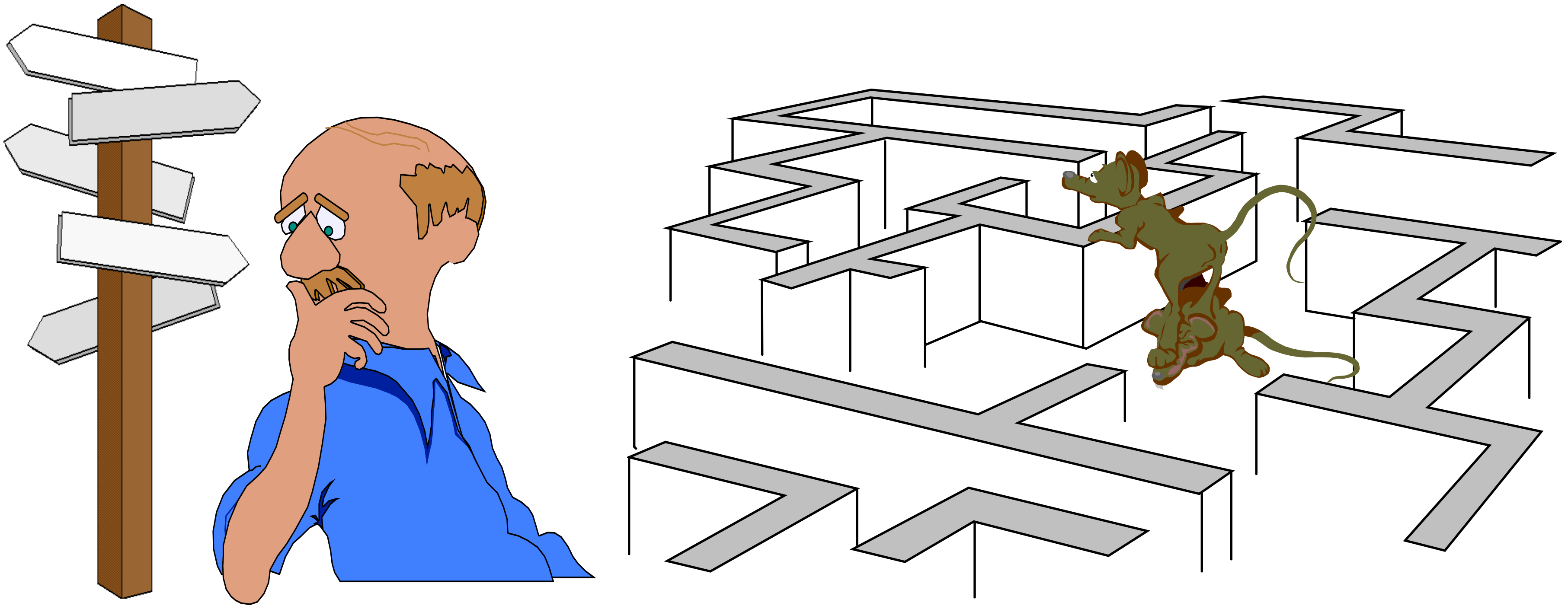
Motion Planning

Howie CHoset

What is Motion Planning?

What is Motion Planning?

- Determining where to go



Overview

- The Basics
  - Motion Planning Statement
  - The World and Robot
  - Configuration Space
  - Metrics

Algorithms

- Start-Goal Methods
- Map-Based Approaches
- Cellular Decompositions

The World consists of...

- Obstacles
  - Already occupied spaces of the world
  - In other words, robots can’t go there
- Free Space
  - Unoccupied space within the world
  - Robots “might” be able to go here
  - To determine where a robot can go, we need to discuss what a *Configuration Space* is

Motion Planning Statement

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

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

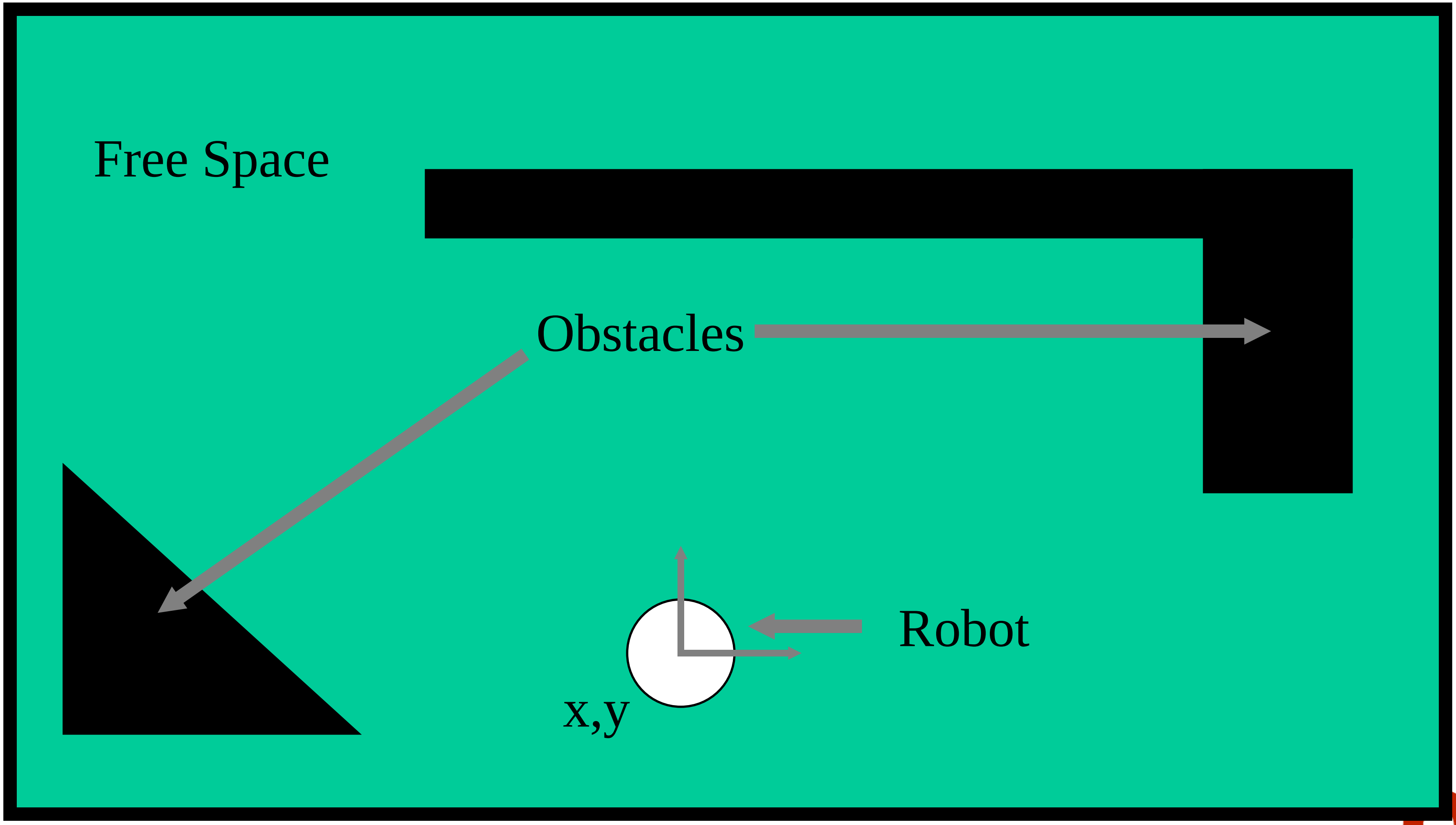
Then the robot's free space, **FS,** is defined as:

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

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

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

Example of a World (and Robot)



What is a good path?

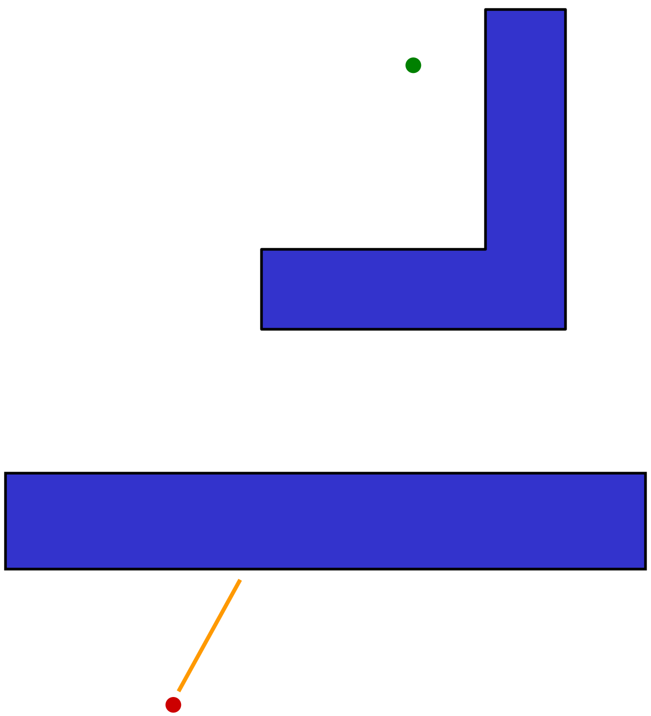
Basics: Metrics

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

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

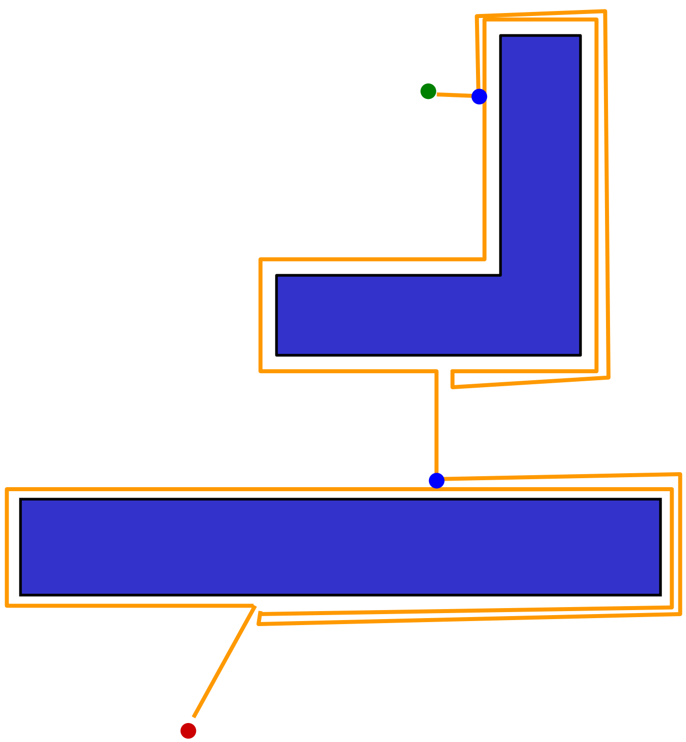
Vladimir Lumelsky & Alexander Stepanov: Algorithmica 1987

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

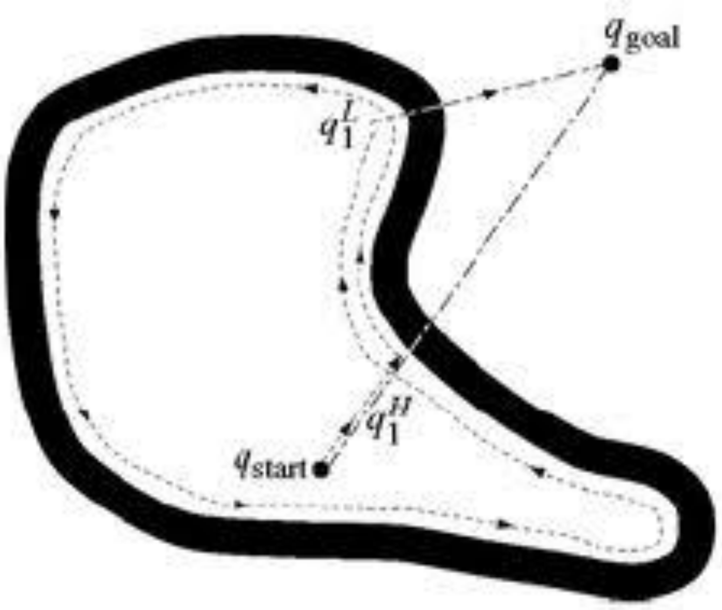
Vladimir Lumelsky & Alexander Stepanov: Algorithmica 1987

Bug 1

But some computing power!

- **known direction to goal**
- **otherwise local sensing**

walls/obstacles & **encoders**



"Bug 1" algorithm

1) head toward goal

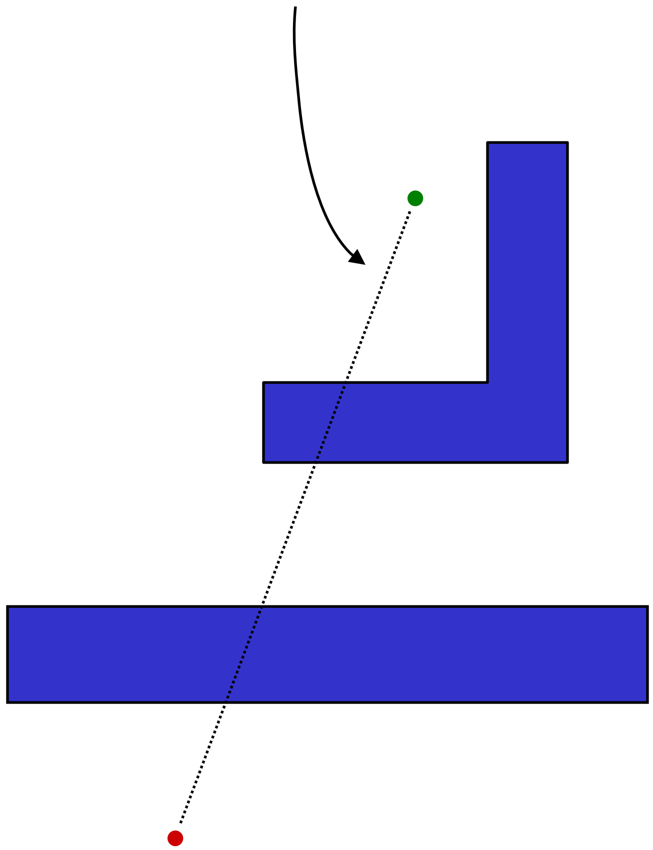
2) if an obstacle is encountered, circumnavigate it *and* remember how close you get to the goal

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

Vladimir Lumelsky & Alexander Stepanov: Algorithmica 1987

Bug2

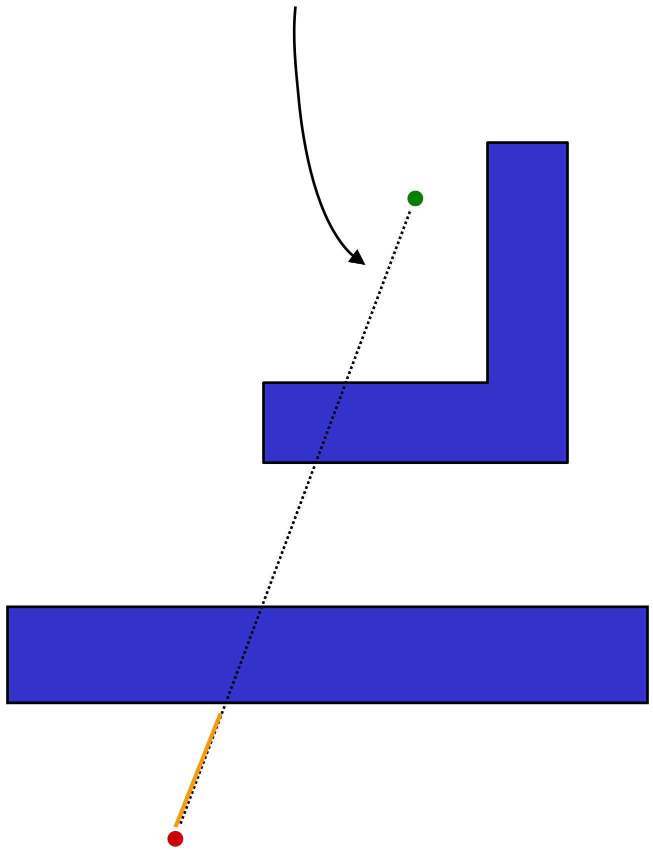
Call the line from the starting point to the goal the ***m-line***



"Bug 2" Algorithm

A better bug?

Call the line from the starting point to the goal the ***m-line***

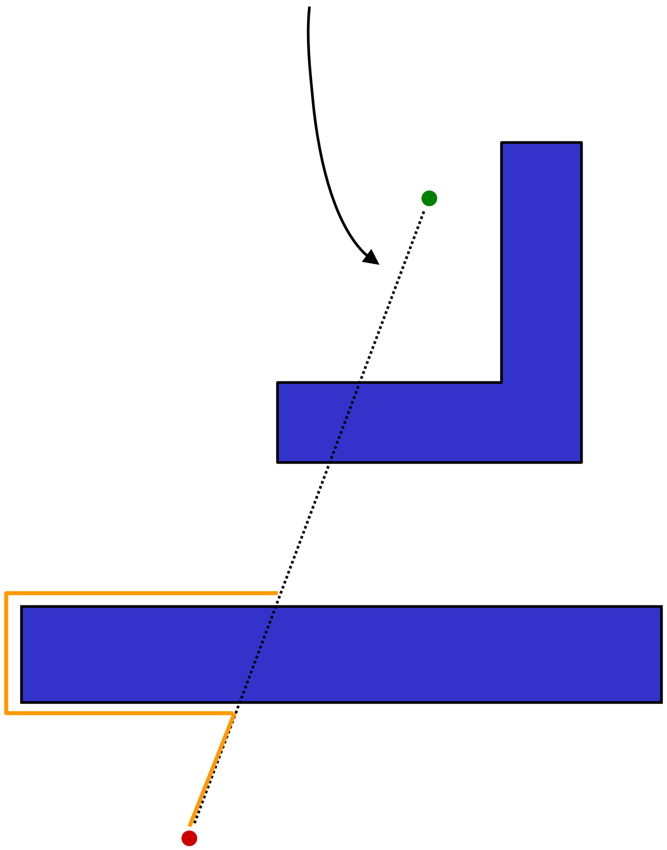


"Bug 2" Algorithm

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

A better bug?

Call the line from the starting point to the goal the ***m-line***

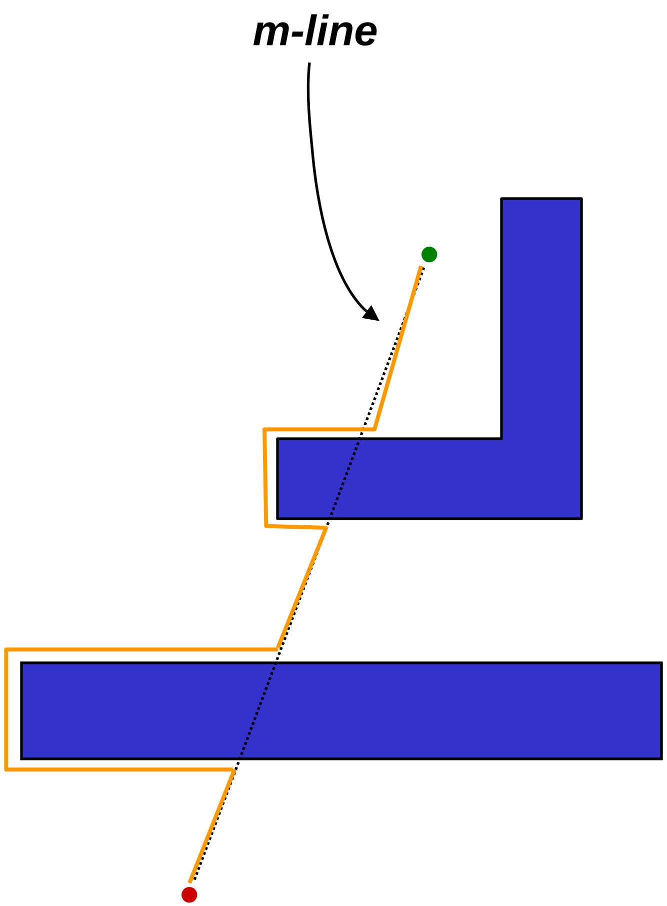


"Bug 2" Algorithm

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.

A better bug?



"Bug 2" Algorithm

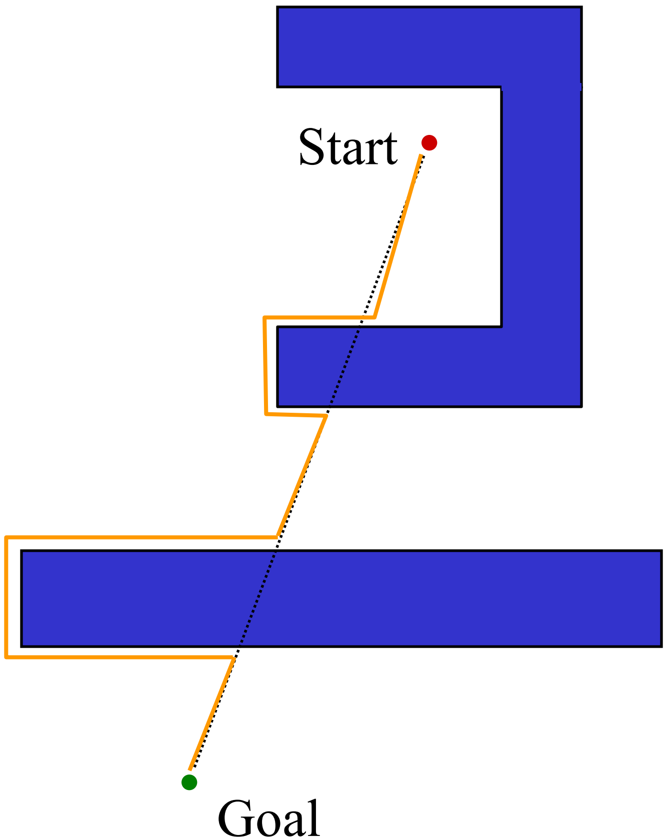
1) head toward goal on the *m-line*

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

3) Leave the obstacle and continue toward the goal

OK ?

A better bug?



"Bug 2" Algorithm

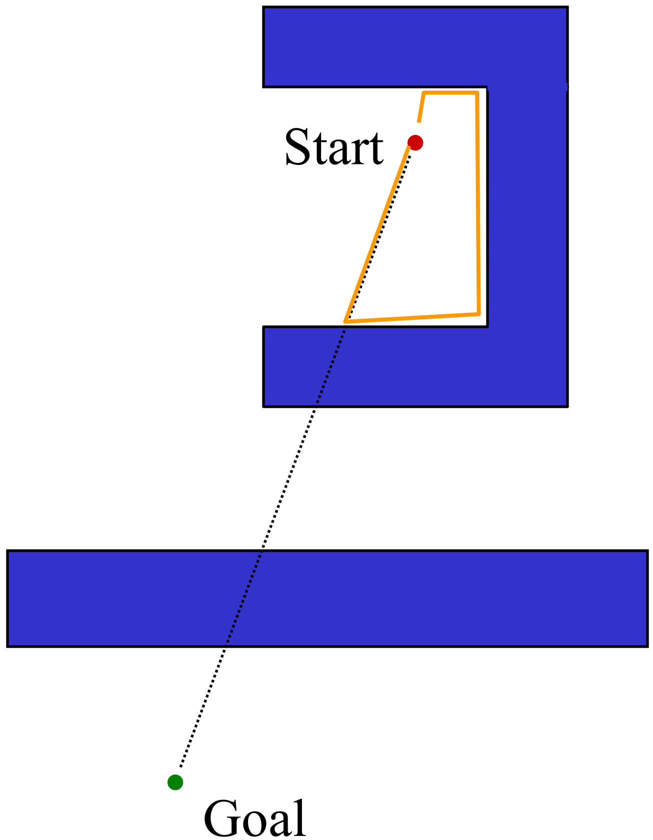
1) head toward goal on the *m-line*

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

3) Leave the obstacle and continue toward the goal

Better or worse than Bug1?

A better bug?



"Bug 2" Algorithm

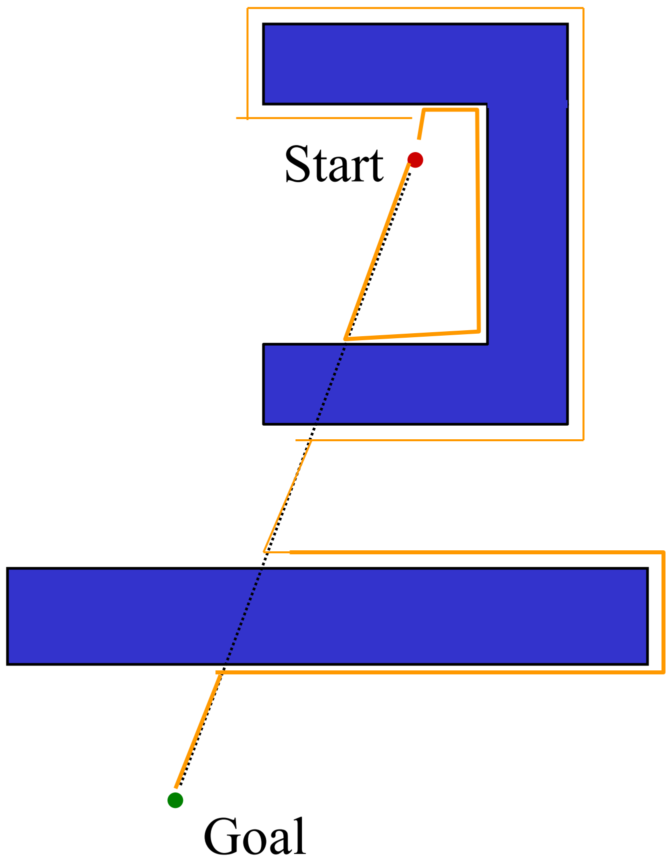
1) head toward goal on the *m-line*

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

3) Leave the obstacle and continue toward the goal

NO! How do we fix this?

A better bug?



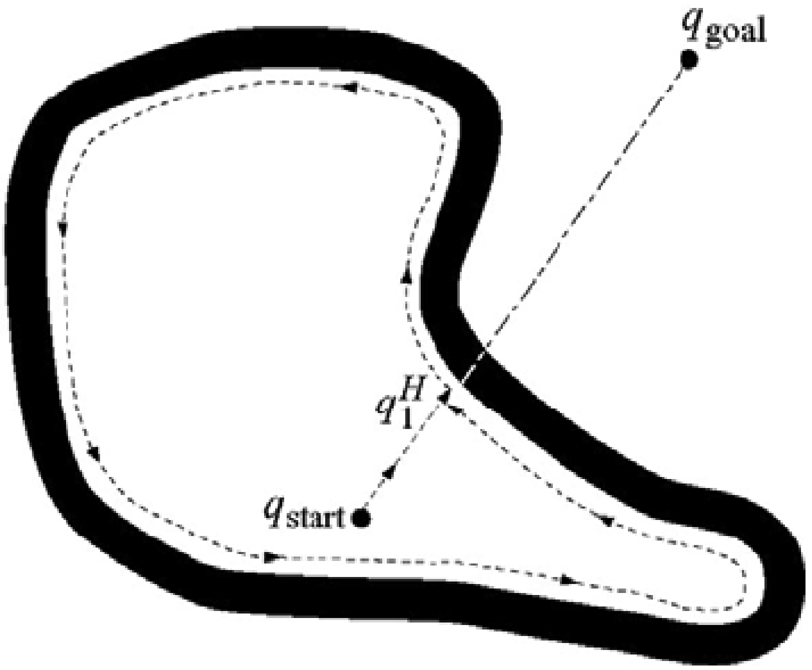
"Bug 2" Algorithm

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

A better bug?



"Bug 2" Algorithm

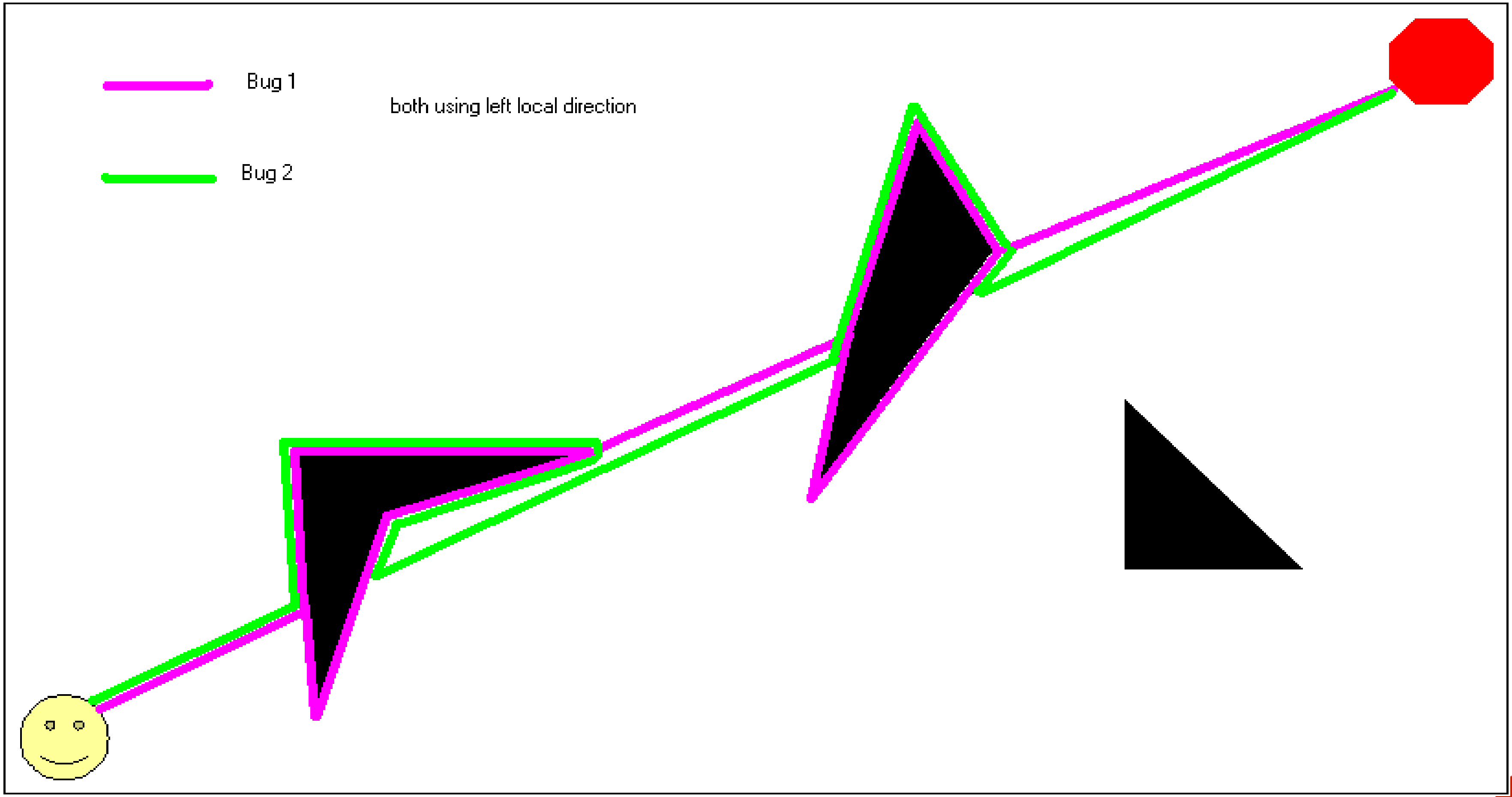
1) head toward goal on the *m-line*

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?

Start-Goal Algorithm: Lumelsky Bug Algorithms



Lumelsky Bug Algorithms

- Unknown obstacles, known start and goal.
- Simple "bump" sensors, encoders.
- Choose arbitrary direction to turn (left/right) to make all turns, called "local direction"
- Motion is like an ant walking around:
  - In Bug 1 the robot goes all the way around each obstacle encountered, recording the point nearest the goal, then goes around again to leave the obstacle from that point
  - In Bug 2 the robot goes around each obstacle encountered until it can continue on its previous path toward the goal

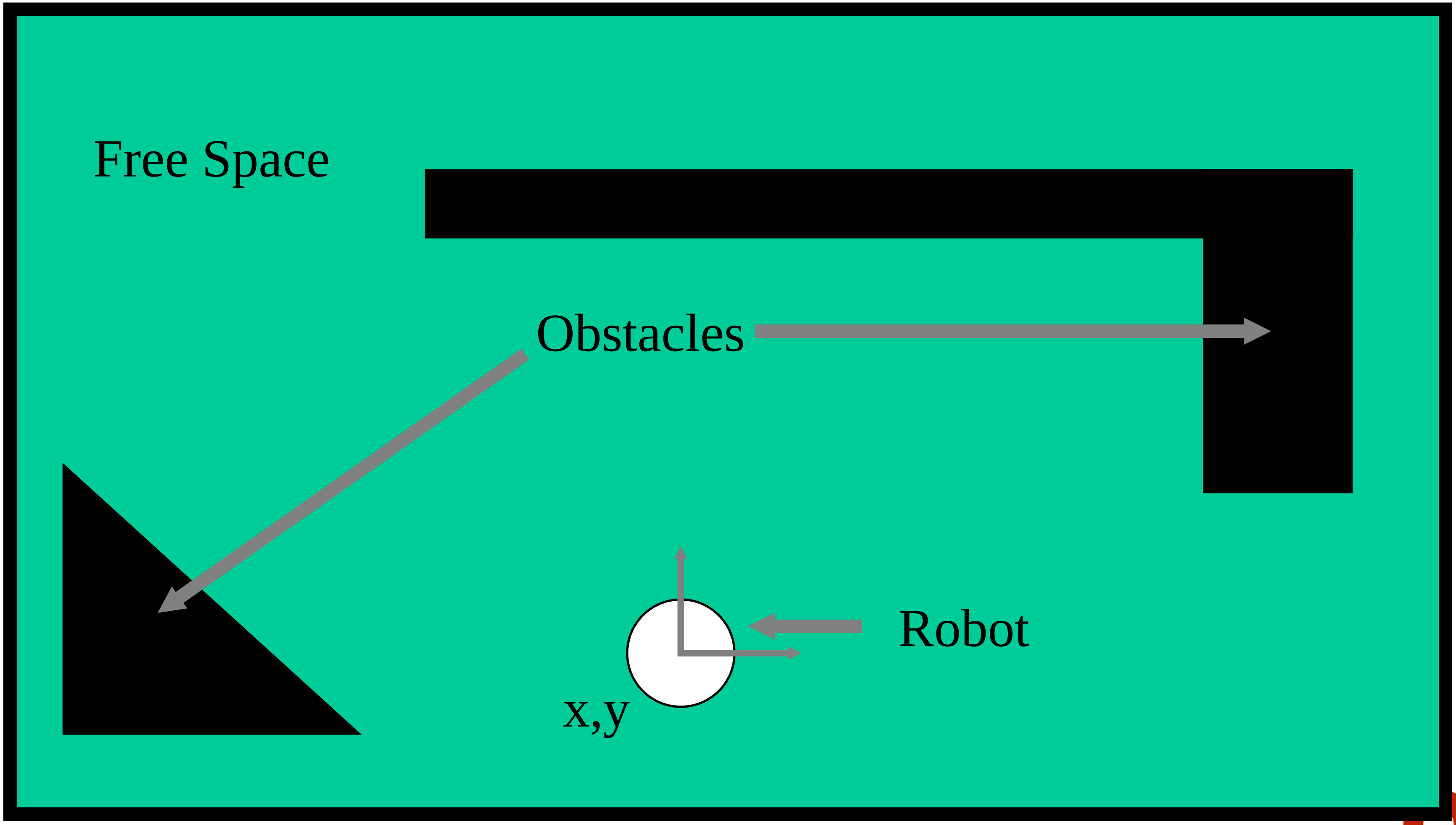
Assumptions?

Assumptions

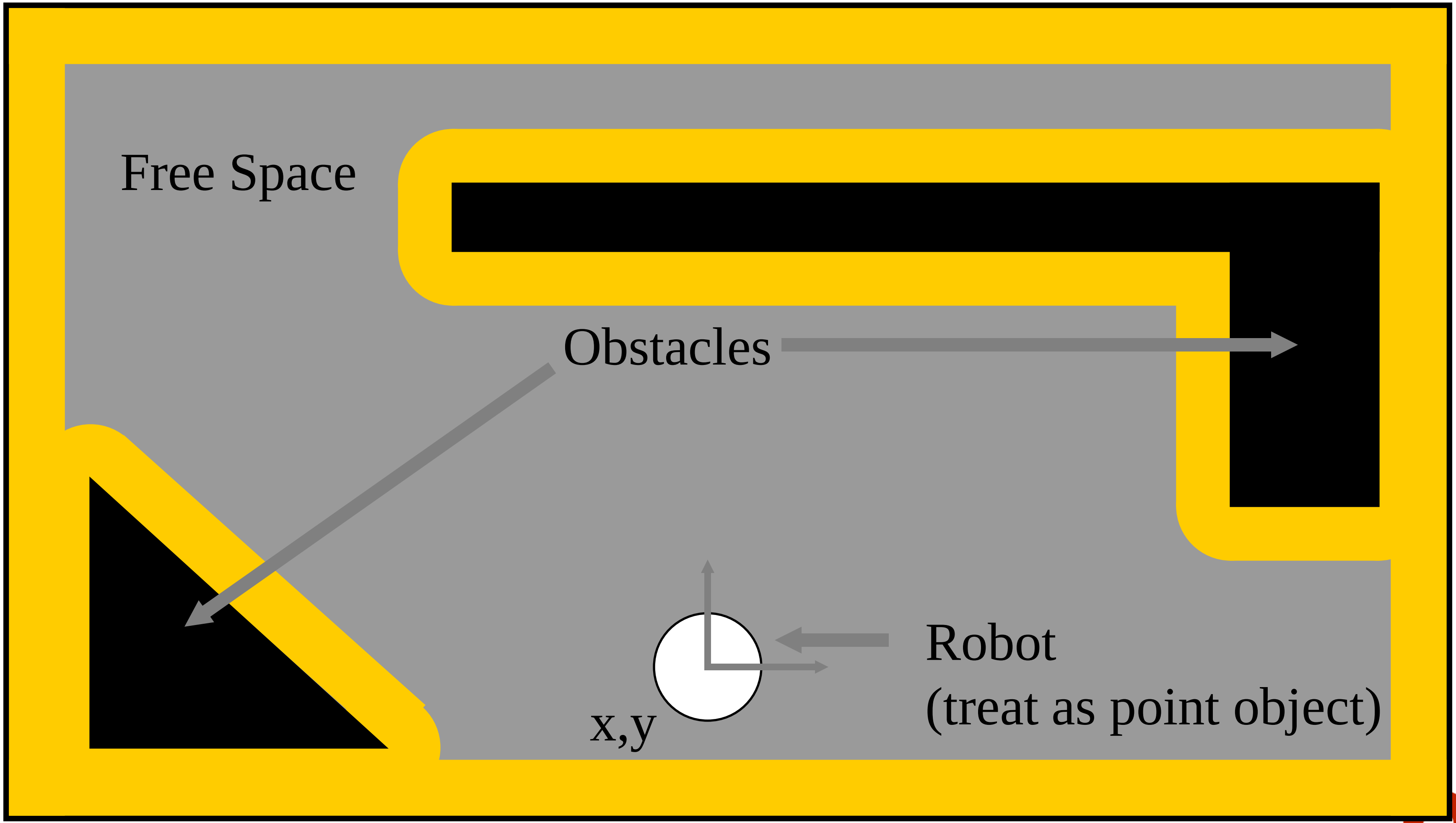
- Size of robot
- Perfect sensing
- Perfect control
- Localization (heading)

What else?

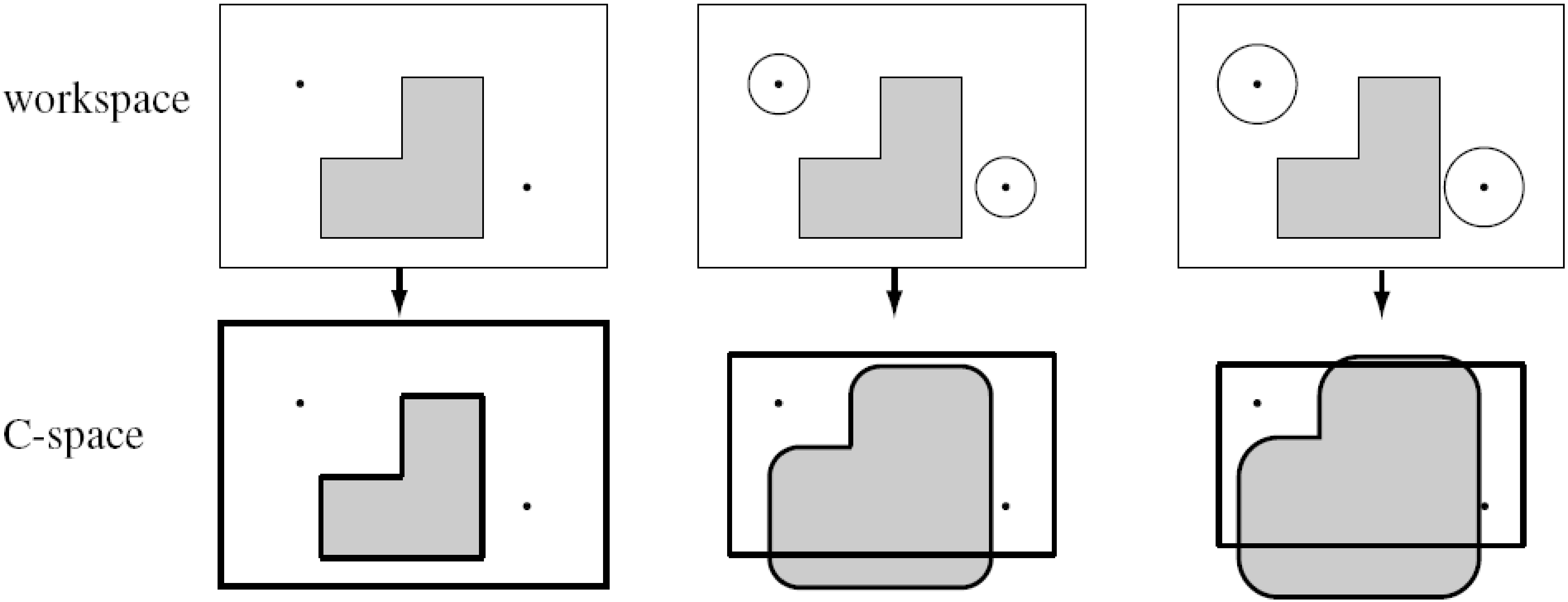
Example of a World (and Robot)



Configuration Space: Accommodate Robot Size

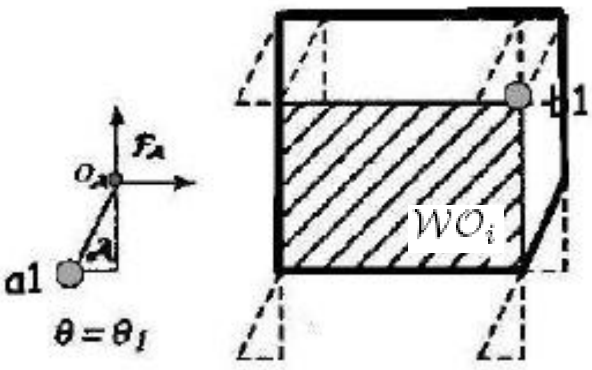


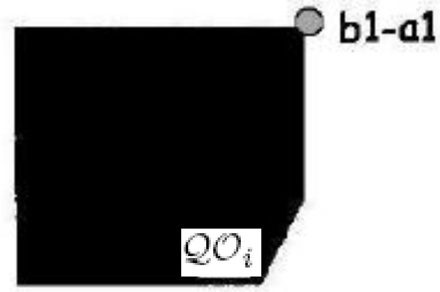
Trace Boundary of Workspace



Pick a reference point…

Translate-only, non-circularly

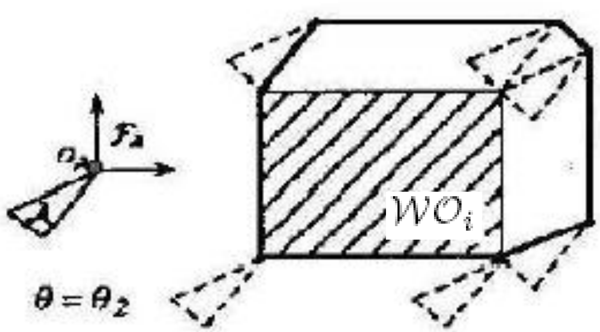


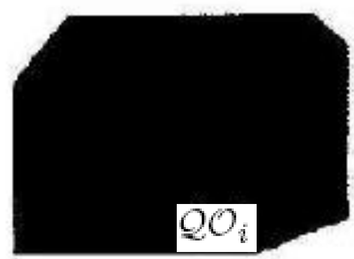


$$QO_i = \{q \in Q \mid R(q) \bigcap WO_i \neq \emptyset\}.$$

Pick a reference point…

Translate-only, non-circularly symmetric





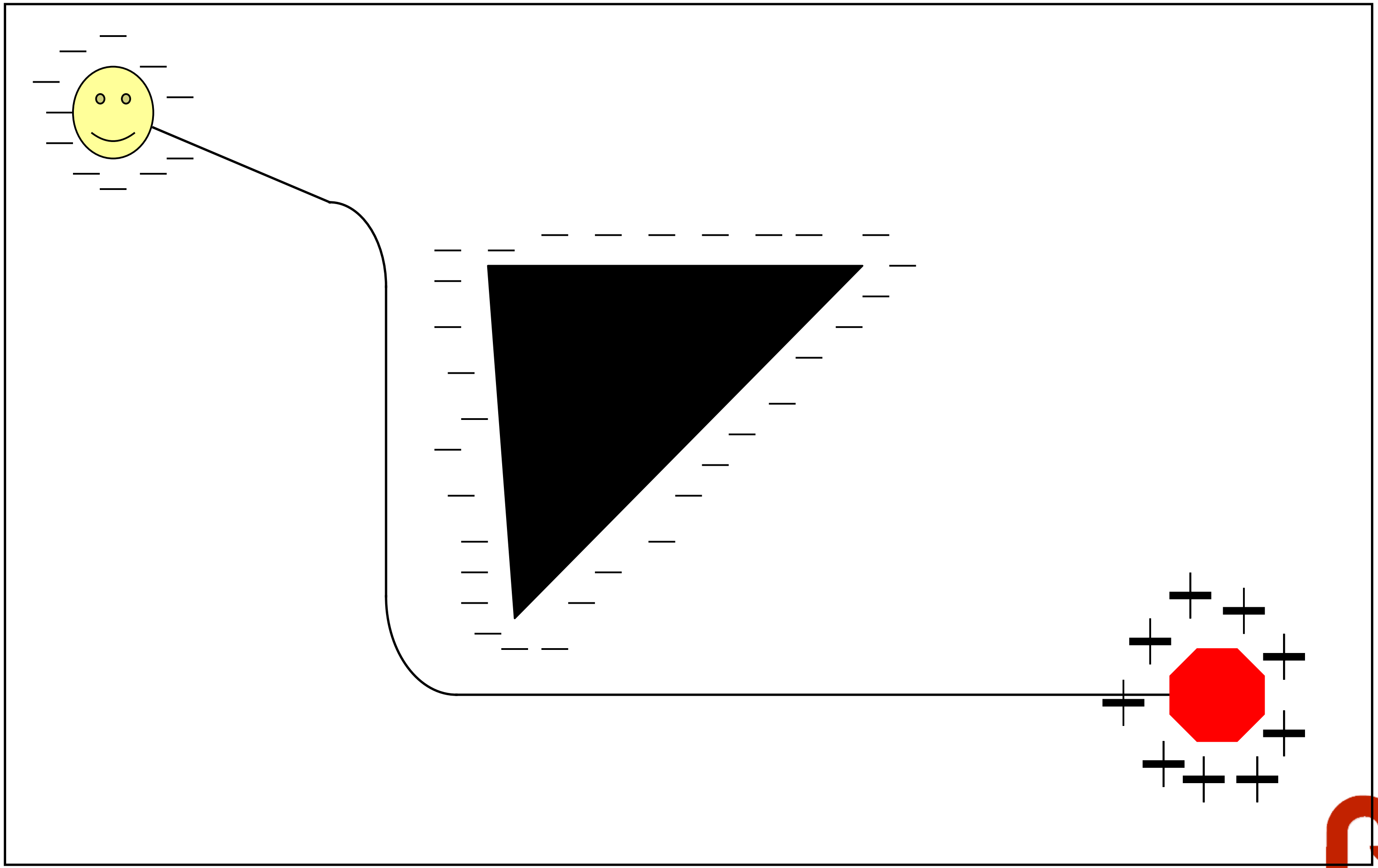
$$\mathcal{QO}_i = \{q \in \mathcal{Q} \mid R(q) \bigcap \mathcal{WO}_i \neq \emptyset\}.$$

Pick a reference point…

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*)

Start-Goal Algorithm: Potential Functions



Attractive/Repulsive Potential Field

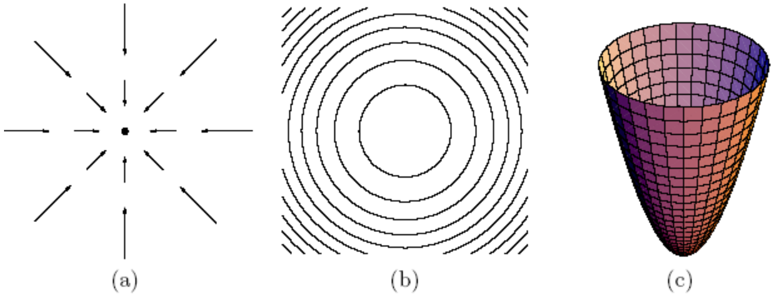
$$U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q)$$

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

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

Artificial Potential Field Methods: Attractive Potential

<u>Quadratic Potential</u>



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

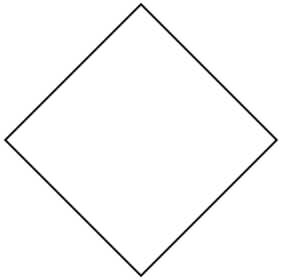
$$\begin{aligned} F_{\text{att}}(q) = \nabla U_{\text{att}}(q) &= \nabla\left(\frac{1}{2}\zeta d^2(q, q_{\text{goal}})\right), \\ &= \frac{1}{2}\zeta \nabla d^2(q, q_{\text{goal}}), \\ &= \zeta(q - q_{\text{goal}}), \end{aligned}$$

THE ROBOTICS INSTITUTE

Distance

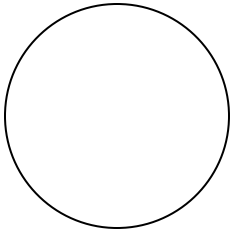
$$d : R^2 \times R^2 \rightarrow R$$

L1 Metric



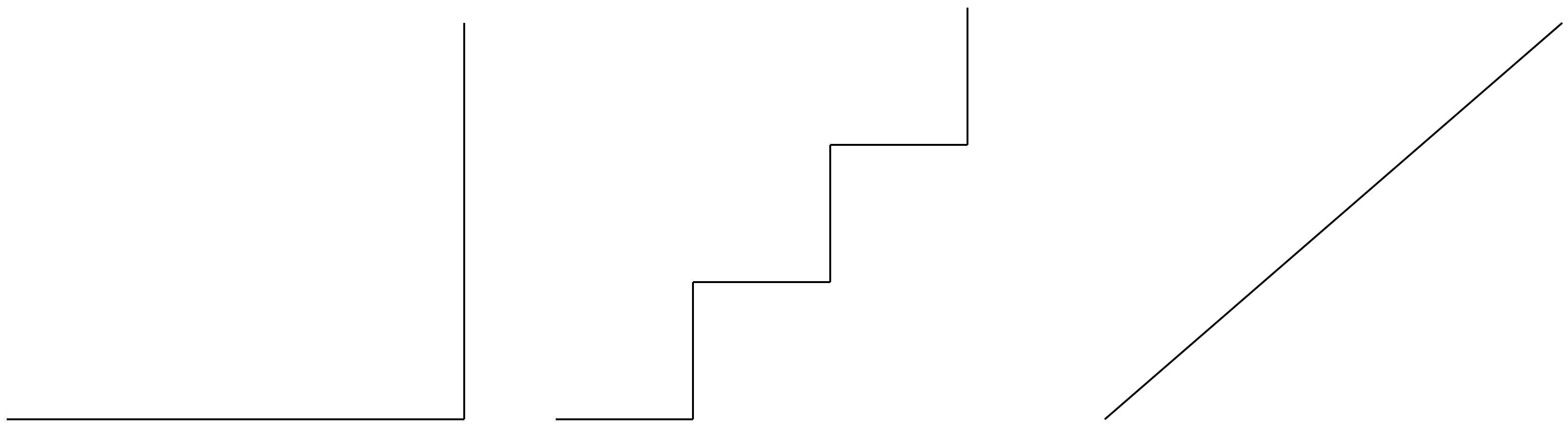
$$d(a,b) = |a_x - b_x| + |a_y - b_y|$$

L2 Metric

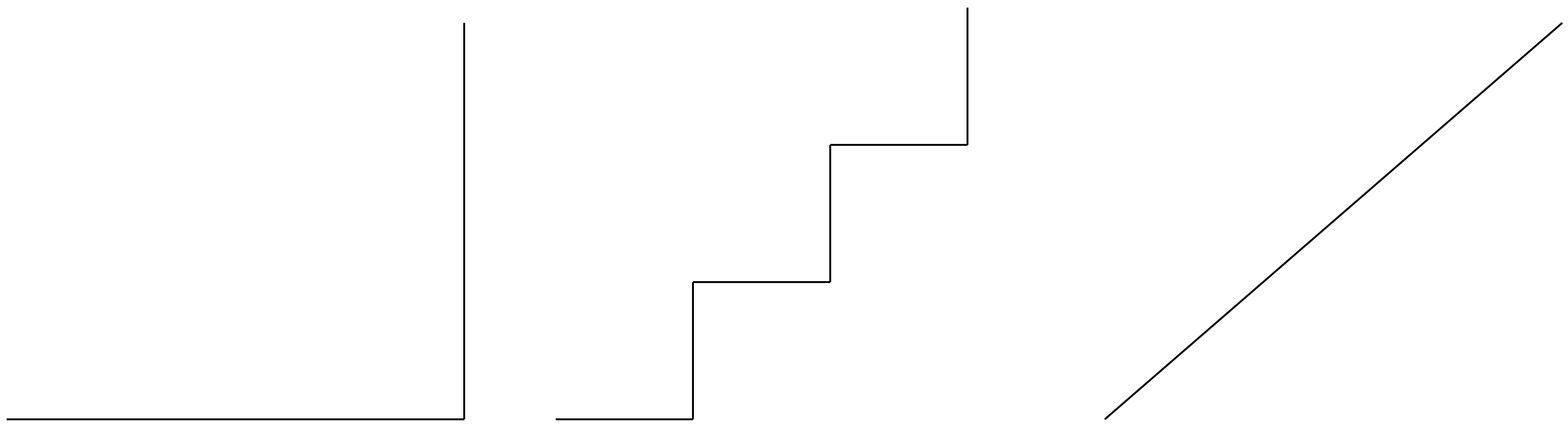


$$d(a,b) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}$$

Path Length
Which is shortest?



Path Length
Depends on metric



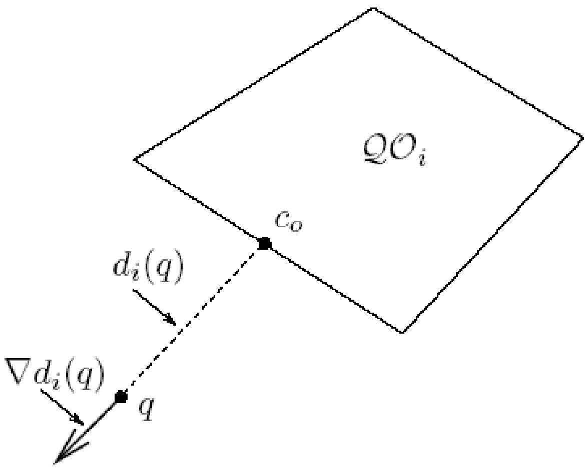
THE ROBOTICS INSTITUTE

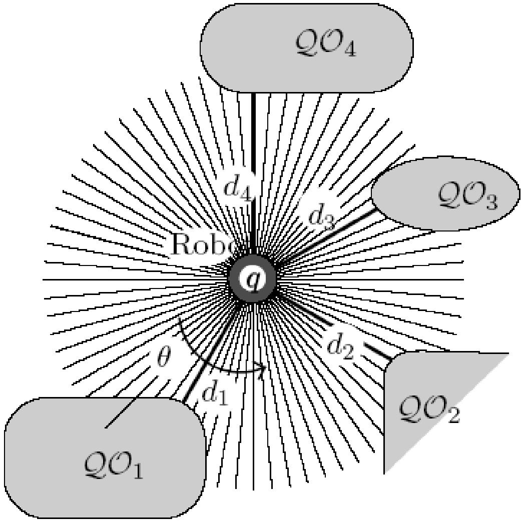
Distance to Obstacle(s)

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

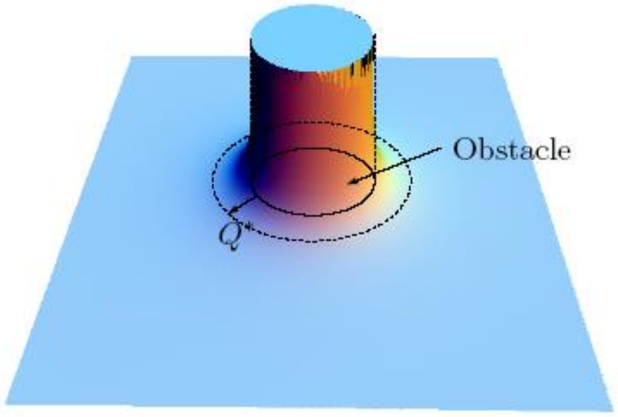
$$\nabla d_i(q) = \frac{q - c}{d(q, c)}$$

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





The Repulsive Potential

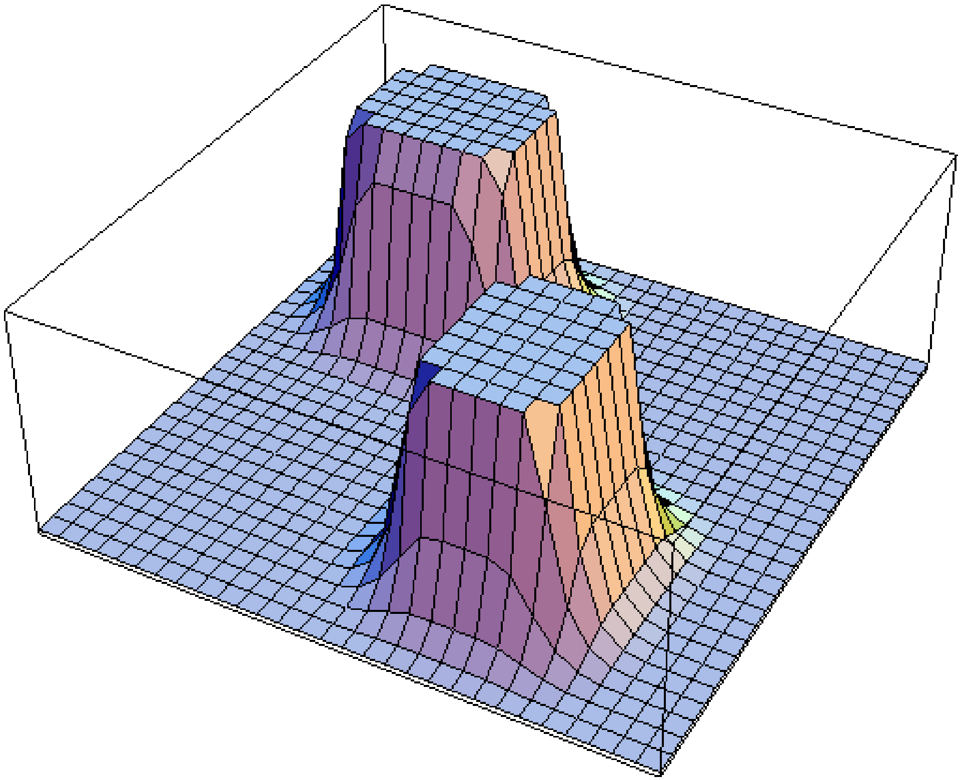


$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{D(q)} - \frac{1}{Q^*})^2, & D(q) \leq Q^*, \\ 0, & D(q) > Q^*, \end{cases}$$

whose gradient is

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

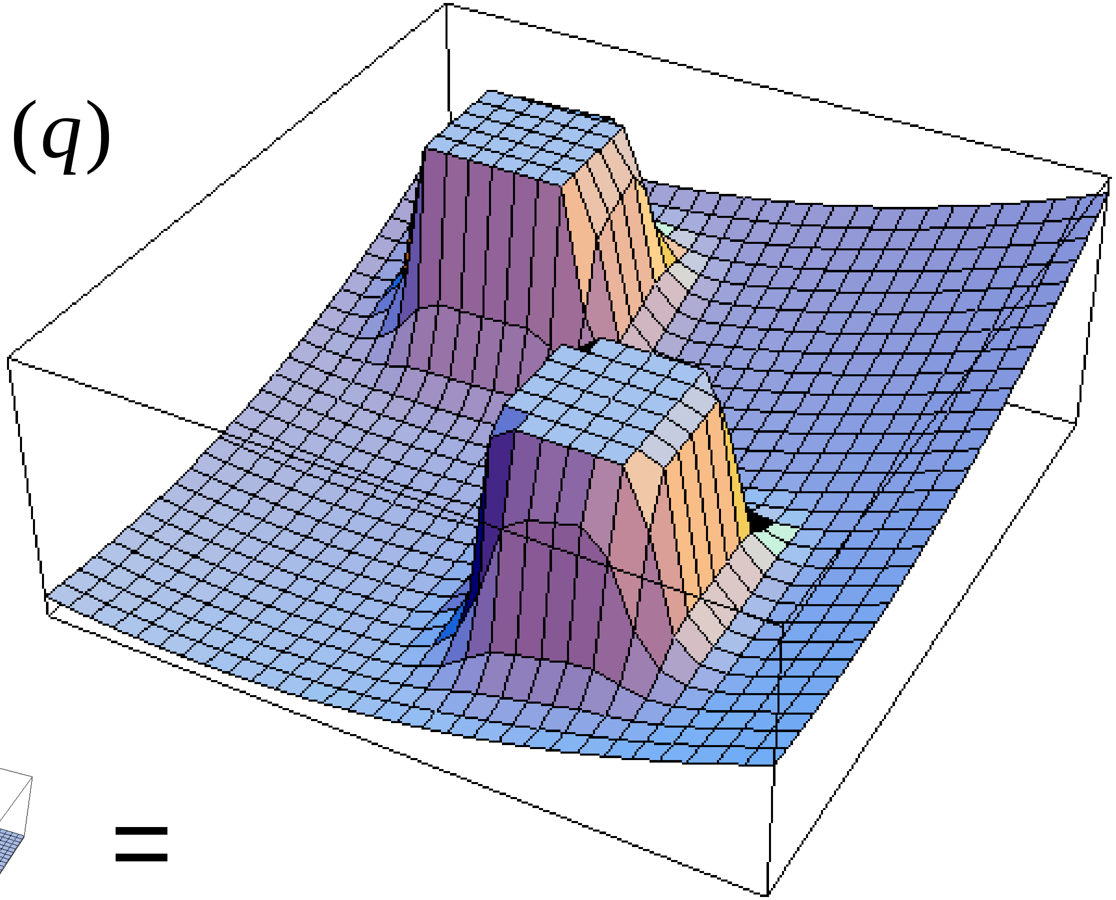
Repulsive Potential

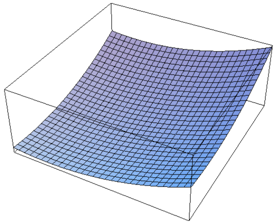


Total Potential Function

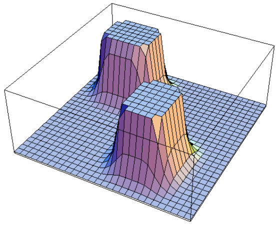
$$U(q) = U_{att}(q) + U_{rep}(q)$$

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



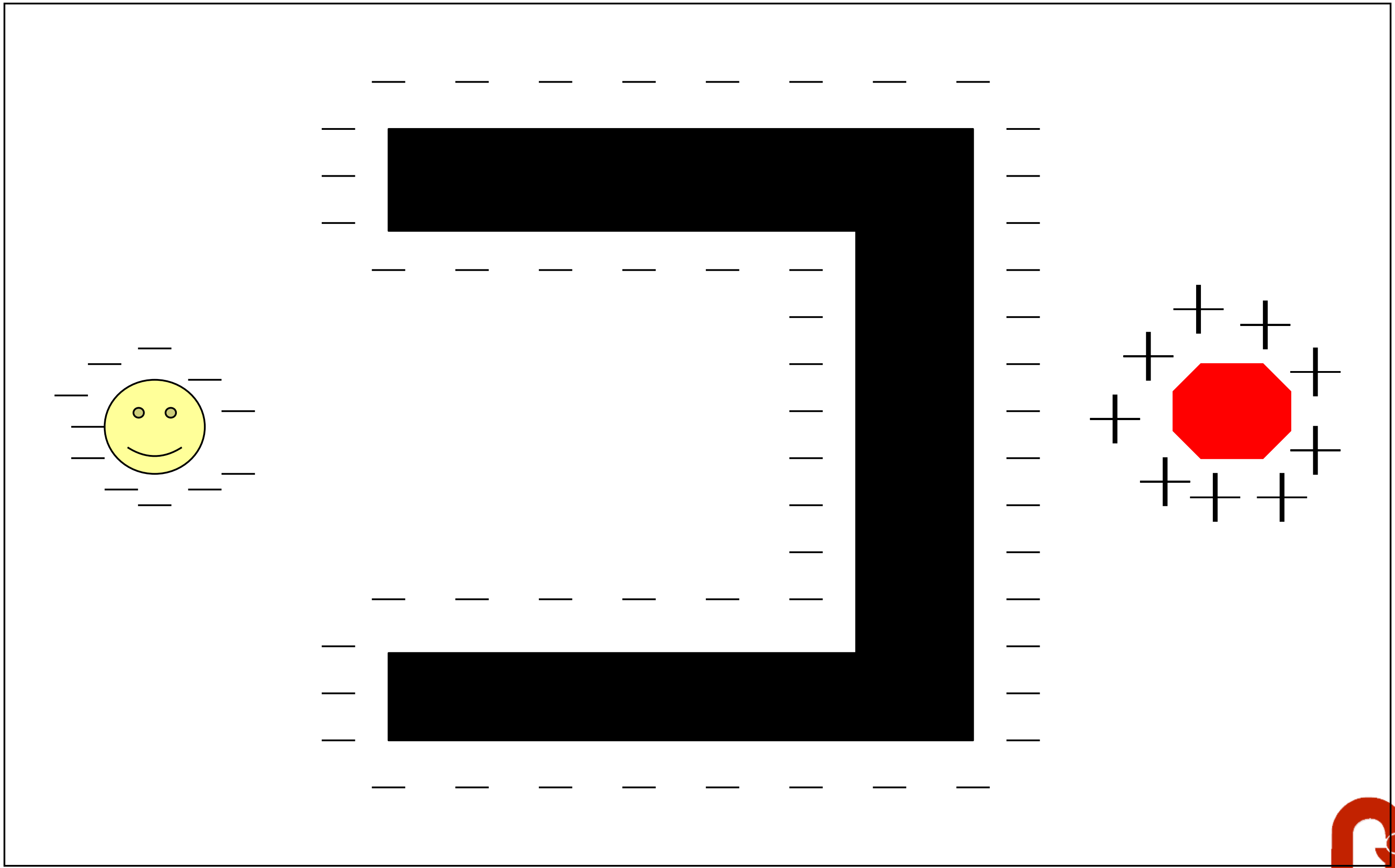


+

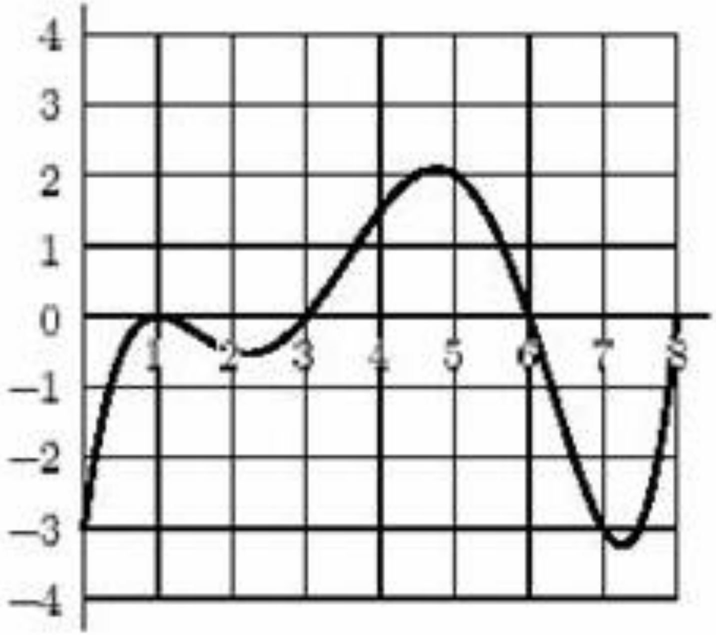


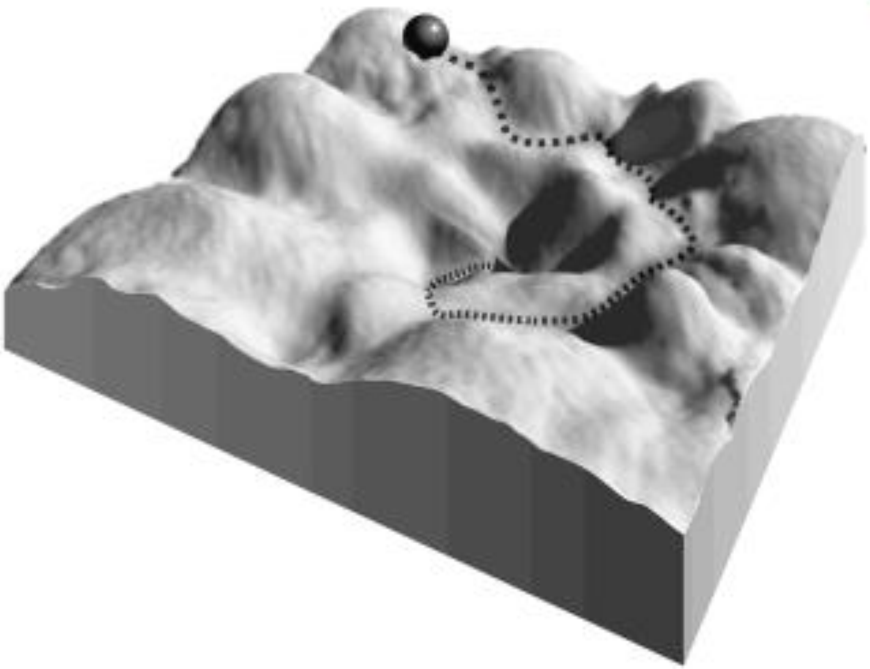
=

Local Minimum Problem with the Charge Analogy



Local Min



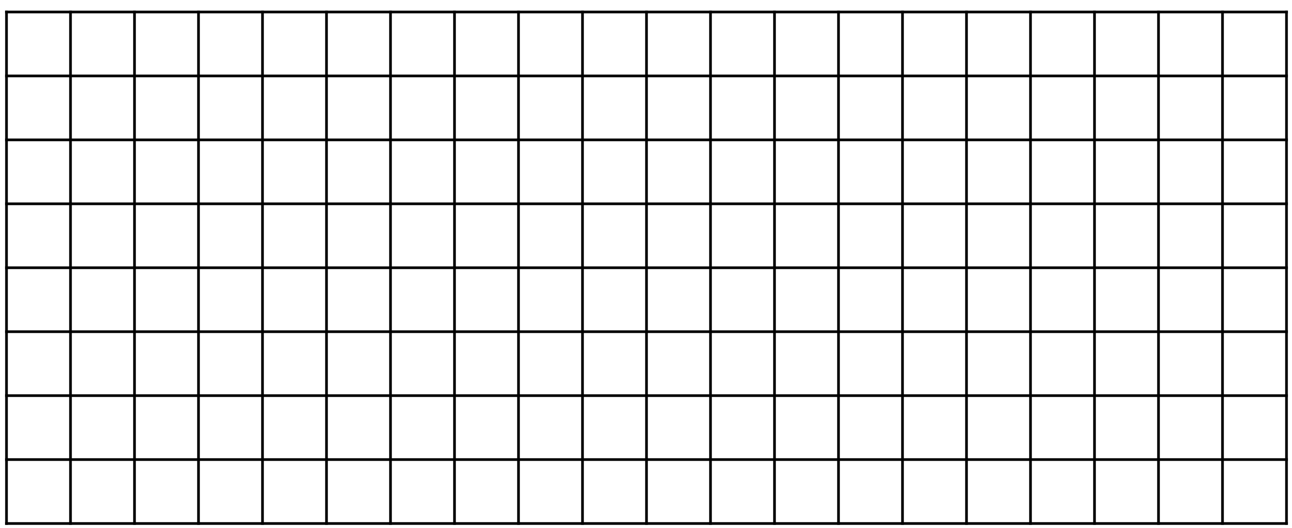


The Wavefront Planner

- 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

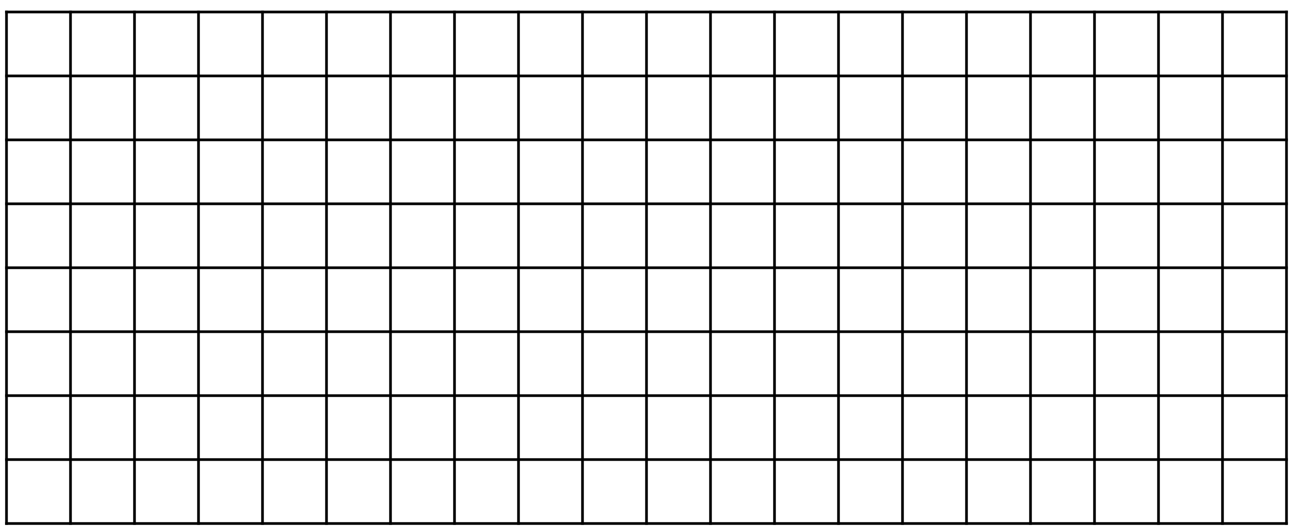
Representations

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



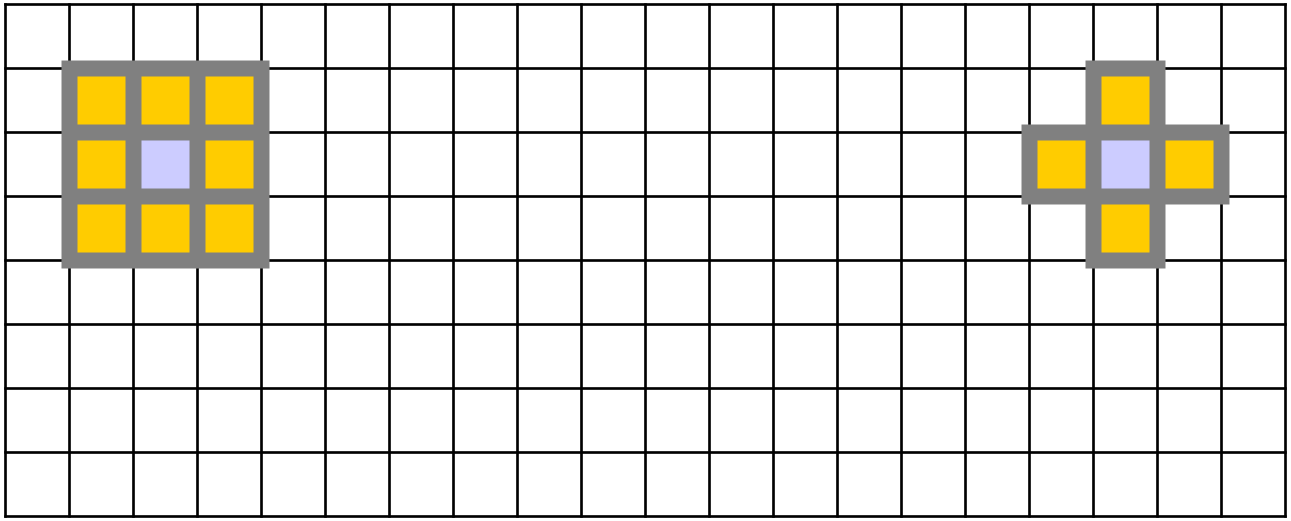
Representations: A Grid

- 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

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

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 ROBOTICS INSTITUTE

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)

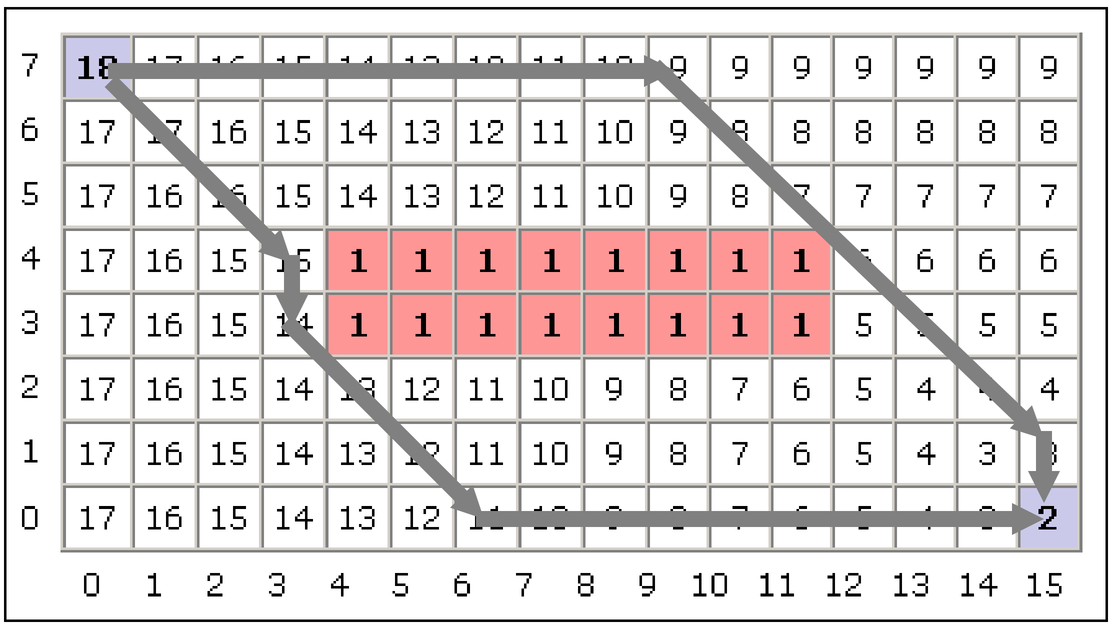
- You're done
  - Remember, 0's should only remain if unreachable regions exist

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

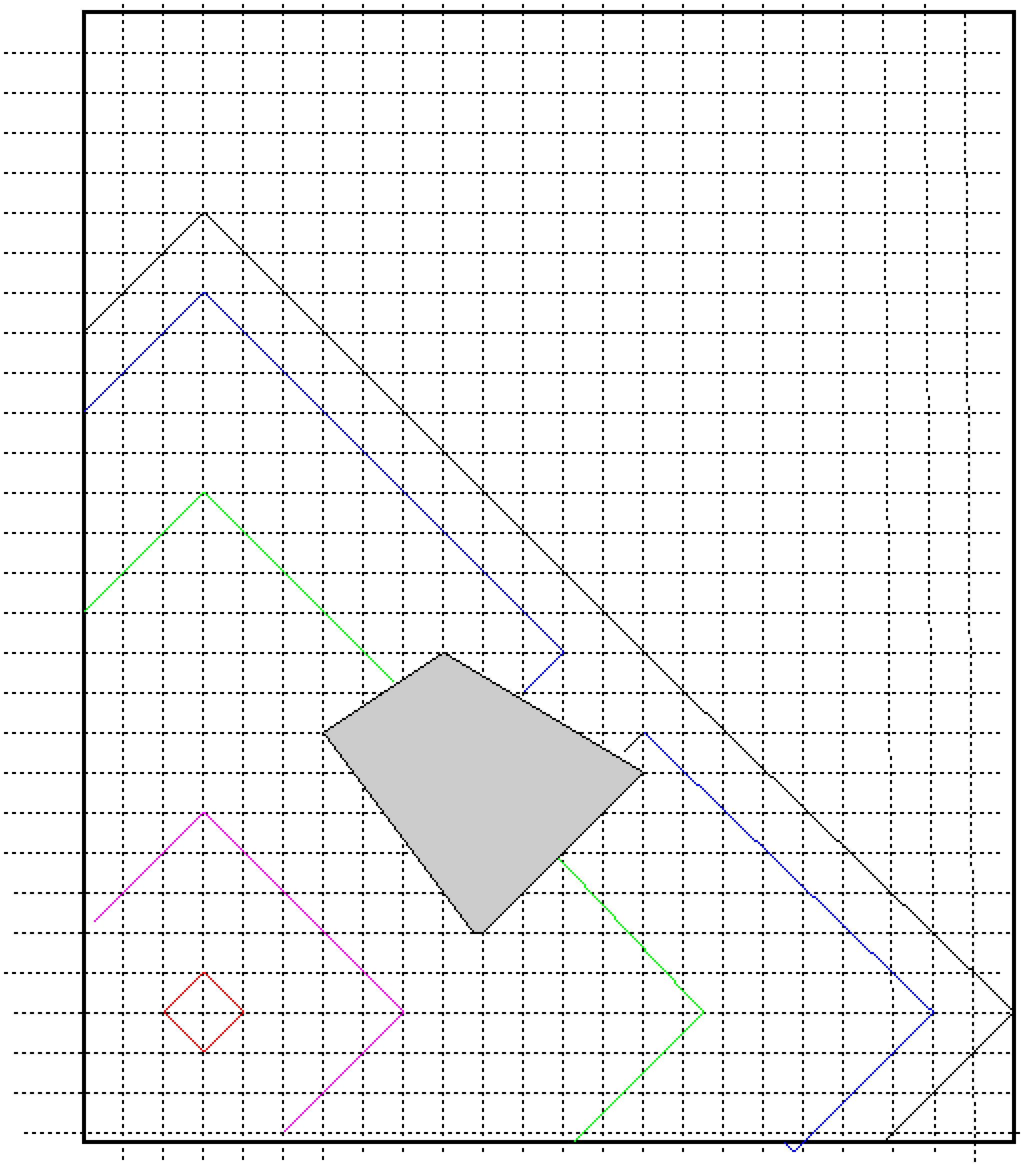
Two possible shortest paths shown



Wavefront (Overview)

- Divide the space into a grid.
- Number the squares starting at the start in either 4 or 8 point connectivity starting at the goal, increasing till you reach the start.
- Your path is defined by any uninterrupted sequence of decreasing numbers that lead to the goal.

This is really a Continuous Solution



Not pixels

Waves bend

L1 distance

THE ROBOTICS INSTITUTE

Rapidly-Exploring Random Tree