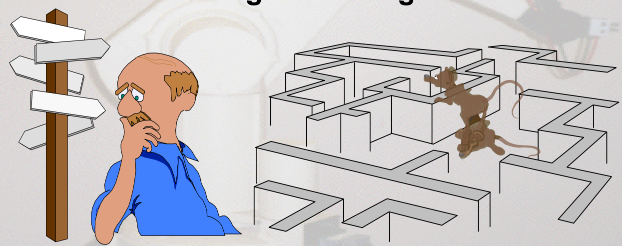
Motion Planning

by Ahmet ÖZKURT

Based on the Notes by
Vincent Lee-Shue
24-354: General Robotics

What is Motion Planning?

- Determining where to go



Overview

- The Basics
  - Motion Planning Statement
  - The World and Robot
  - Configuration Space
  - Metrics
- Path Planning Algorithms
  - Start-Goal Methods
  - Map-Based Approaches
  - Cellular Decompositions
- Applications
  - Coverage

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 C_i denotes the i'th obstacle,

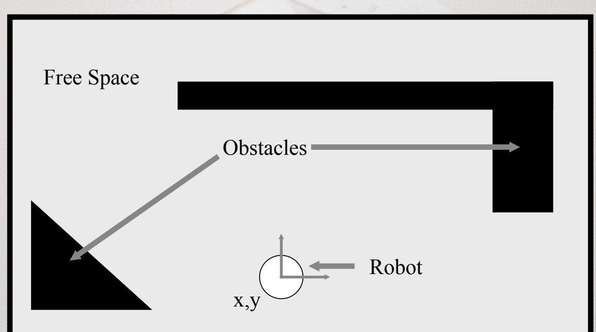
Then the robot's free space, FS, is defined as:

$$FS = W - (\cup C_i)$$

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

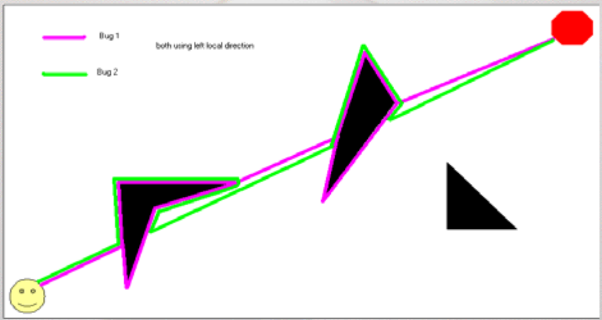
where $c(0)$ is q_{start} and $c(1)$ is q_{goal}

Example of a World (and Robot)



7

Start-Goal Algorithm: Lumelsky Bug Algorithms



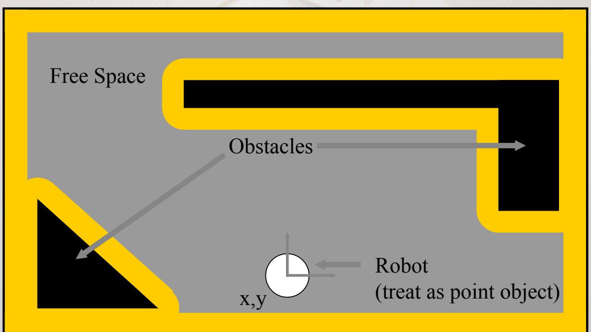
8

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**

9

Configuration Space: Accommodate Robot Size



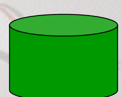
10

The Configuration Space

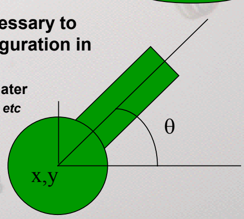
- **What it is**
  - **A set of "reachable" areas constructed from knowledge of both the robot and the world**
- **How to create it**
  - **First abstract the robot as a point object. Then, enlarge the obstacles to account for the robot's footprint and degrees of freedom**
  - **In our example, the robot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

11

Configuration Space: the robot has...

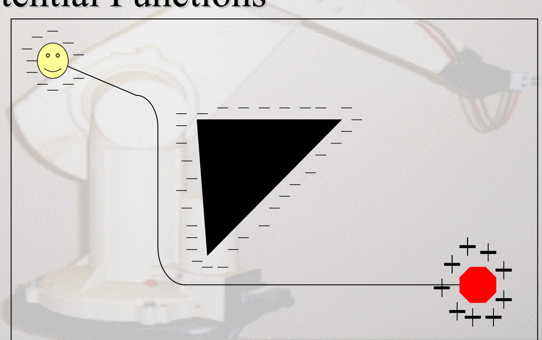


- **A Footprint**
  - **The amount of space a robot occupies**
- **Degrees of Freedom**
  - **The number of variables necessary to fully describe a robot's configuration in space**
    - **You'll cover this more in depth later**
    - ***fun with non-holonomic constraints, etc***



12

Start-Goal Algorithm: Potential Functions



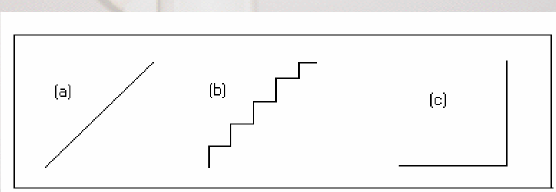
Potential Function Equations

Basics: Metrics

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

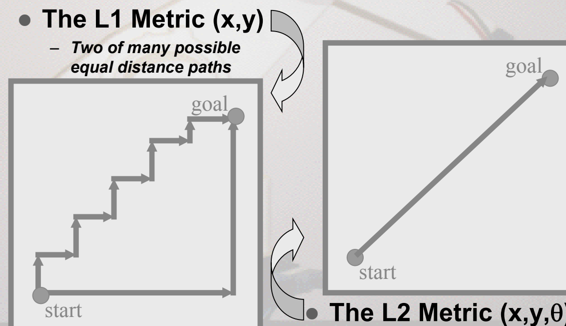
Basics: Movement Metrics

- Many ways to measure distance; two are:
  - L1 metric
    - (x,y) : $|x| + |y| = const$
  - L2 metric
    - (x,y) : $x^2 + y^2 = const$



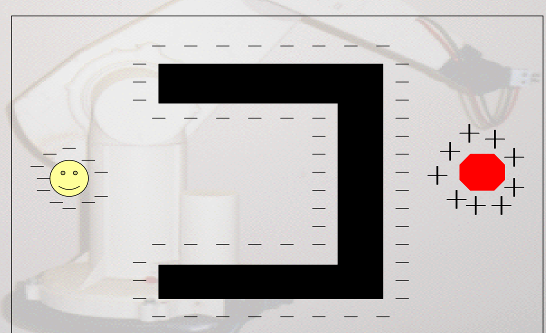
Basics: Movement Metrics

- The L1 Metric (x,y)
  - *Two of many possible equal distance paths*



- The L2 Metric (x,y,θ)

Local Minimum Problem with the Charge Analogy

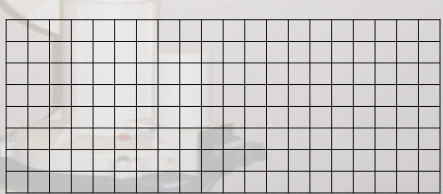


The Wavefront Planner

- 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

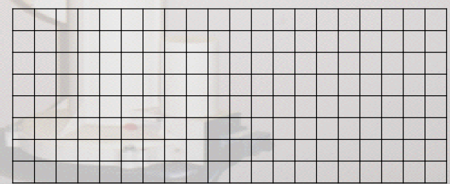
Representations

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



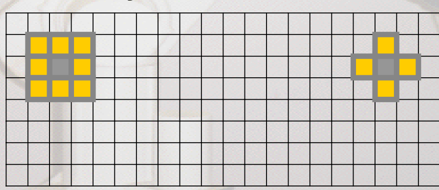
Representations: A Grid

- **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...**

	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**	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**

The Wavefront in Action (Part 4)

- **And again...**

	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**	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**

The Wavefront in Action (Part 5)

- **And again until...**

	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	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**

The Wavefront in Action (Done)

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

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

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

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

Wavefront (Overview)

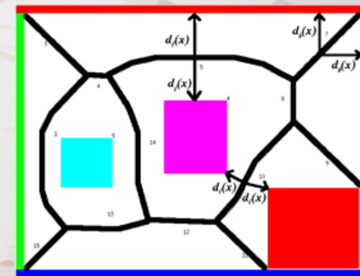
- **Divide the space into a grid.**
- **Number the squares starting at the start in either 4 or 8 point connectivity starting at the goal, increasing till you reach the start.**
- **Your path is defined by any uninterrupted sequence of decreasing numbers that lead to the goal.**

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

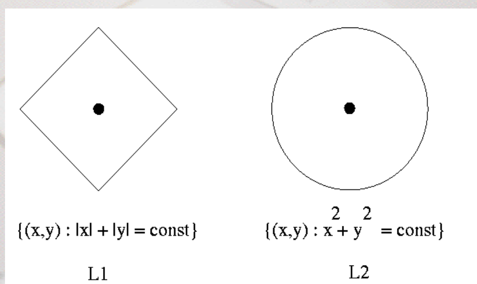
Roadmap: GVG

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



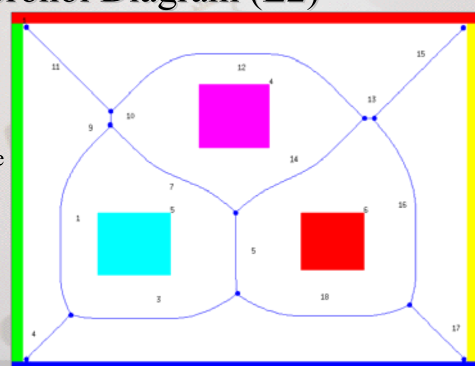
- This generates a very safe roadmap which avoids obstacles as much as possible

Voronoi Diagram: Metrics



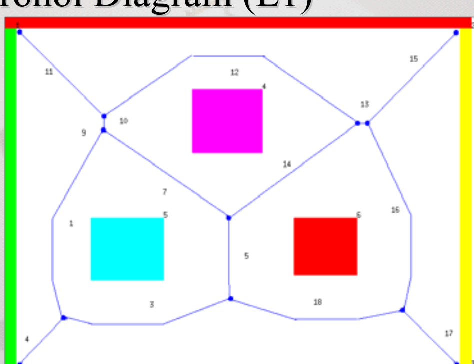
Voronoi Diagram (L2)

Note the curved edges



Voronoi Diagram (L1)

Note the lack of curved edges

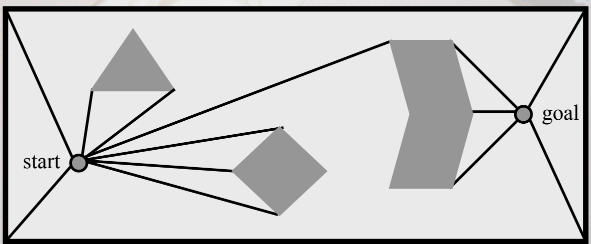


Roadmap: Visibility Graph

- 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

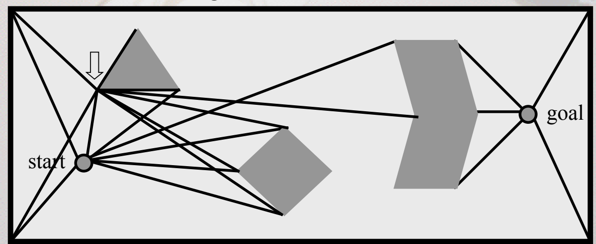
The Visibility Graph in Action (Part 1)

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



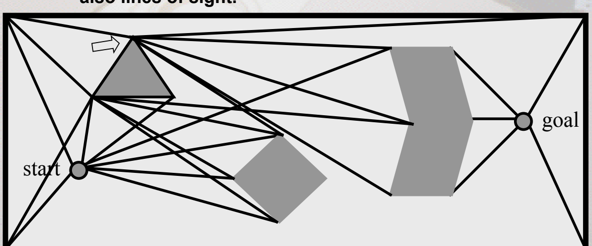
The Visibility Graph in Action (Part 2)

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



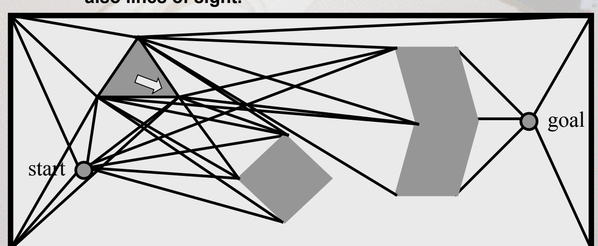
The Visibility Graph in Action (Part 3)

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



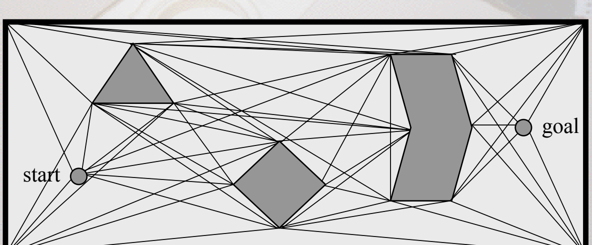
The Visibility Graph in Action (Part 4)

- 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.

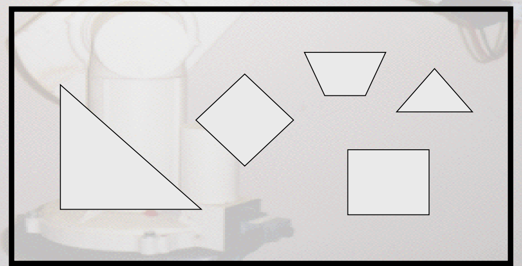


Visibility Graph Overview

- 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.

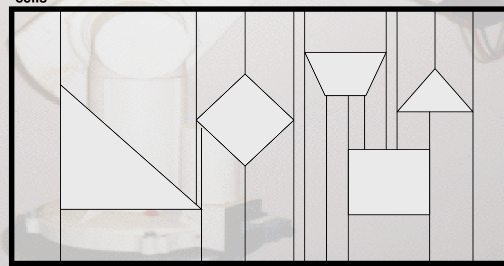
Cell Decompositions: Trapezoidal Decomposition

- A way to divide the world into smaller regions
- Assume a polygonal world



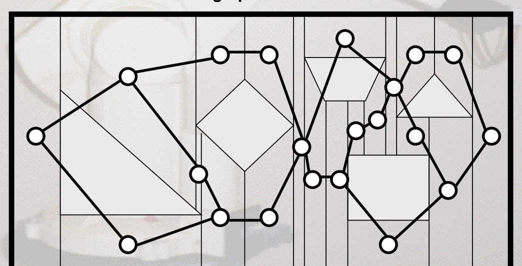
Cell Decompositions: Trapezoidal Decomposition

- Simply draw a vertical line from each vertex until you hit an obstacle. This reduces the world to a union of trapezoid-shaped cells



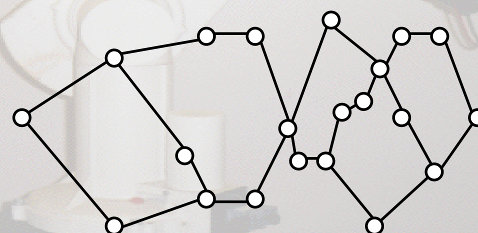
Applications: Coverage

- By reducing the world to cells, we've essentially abstracted the world to a graph.



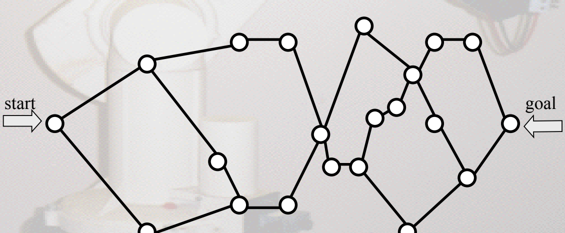
Find a path

- By reducing the world to cells, we've essentially abstracted the world to a graph.



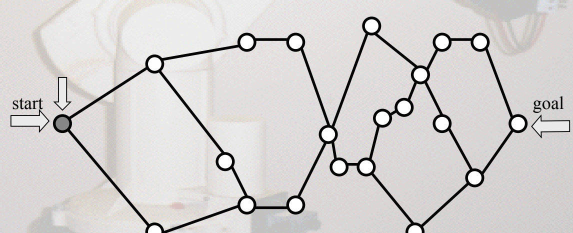
Find a path

- With an adjacency graph, a path from start to goal can be found by simple traversal



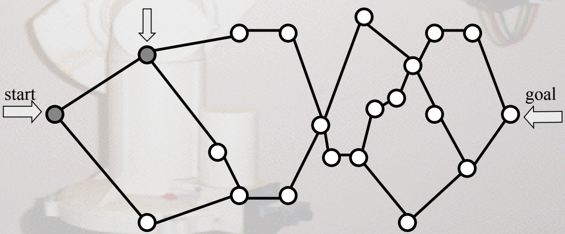
Find a path

- With an adjacency graph, a path from start to goal can be found by simple traversal



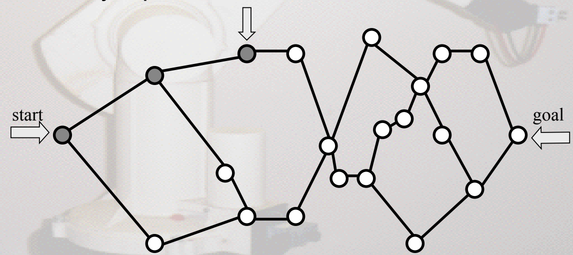
Find a path

- **With an adjacency graph, a path from start to goal can be found by simple traversal**



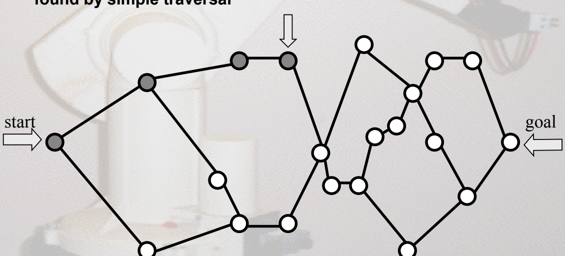
Find a path

- **With an adjacency graph, a path from start to goal can be found by simple traversal**



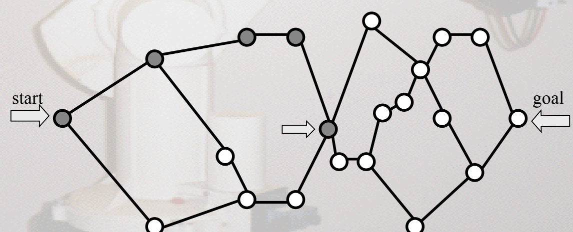
Find a path

- **With an adjacency graph, a path from start to goal can be found by simple traversal**



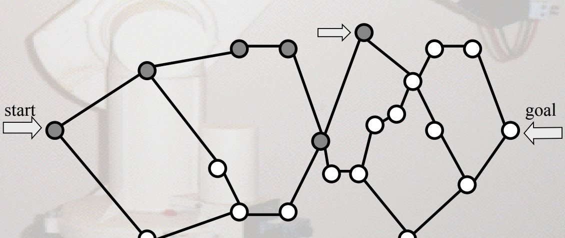
Find a path

- **With an adjacency graph, a path from start to goal can be found by simple traversal**



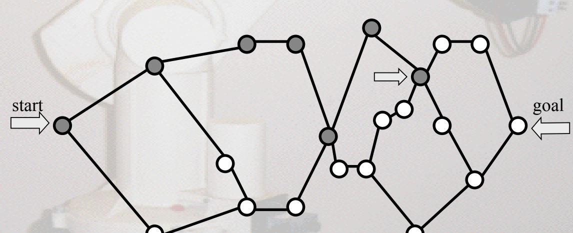
Find a path

- **With an adjacency graph, a path from start to goal can be found by simple traversal**



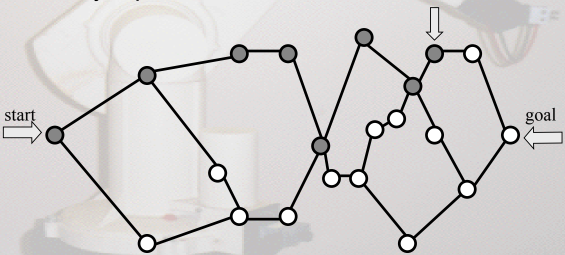
Find a path

- **With an adjacency graph, a path from start to goal can be found by simple traversal**



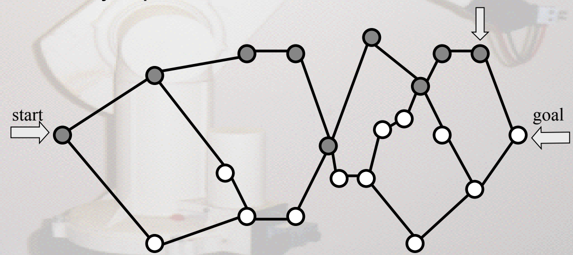
Find a path

- With an adjacency graph, a path from start to goal can be found by simple traversal



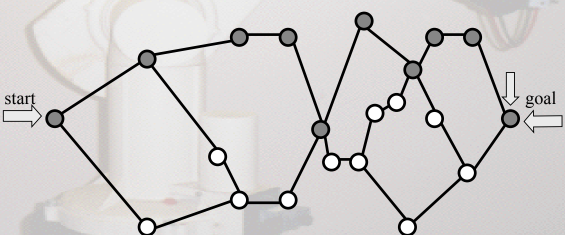
Find a path

- With an adjacency graph, a path from start to goal can be found by simple traversal



Find a path

- With an adjacency graph, a path from start to goal can be found by simple traversal

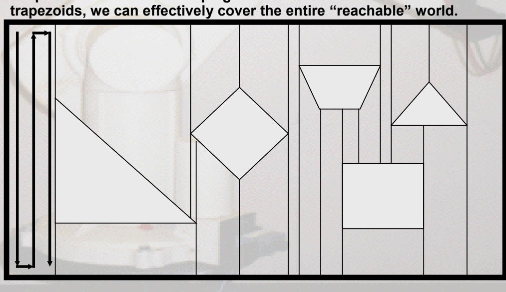


Applications: Coverage

- **First, a distinction between sensor and detector must be made**
- ***Sensor*: Senses obstacles**
- ***Detector*: What actually does the coverage**
- **We'll be observing the simple case of having an omniscient sensor and having the detector's footprint equal to the robot's footprint**

Cell Decompositions: Trapezoidal Decomposition

- How is this useful? Well, trapezoids can easily be covered with simple back-and-forth sweeping motions. If we cover all the trapezoids, we can effectively cover the entire "reachable" world.



Applications: Coverage

- Simply visit all the nodes, performing a sweeping motion in each, and you're done.

Conclusion: Complete Overview

- **The Basics**
  - Motion Planning Statement
  - The World and Robot
  - Configuration Space
  - Metrics
- **Path Planning Algorithms**
  - Start-Goal Methods
    - Lumelsky Bug Algorithms
    - Potential Charge Functions
    - The Wavefront Planner
  - Map-Based Approaches
    - Generalized Voronoi Graphs
    - Visibility Graphs
  - Cellular Decompositions => Coverage
- ***Done with Motion Planning!***

You may now rejoice!

(for now…)