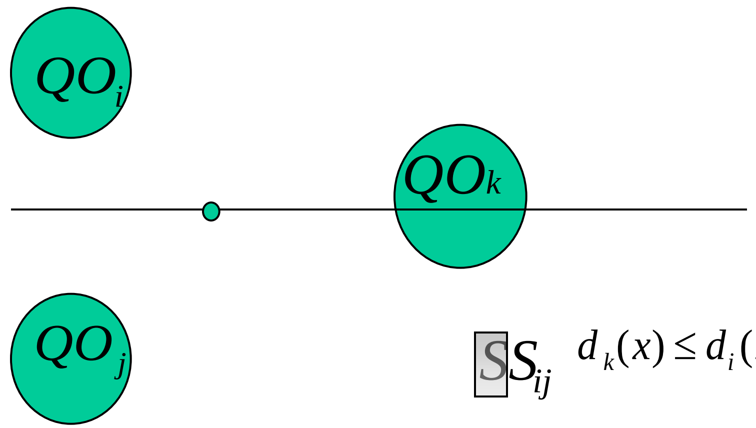
- *Two-equidistant surface*

$$S_{ij} = \{x \in Q_{\text{free}} : d_i(x) - d_j(x) = 0\}$$



More Rigorous Definition

Going through obstacles



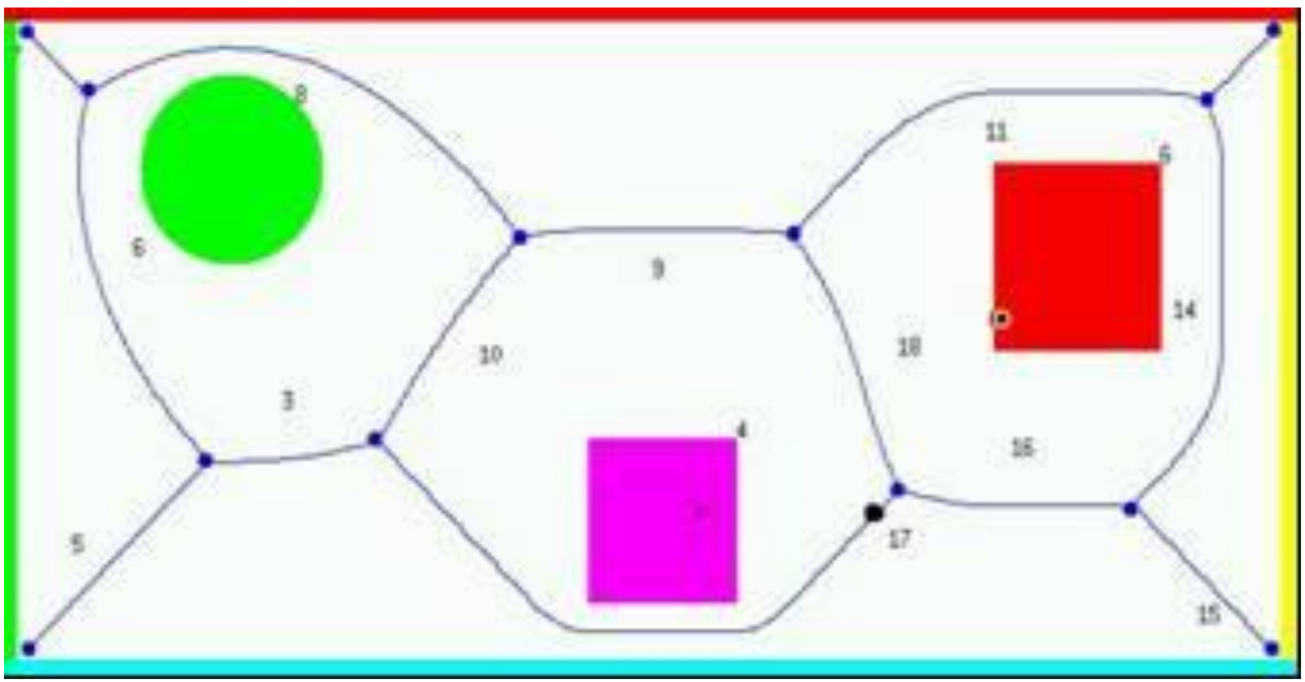
$SS_{ij} \quad d_k(x) \le d_i(x) = d_j(x)$

Two-equidistant face

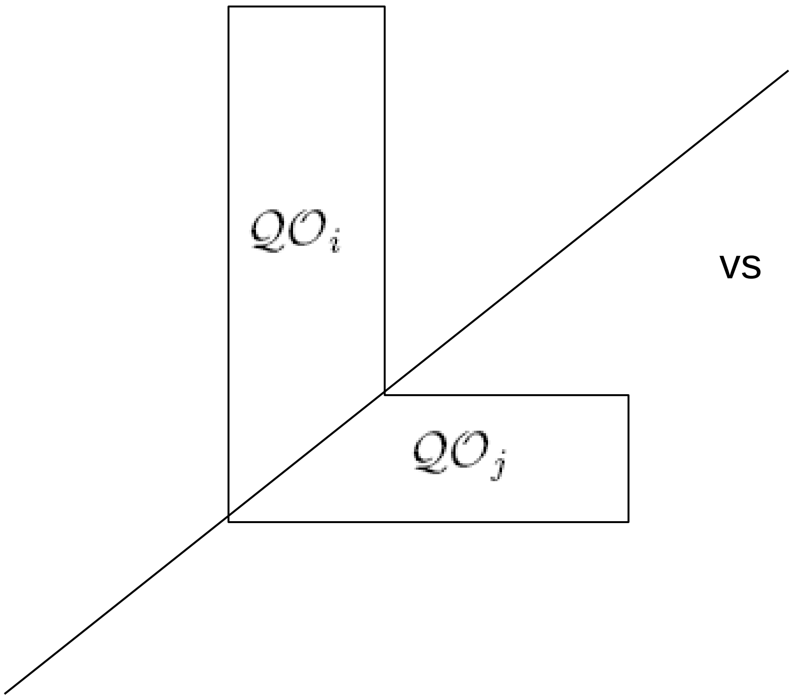
$$F_{ij} = \{x \in SS_{ij} : d_i(x) = d_j(x) \le d_h(x), \forall h \ne i, j\}$$

General Voronoi Diagram

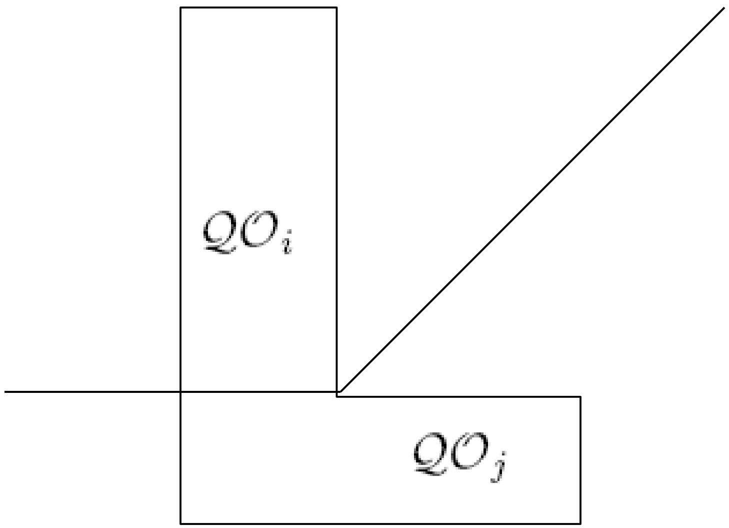
$$\mathrm{GVD} = \bigcup_{i=1}^{n-1} \bigcup_{j=i+1}^{n} F_{ij}$$



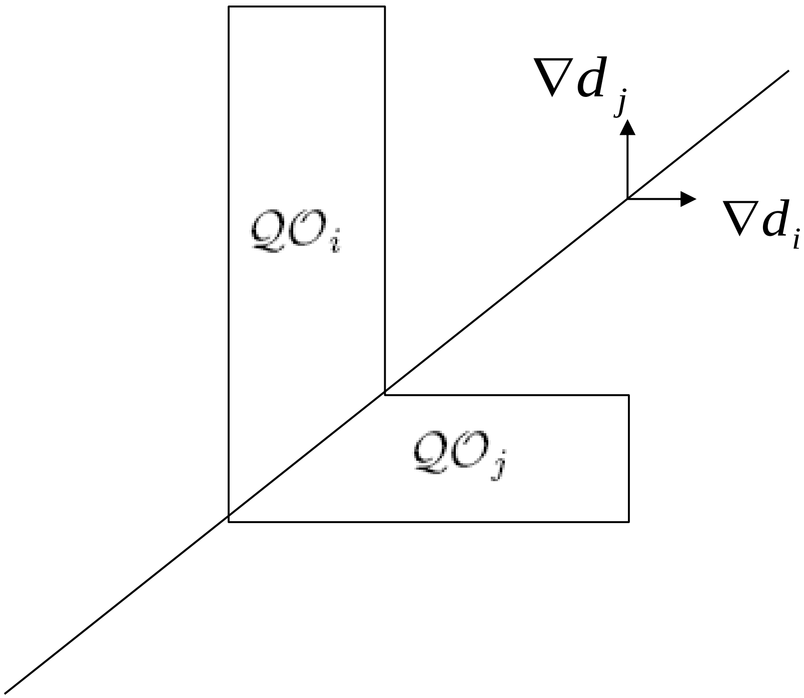
What about concave obstacles?



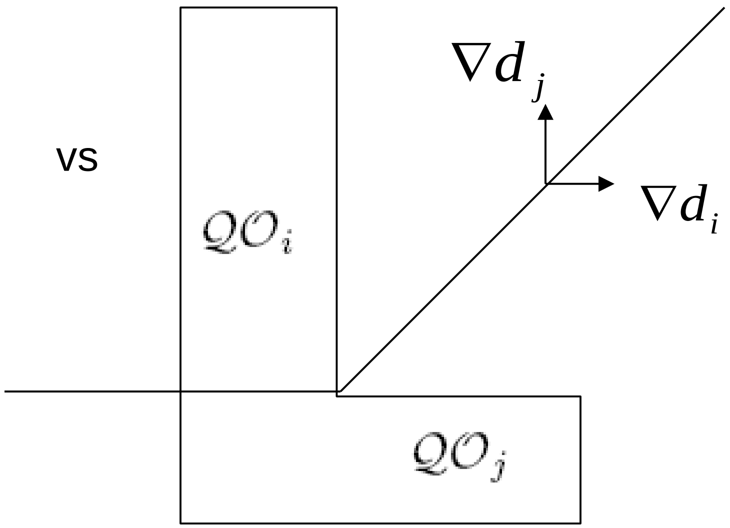
vs



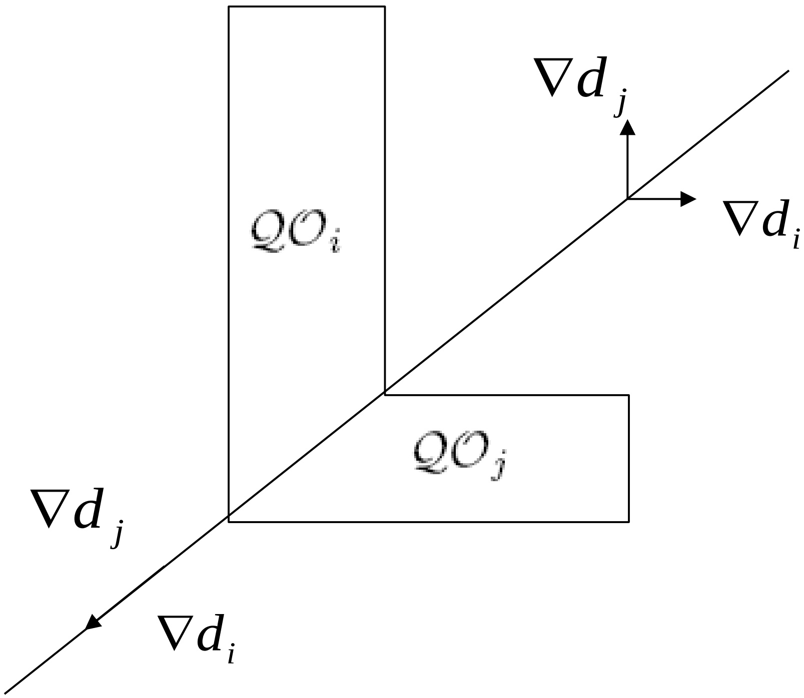
What about concave obstacles?



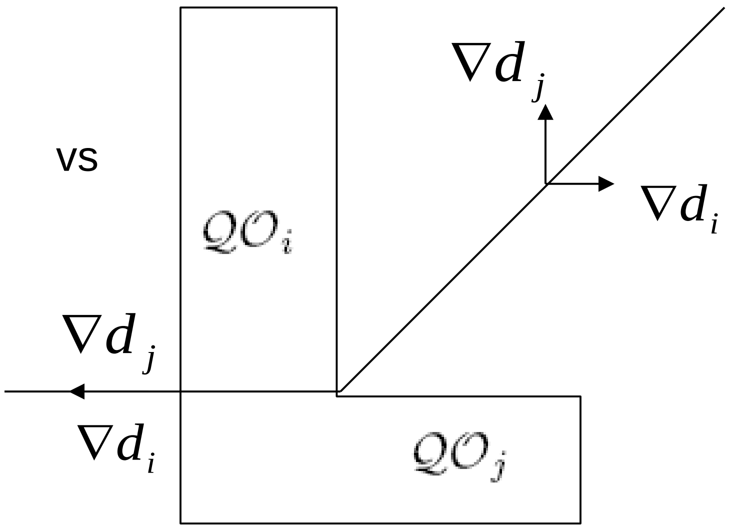
vs



What about concave obstacles?



vs



Two-Equidistant

- *Two-equidistant surface*

$$S_{ij} = \{x \in Q_{\text{free}} : d_i(x) - d_j(x) = 0\}$$

Two-equidistant surjective surface

$$SS_{ij} = \{x \in S_{ij} : \nabla d_i(x) \neq \nabla d_j(x)\}$$

Two-equidistant Face

$$F_{ij} = \{x \in SS_{ij} : d_i(x) \leq d_h(x), \forall h \neq i\}$$

$$\text{GVD} = \bigcup_{i=1}^{n-1} \bigcup_{j=i+1}^{n} F_{ij}$$