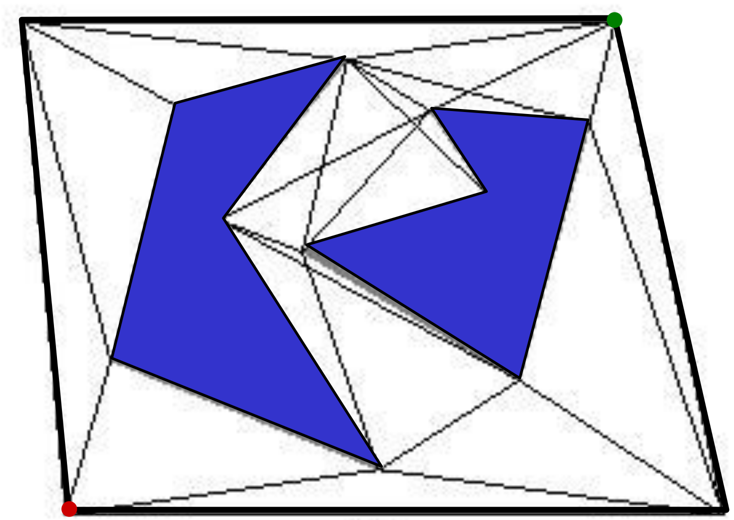
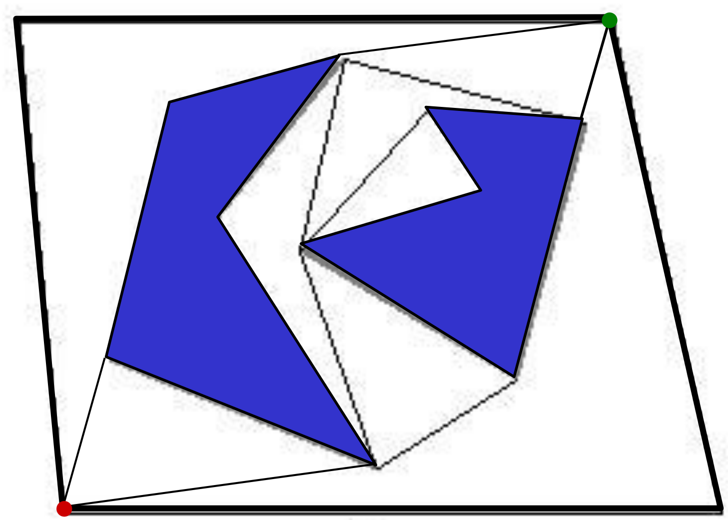
Roadmaps

Vertex Visibility Graph



Full visibility graph

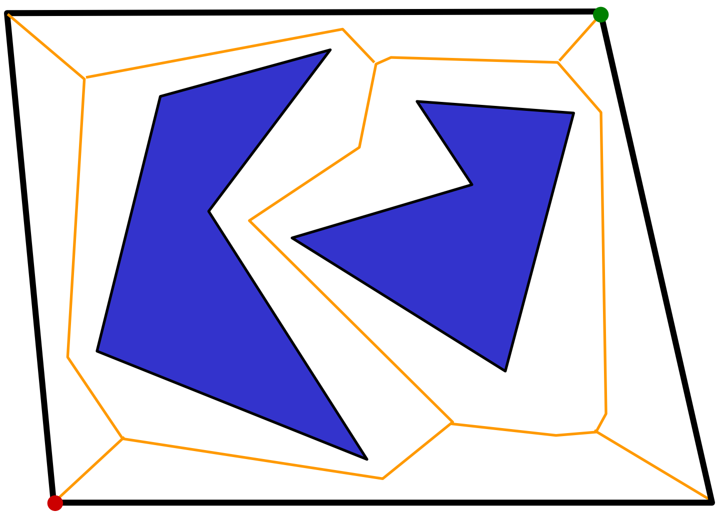


Reduced visibility graph, i.e., not including segments that extend into obstacles on either side.

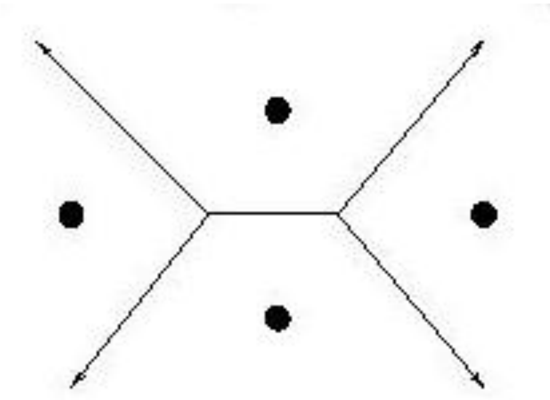
(but keeping endpoints' roads)

what else might we do ?

An alternative roadmap



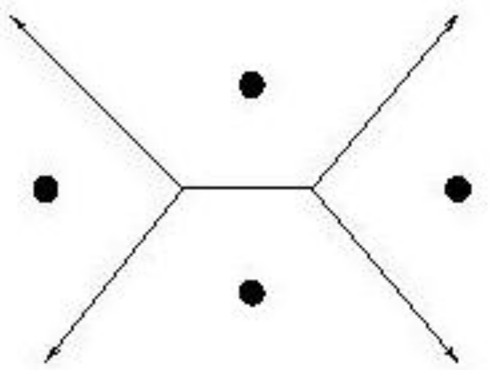
Voronoi diagrams



These line segments make up the **Voronoi diagram** for the four points shown here.

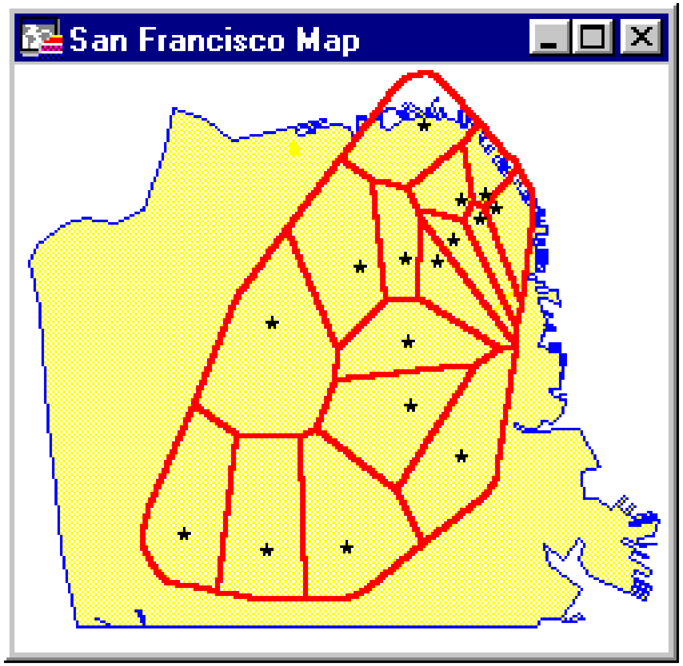
Solves the "Post Office Problem"

Voronoi diagrams



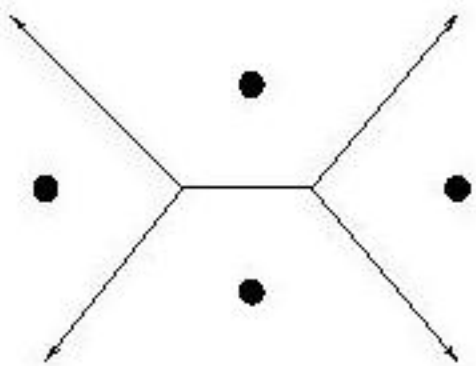
These line segments make up the **Voronoi diagram** for the four points shown here.

Solves the “Post Office Problem”



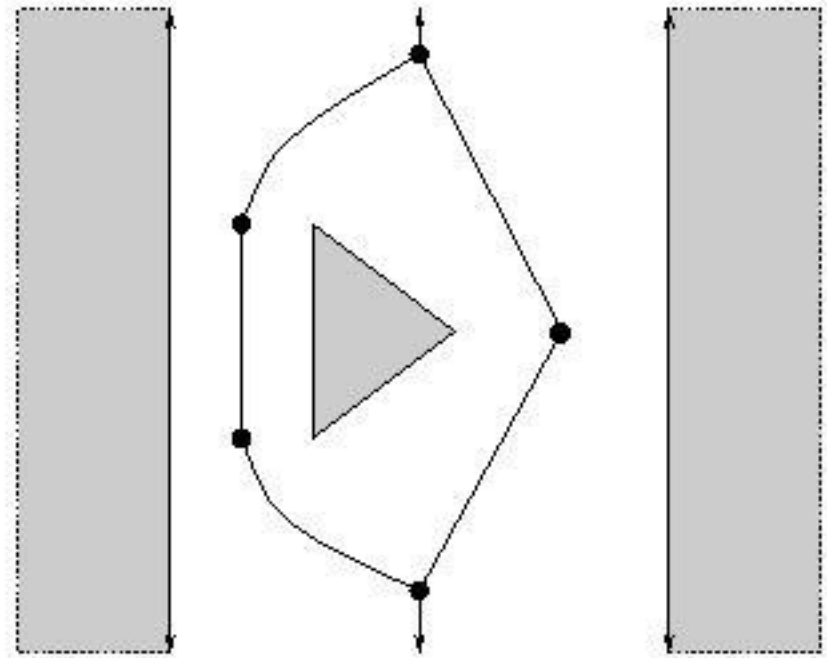
or, perhaps, more important problems...

Voronoi diagrams



“true” Voronoi diagram

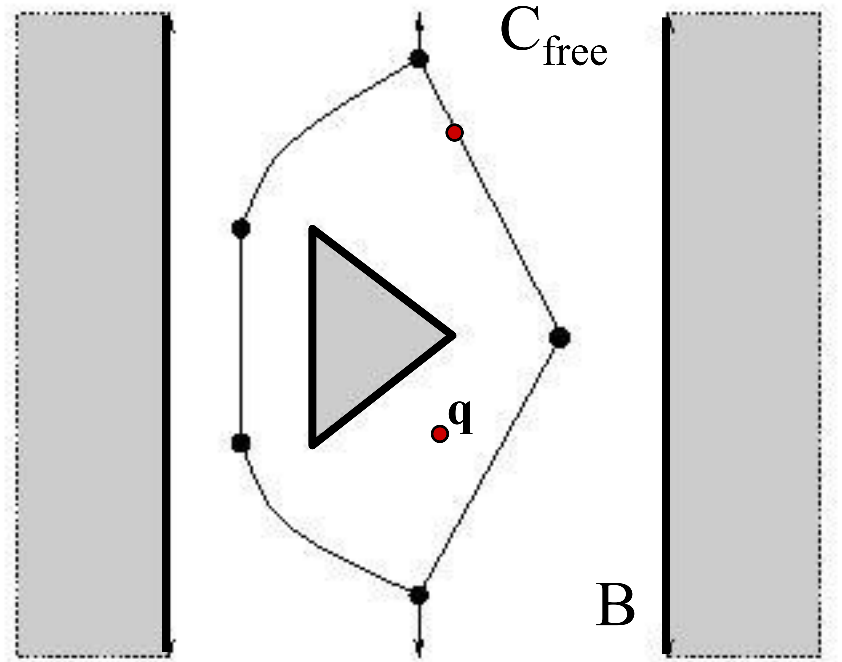
(isolates a set of points)



generalized Voronoi diagram

What is it?

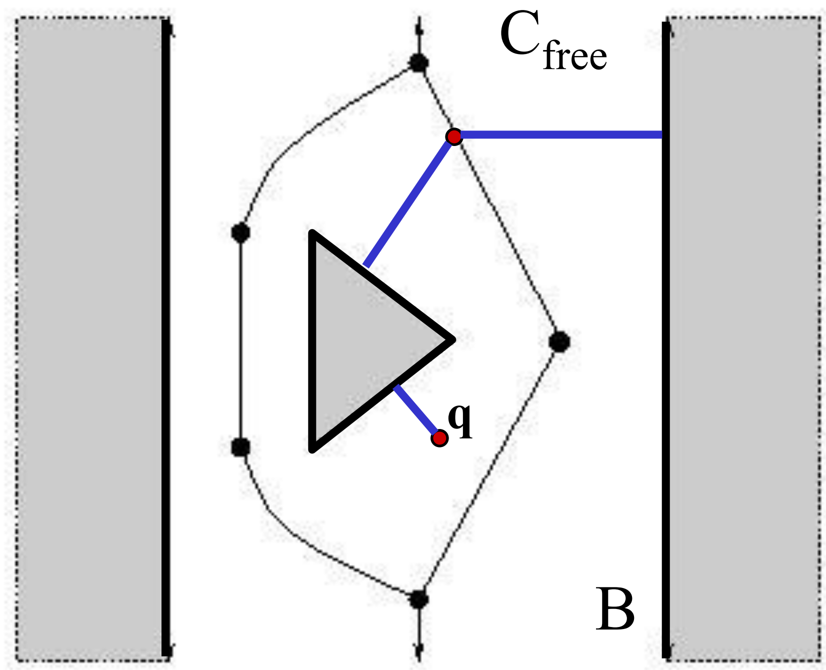
Voronoi diagrams



Let B = the boundary of C_{free} .

Let **q** be a point in C_{free} . (•)

Voronoi diagrams

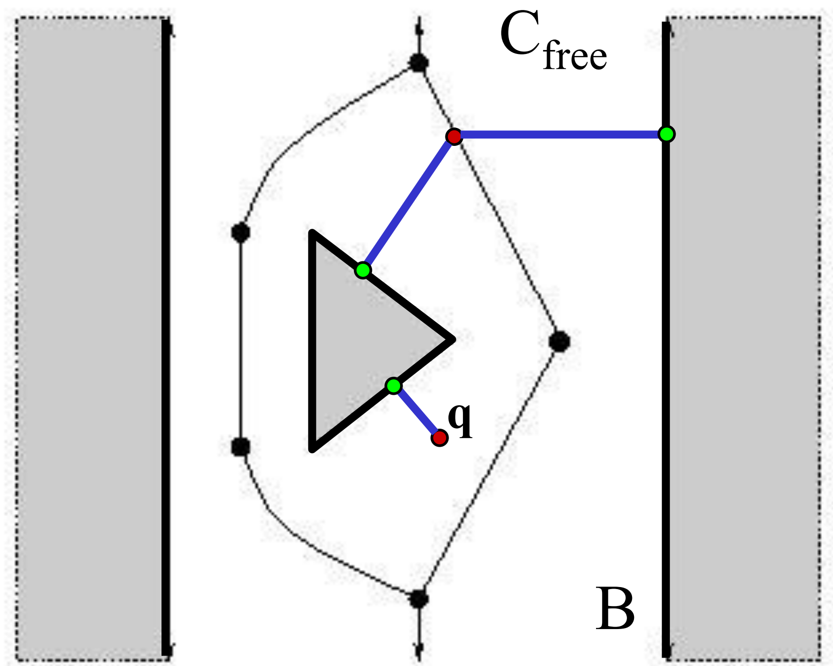


Let B = the boundary of C_{free} .

Let **q** be a point in C_{free} .

Define *clearance*(q) = min { | q - p | }, for all p ∈ B

Voronoi diagrams



Let B = the boundary of C_{free} .

Let **q** be a point in C_{free} .

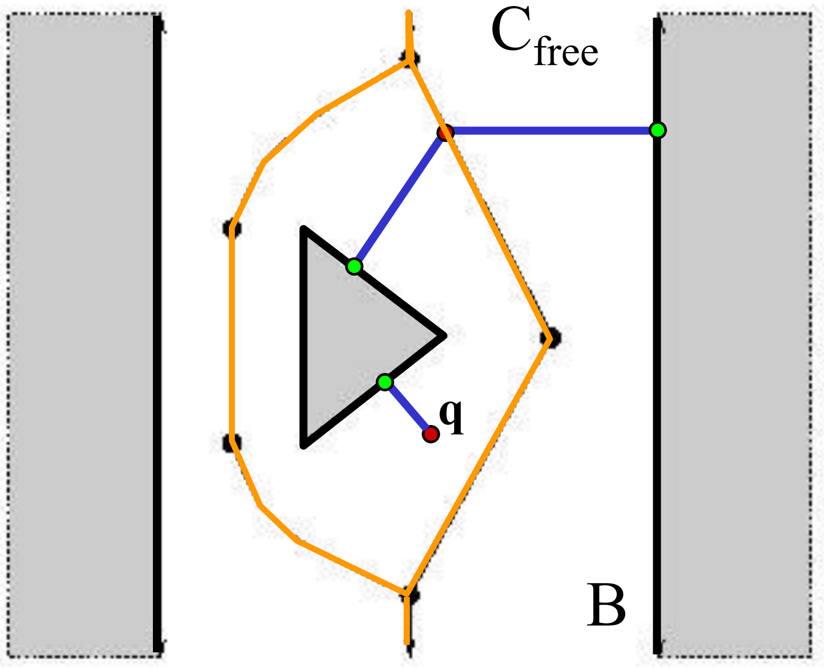
Define *clearance*(q) = min { | q - p | }, for all p ∈ B

Define *near*(q) = { p ∈ B such that | q - p | = *clearance*(q) }

Voronoi diagrams

Evaluation

+ maximizes distance from obstacles
+ reduces to graph search
+ can be used in higher-dimensions
- nonoptimal
- real diagrams tend to be noisy



Let B = the boundary of C_{free} .

Let **q** be a point in C_{free} .

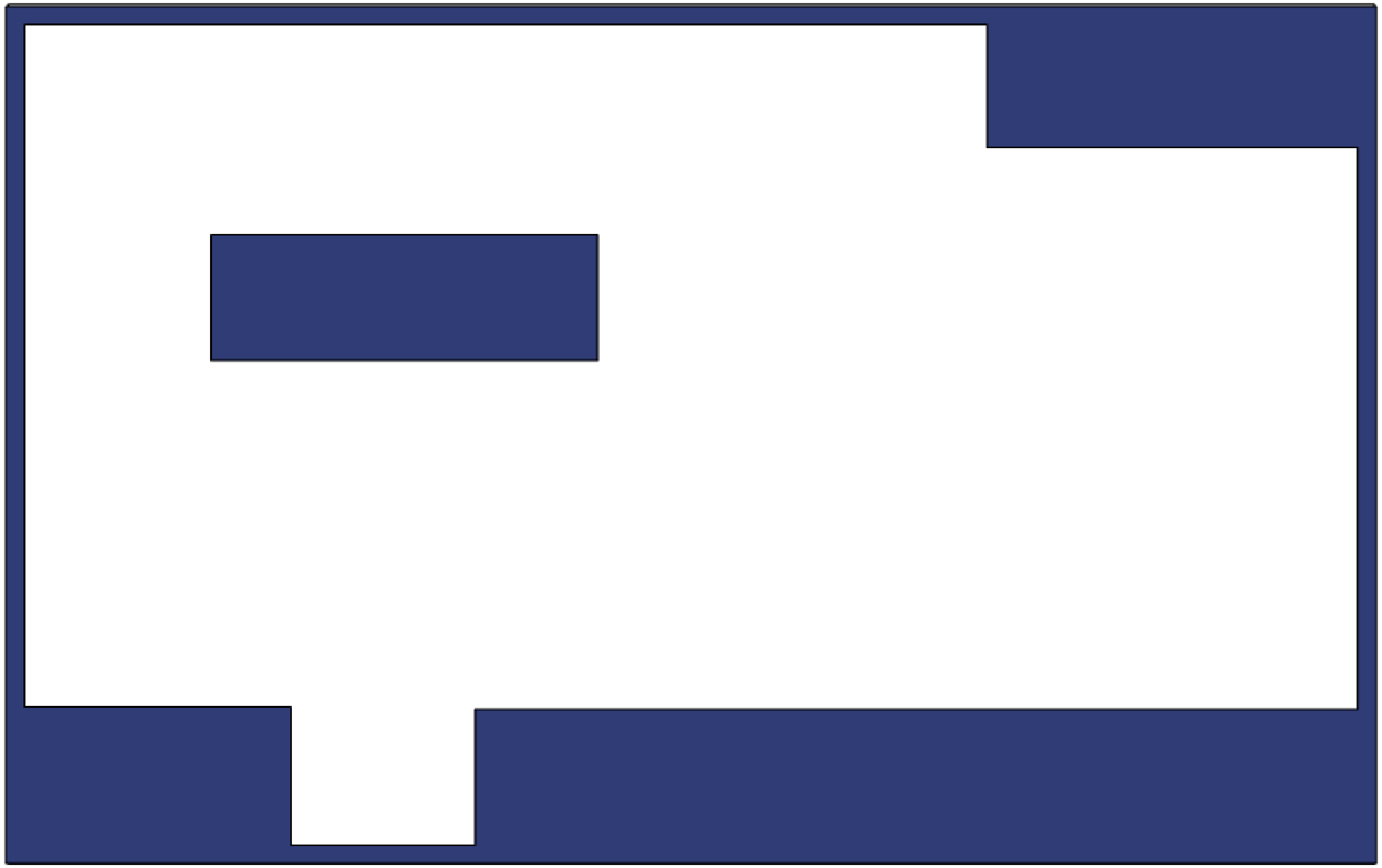
Define *clearance*(q) = min { | q - p | }, for all p $\in$ B

Define *near*(q) = { p $\in$ B such that | q - p | = *clearance*(q) }

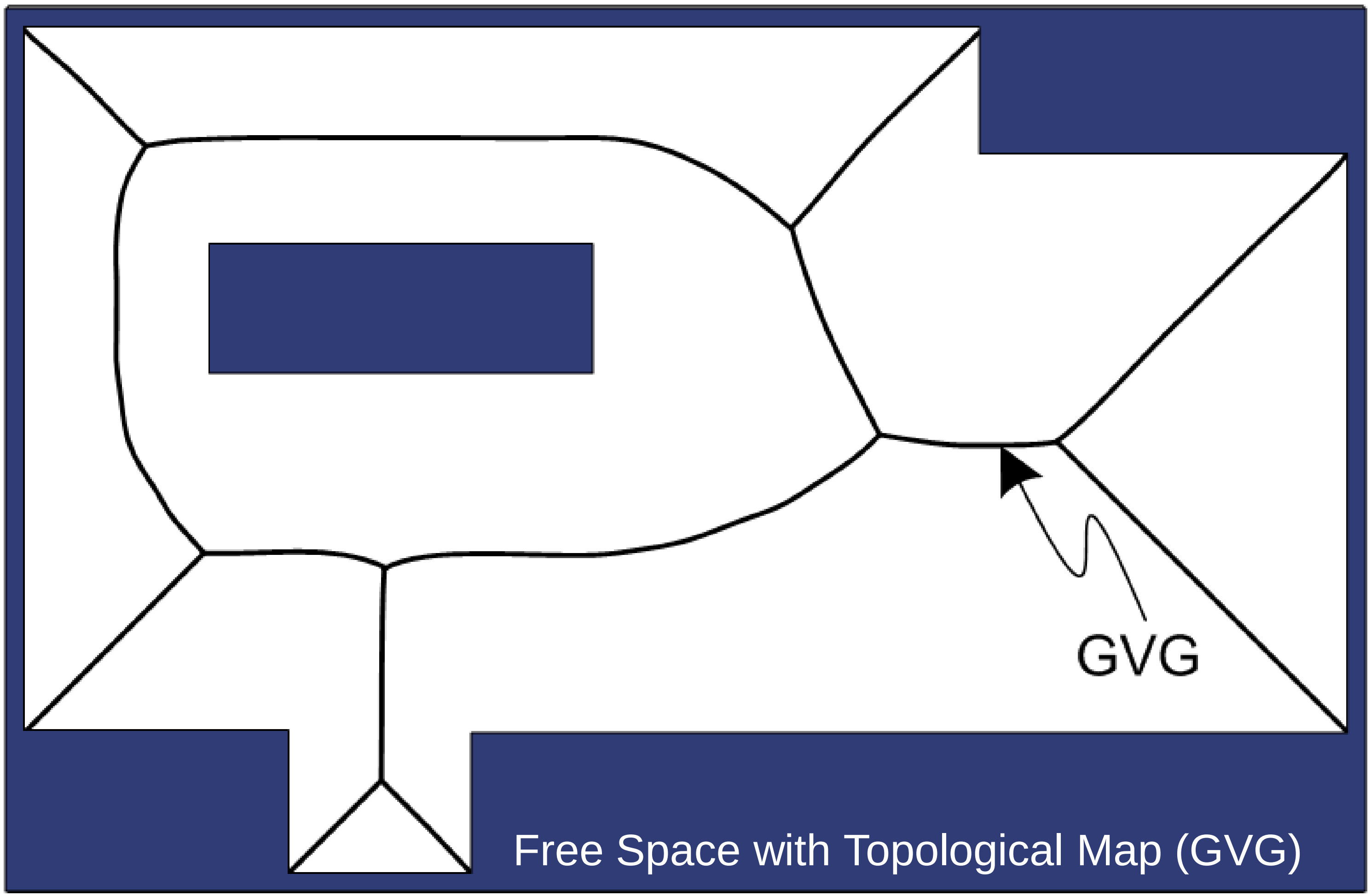
q is in the *Voronoi diagram* of C_{free} if | *near*(q) | > 1

number of set elements

Generalized Voronoi Graph (GVG)

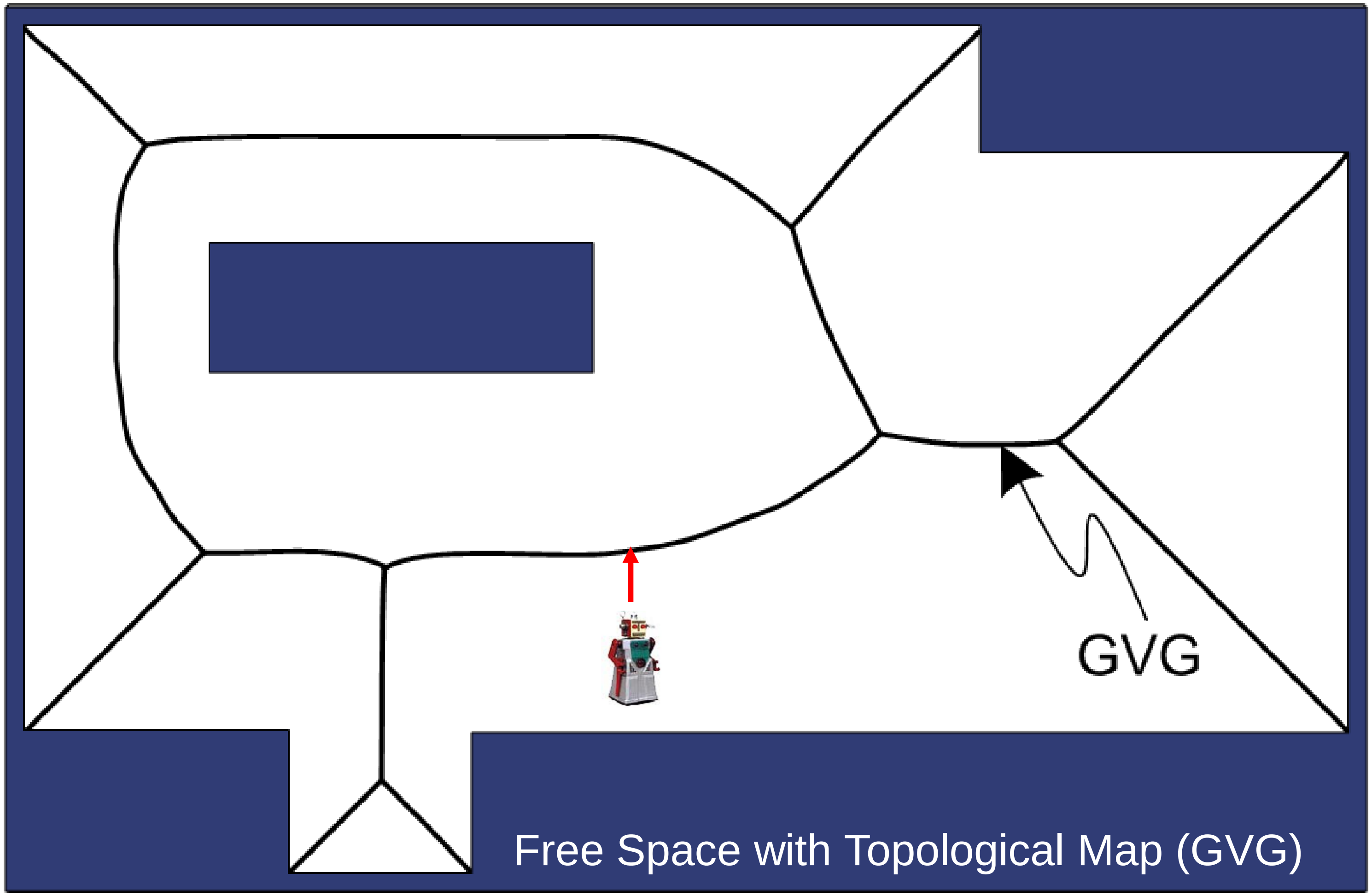


Generalized Voronoi Graph (GVG)



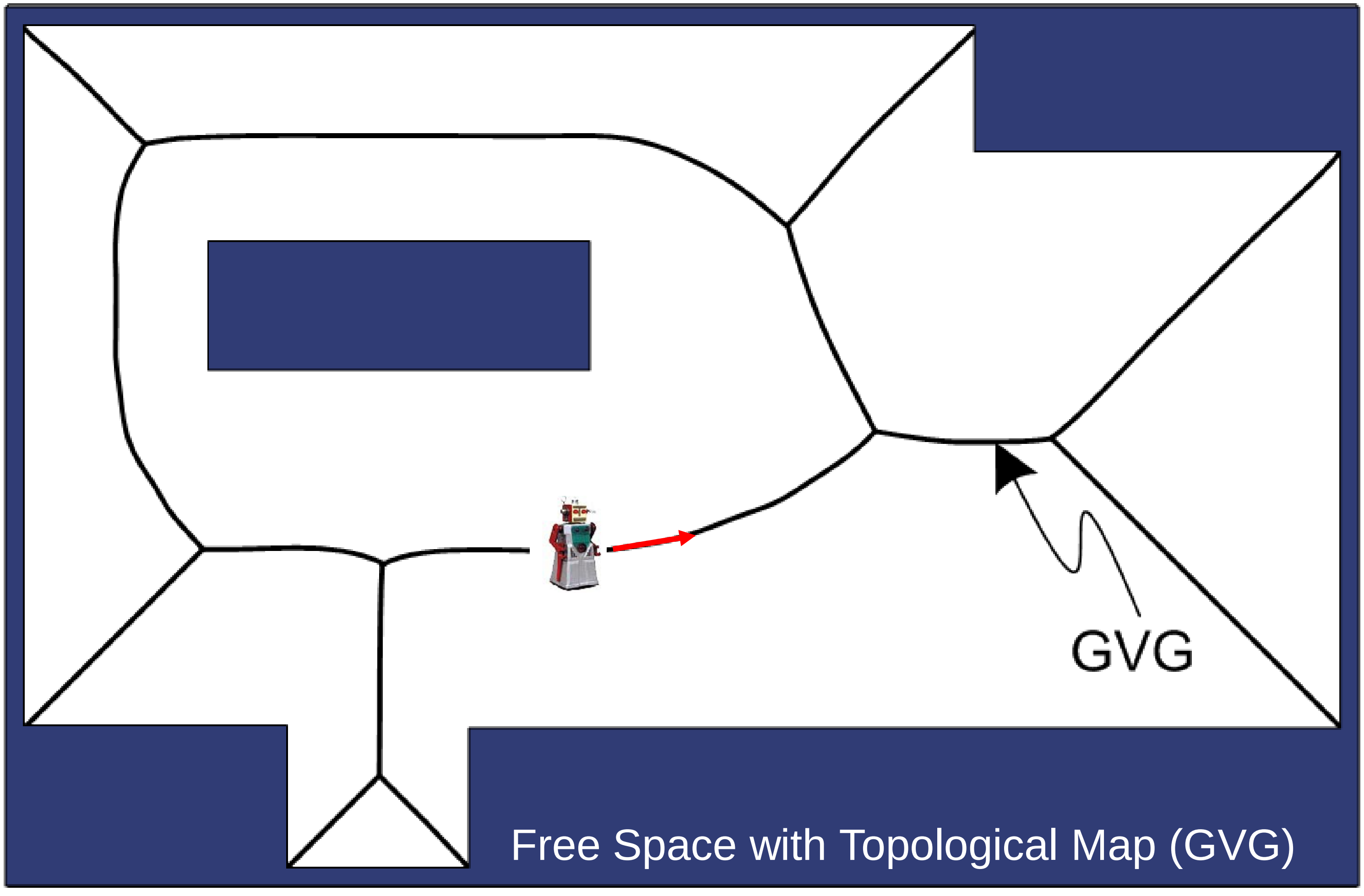
Generalized Voronoi Graph (GVG)

- Access GVG



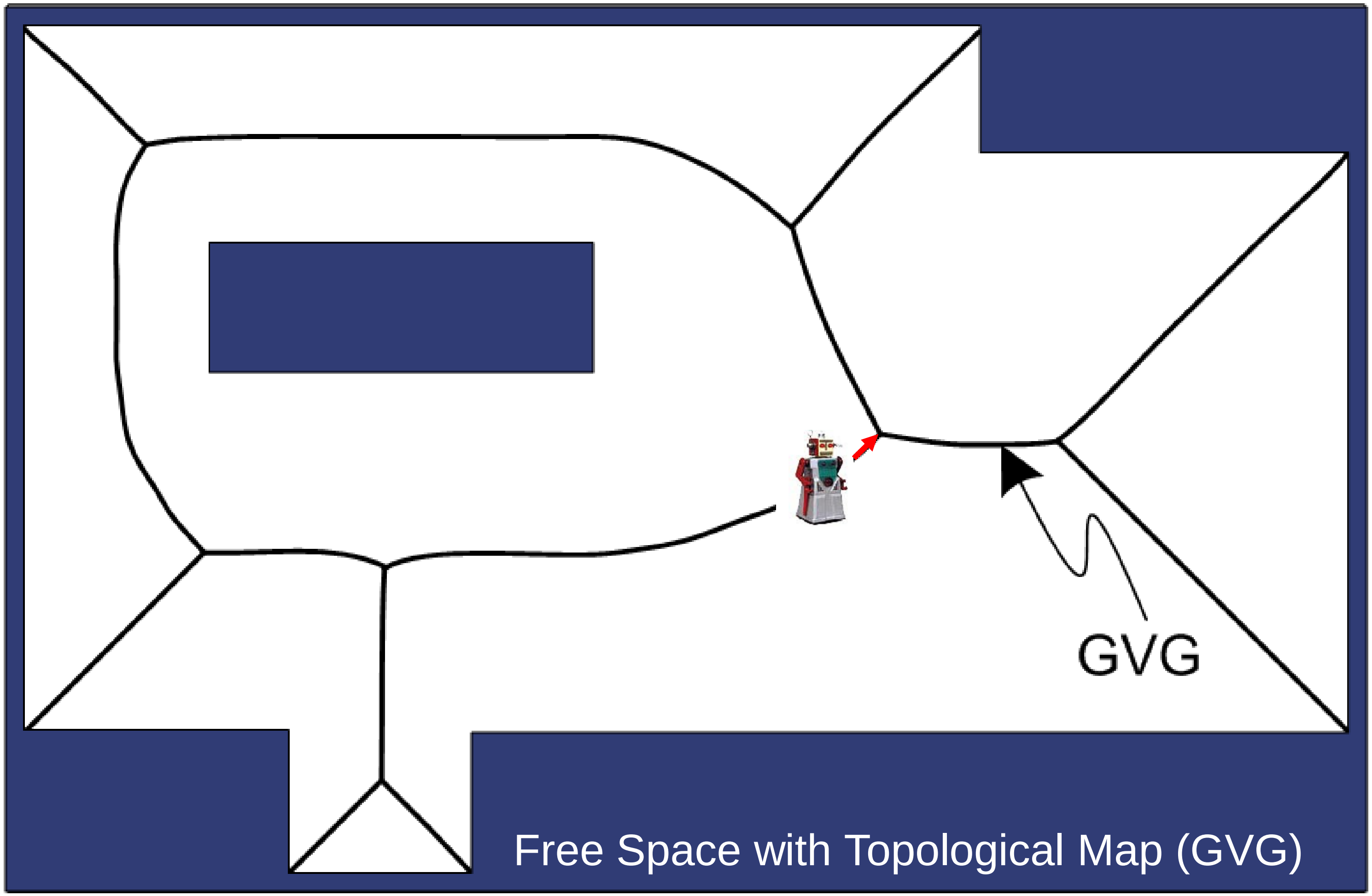
Generalized Voronoi Graph (GVG)

- Access GVG
- Follow Edge



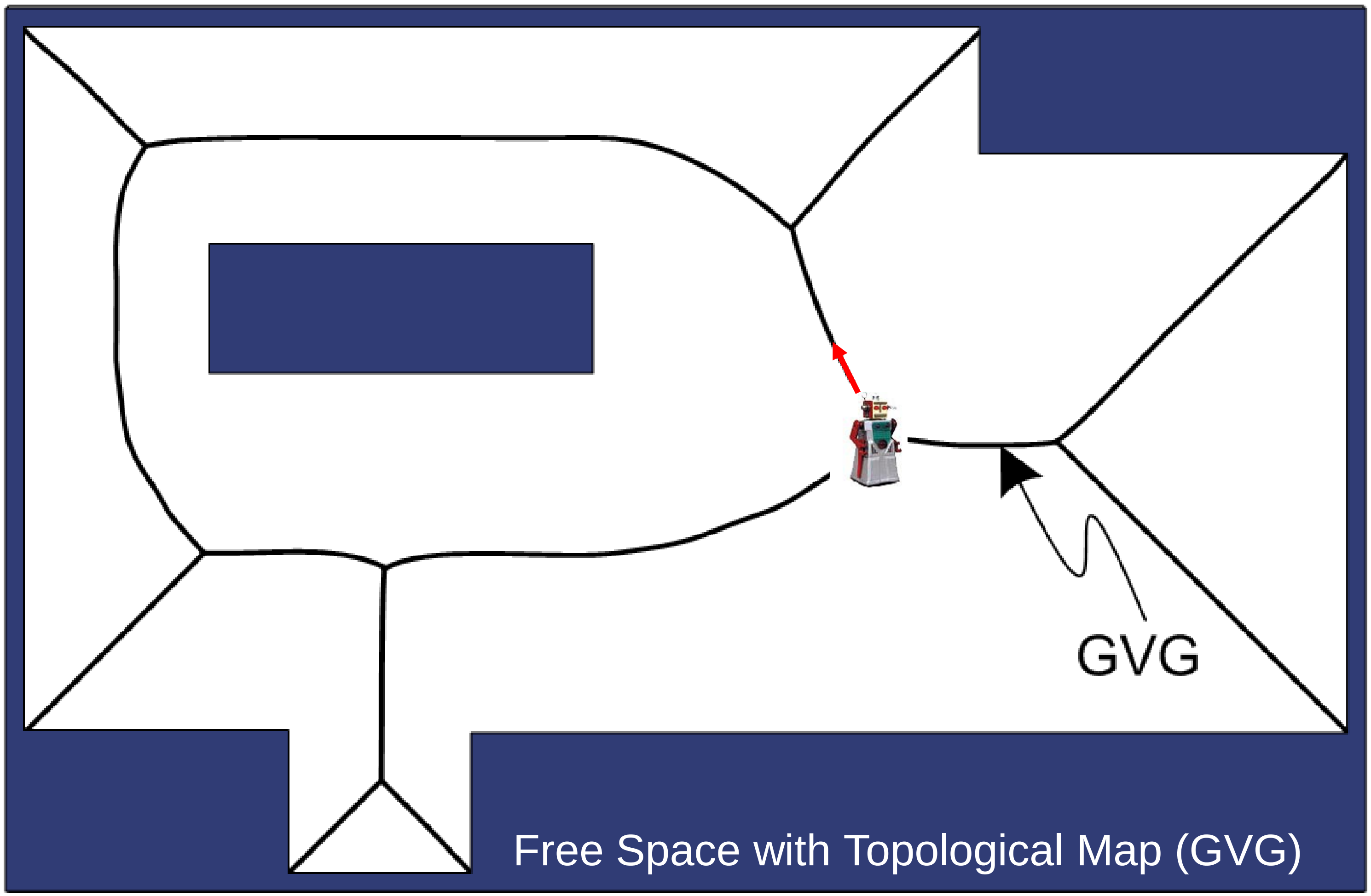
Generalized Voronoi Graph (GVG)

- Access GVG
- Home to the MeetPoint
- Follow Edge



Generalized Voronoi Graph (GVG)

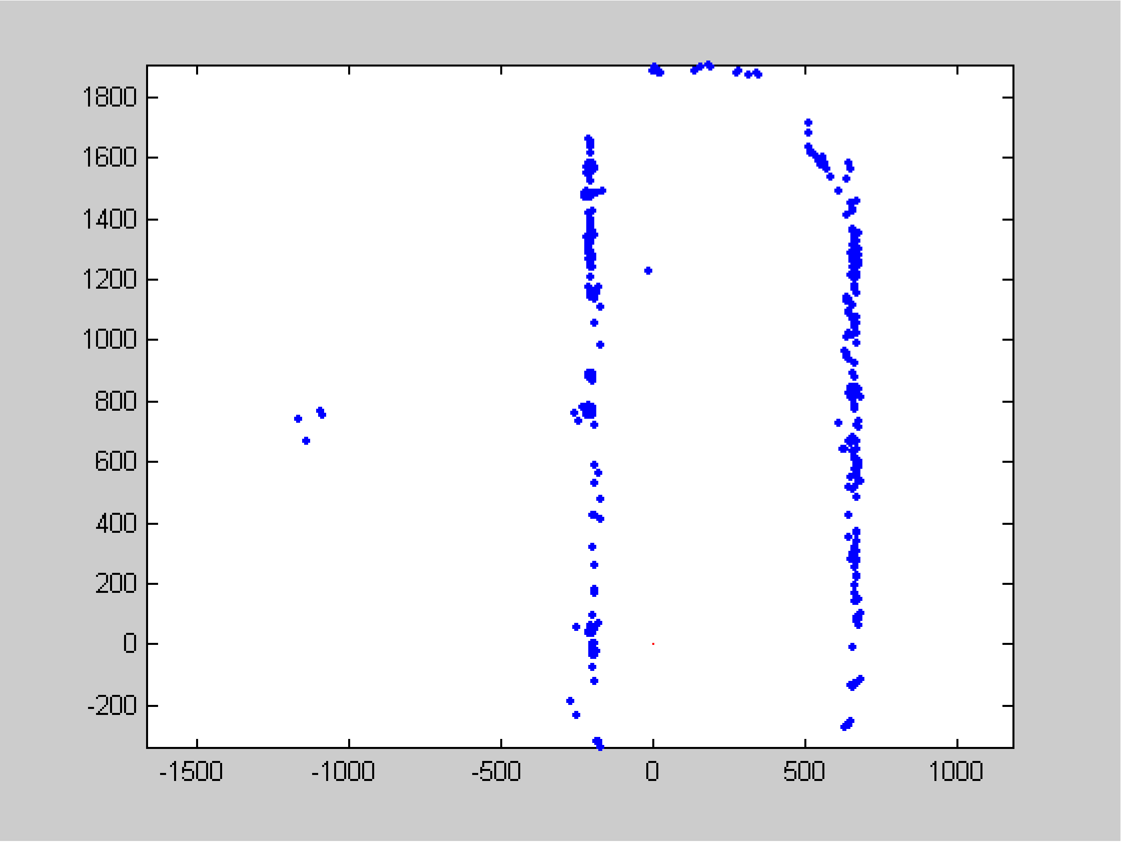
- Access GVG
- Follow Edge
- Home to the MeetPoint
- Select Edge



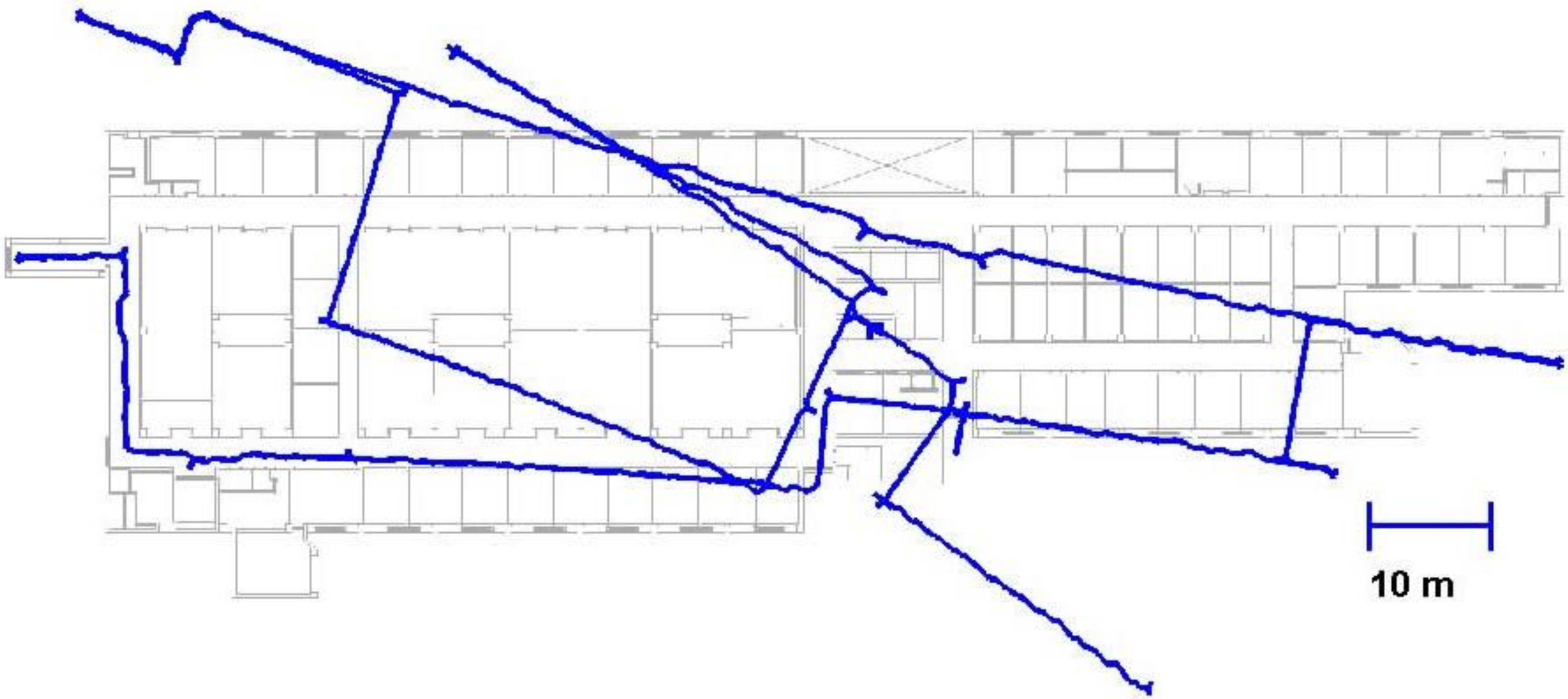
GVG construction using sonar

- Nomadic Scout
- Sonar (GVG navigation)
- Camera with omni-directional mirror (feature detection)
- Onboard 1.2 GHz processor

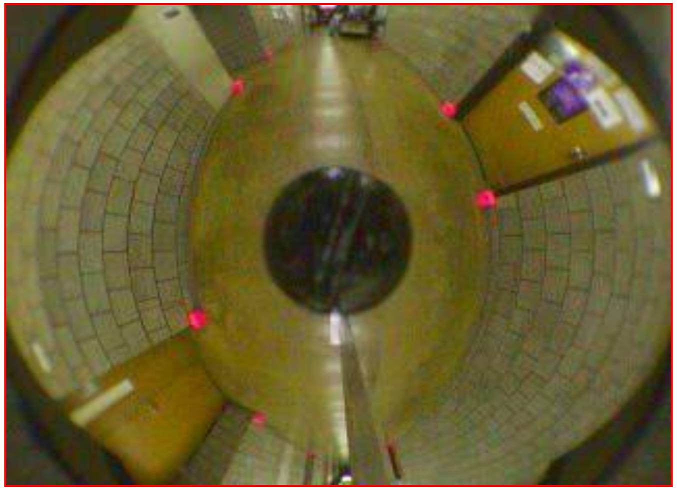
GVG construction using sonar



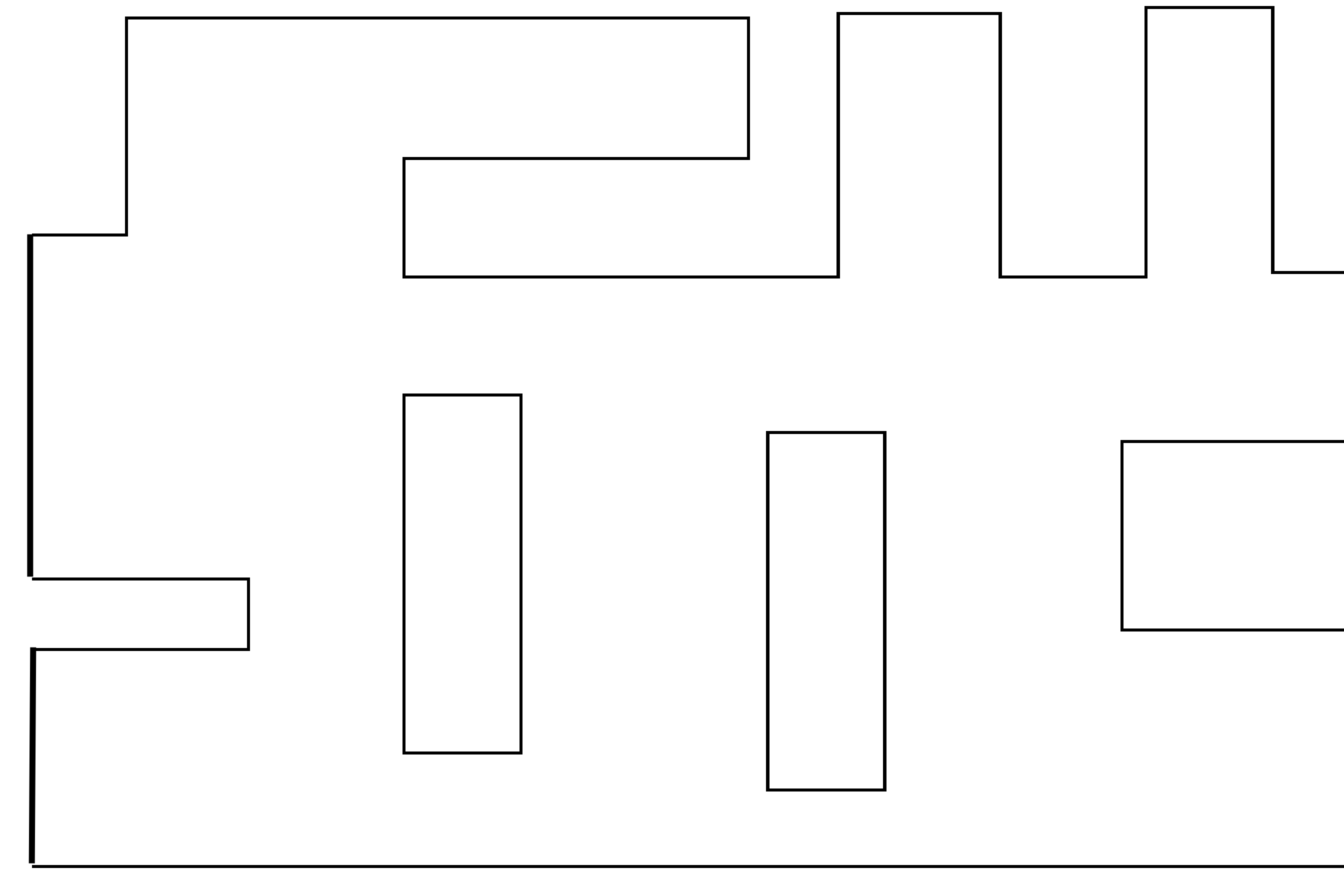
GVG construction using sonar



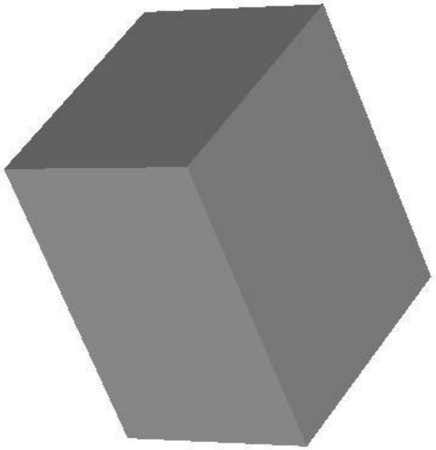
Slammer in Action

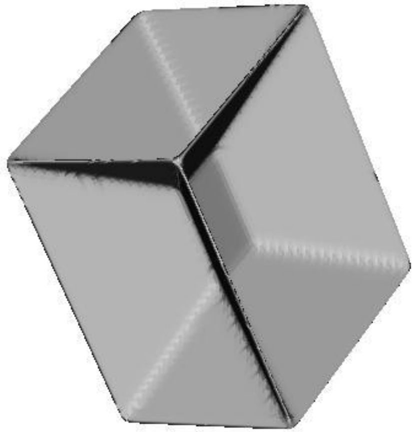


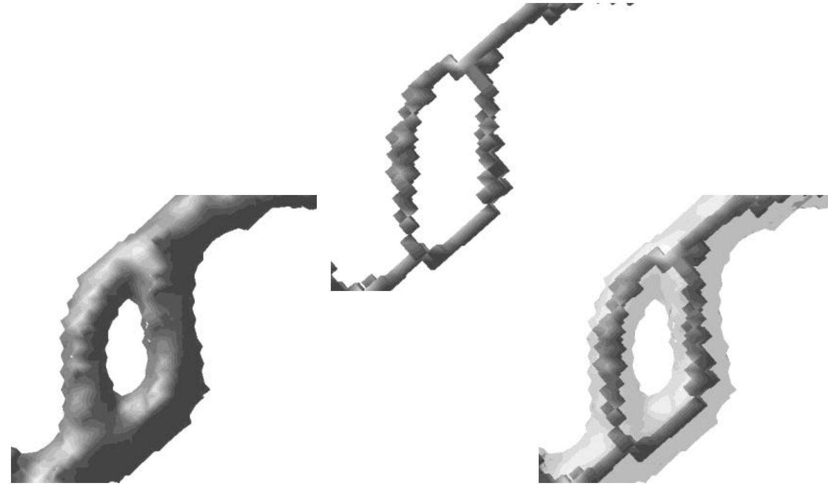
What is the Voronoi Graph?



Voronoi applications

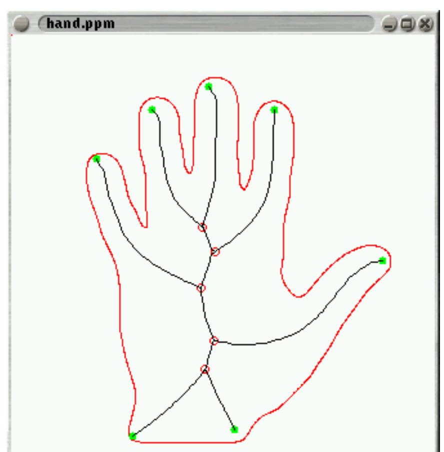






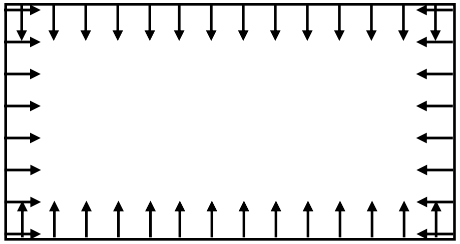
A retraction of a 3d object
== "*medial surface*"

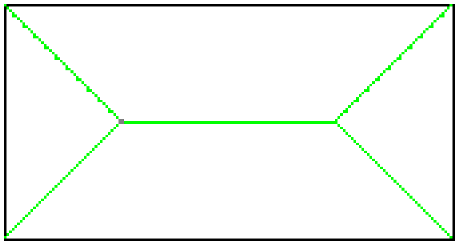
what? → Skeletonizations resulting from constant-speed curve evolution

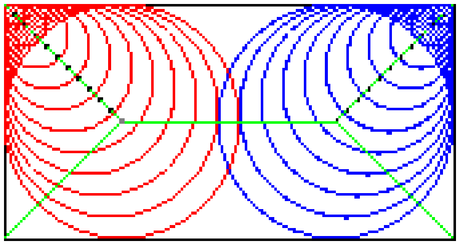


in 2d, it's called a *medial axis*

skeleton ⟷ shape

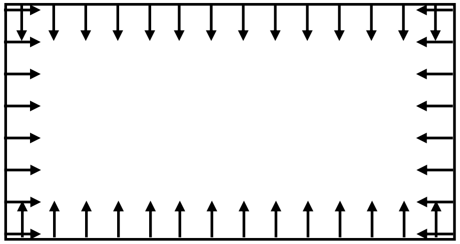

curve evolution


where wavefronts collide

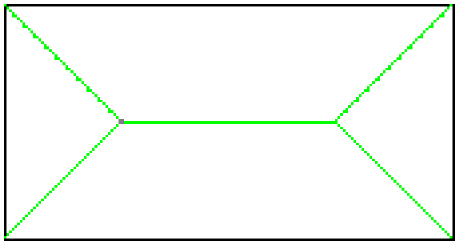

centers of maximal disks

again reduces a 2d (or higher) problem to a question about graphs...

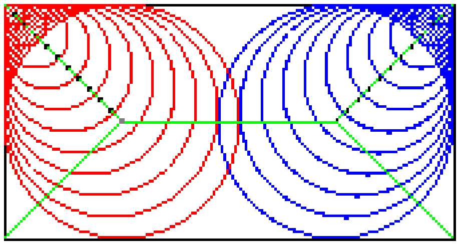
skeleton ⟷ shape



curve evolution

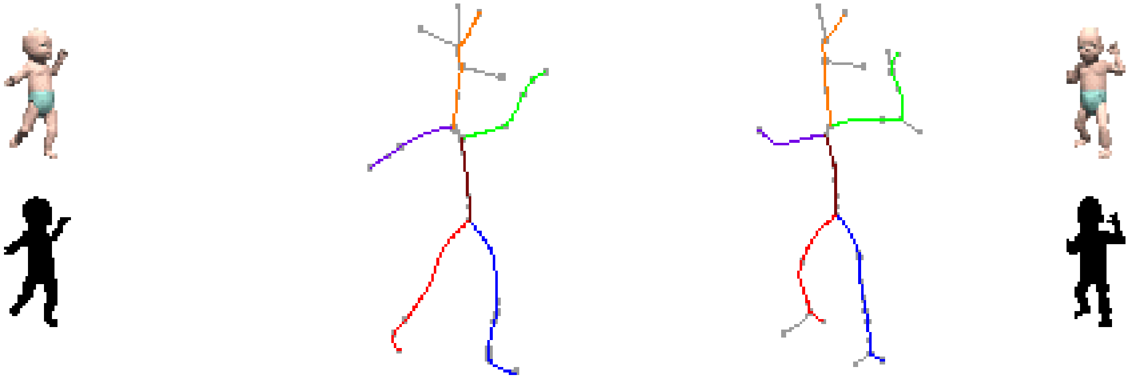


where wavefronts collide



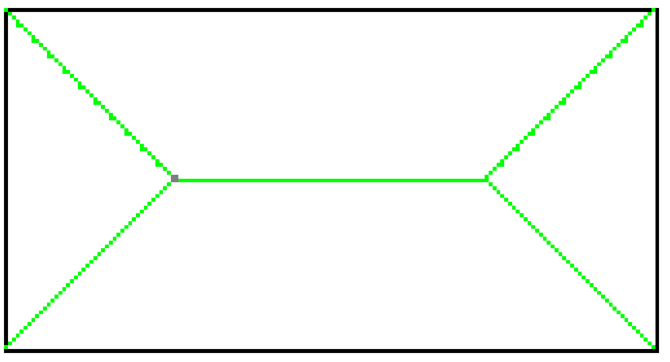
centers of maximal disks

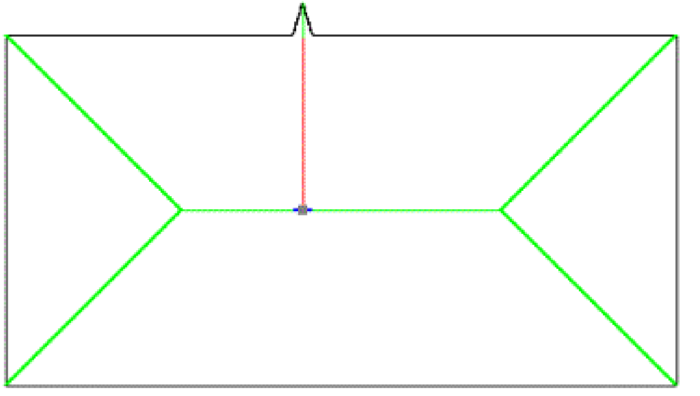
again reduces a 2d (or higher) problem to a question about graphs...



graph matching

Problems



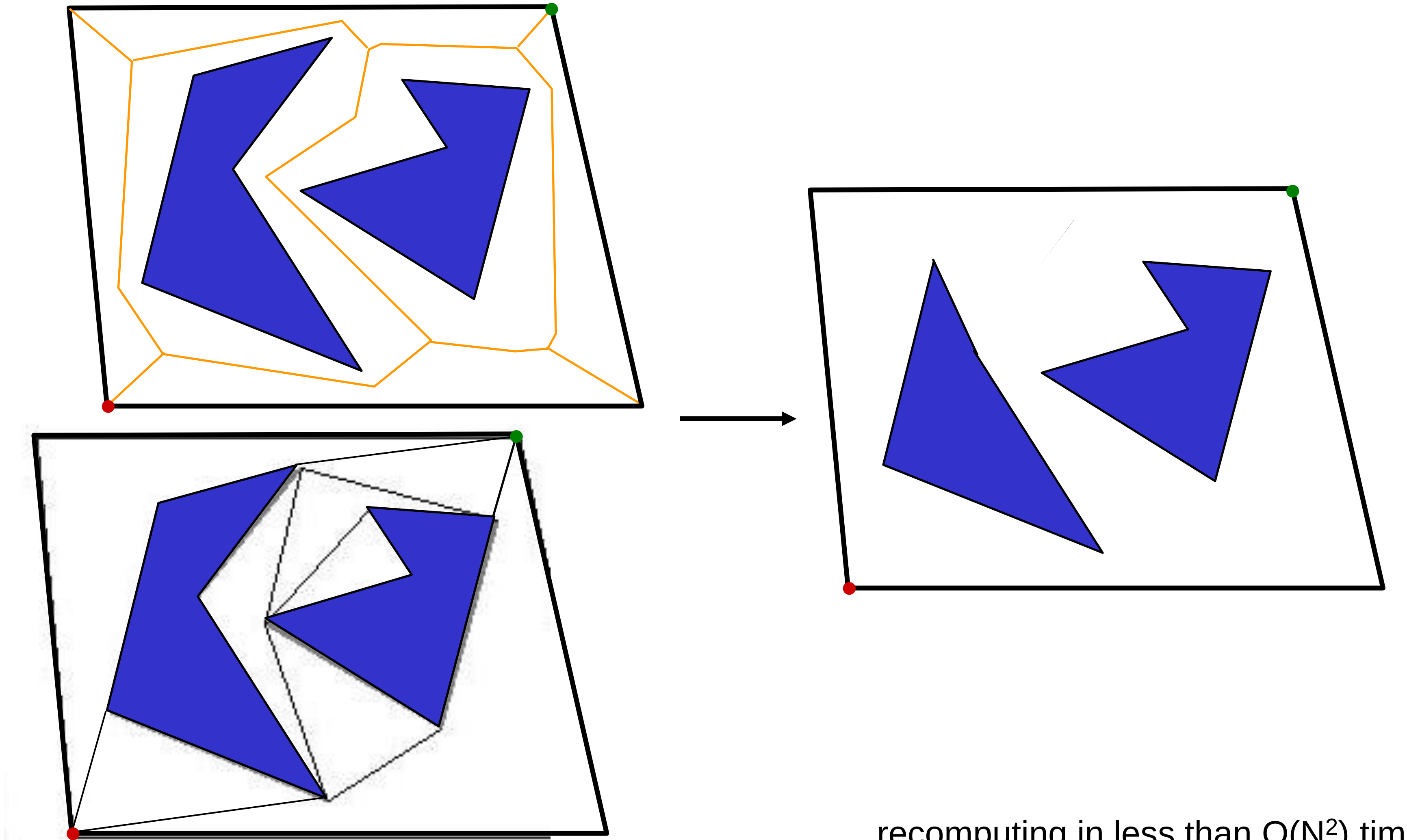


The skeleton is sensitive to small changes in the object's boundary.

- graph isomorphism (and lots of other graph questions) : NP-complete

Roadmap problems

If an obstacle decides to roll away... (or wasn't there to begin with)



recomputing in less than $O(N^2)$ time?

Experiments at CSA

- Autonomous Over-the-Horizon Navigation

CSA Experiments

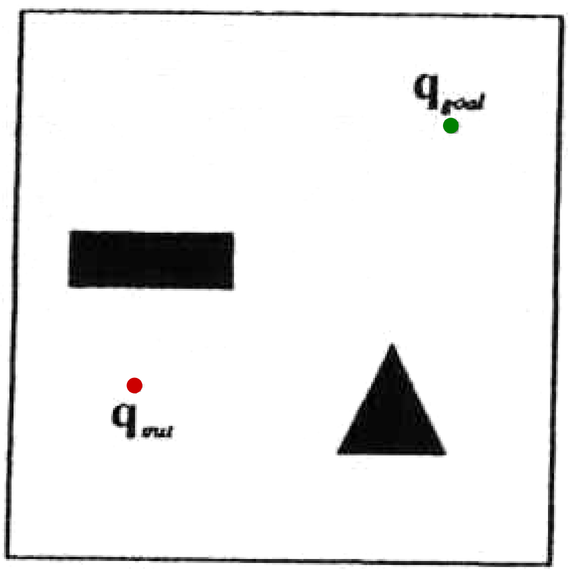
CSA Experiments

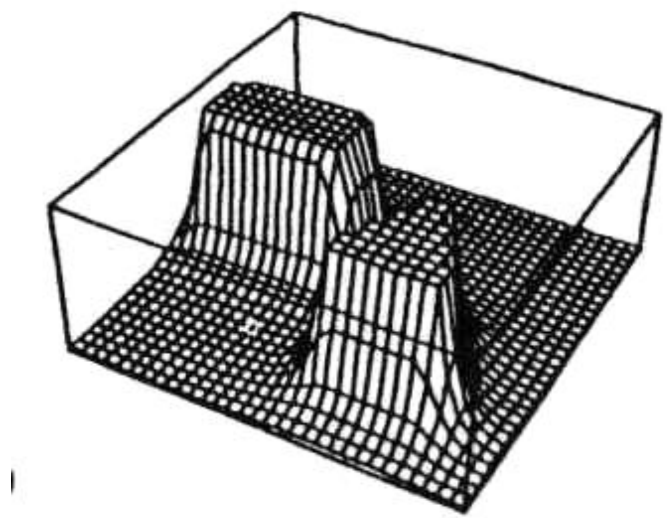
Practical Considerations

Path Planning

Potential Field methods

- compute a repulsive force away from obstacles

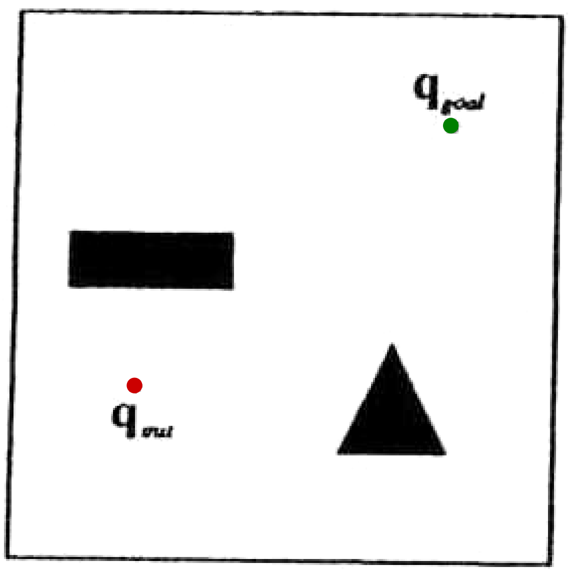


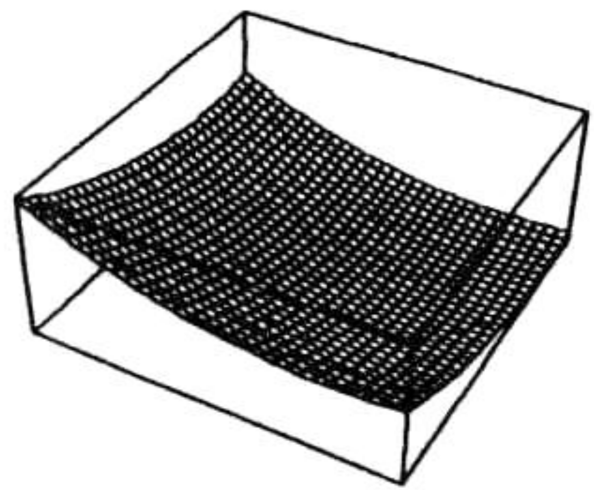


Local techniques

Potential Field methods

- compute a repulsive force away from obstacles
- compute an attractive force toward the goal



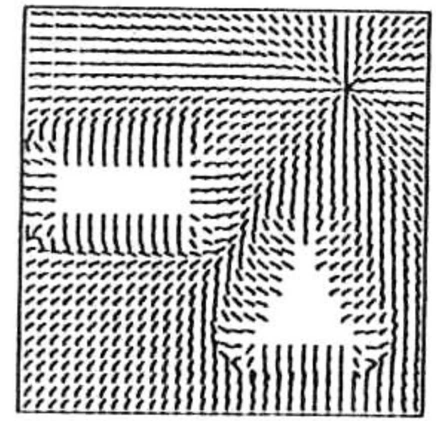


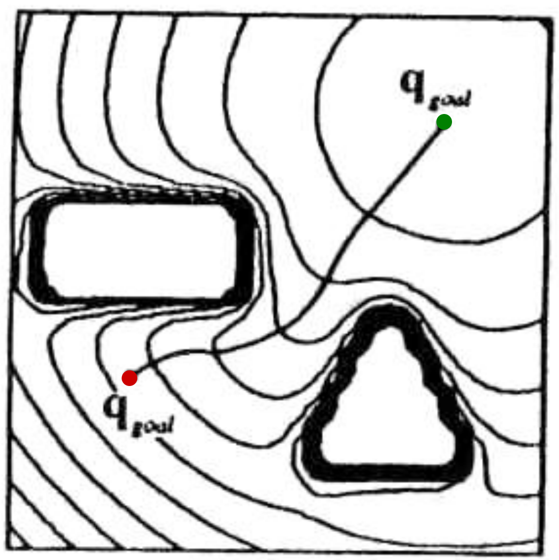
Local techniques

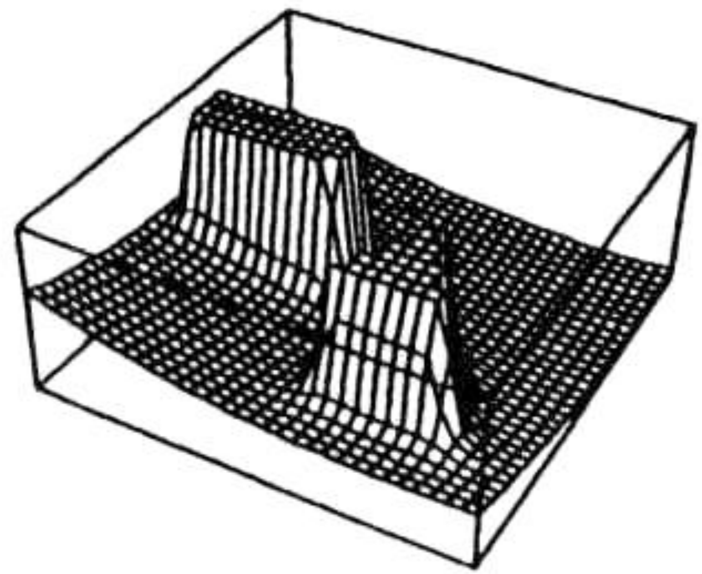
Potential Field methods

- compute a repulsive force away from obstacles
- compute an attractive force toward the goal

→ let the sum of the forces control the robot







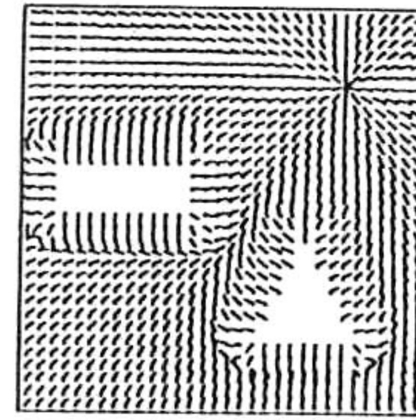
key advantages?

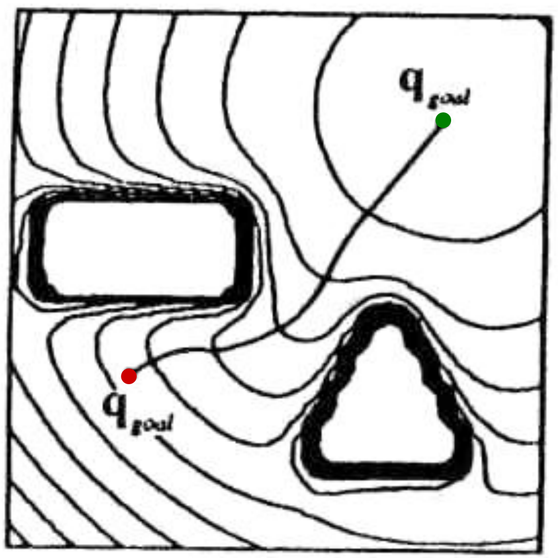
Local techniques

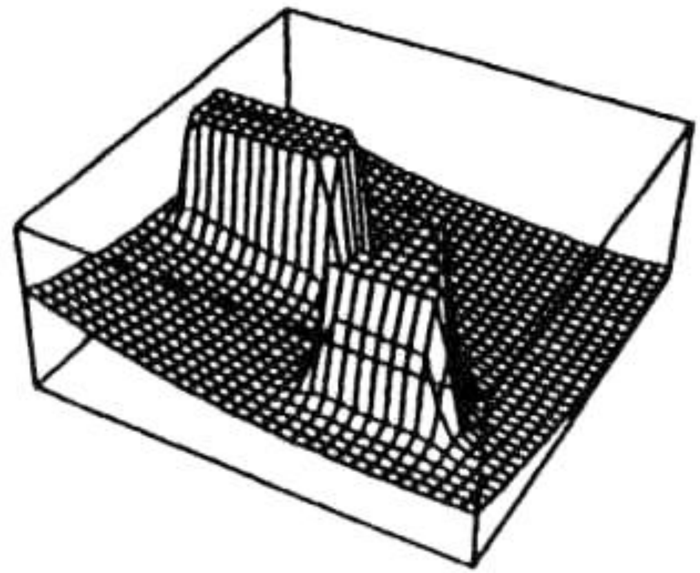
Potential Field methods

- compute a repulsive force away from obstacles
- compute an attractive force toward the goal

→ let the sum of the forces control the robot

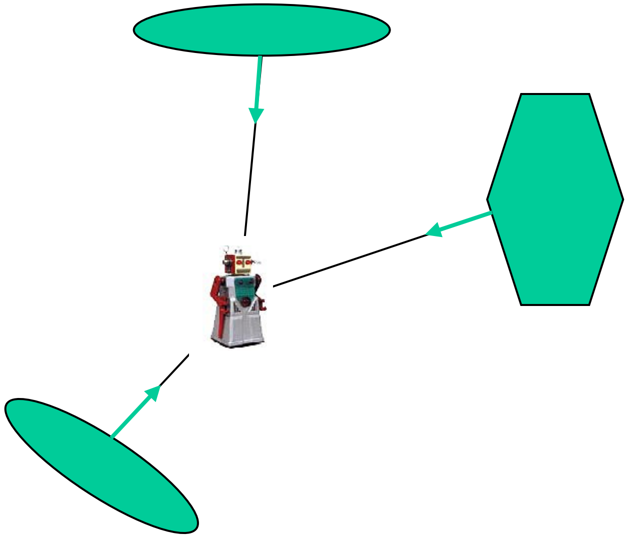






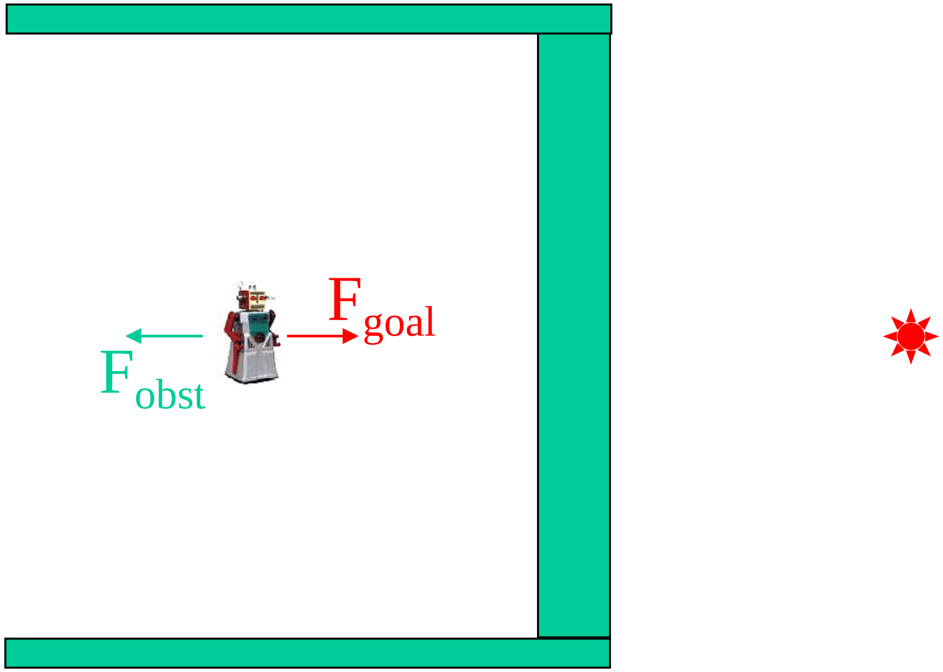
To a large extent, this is computable from sensor readings

Sensor Based Calculations



Major Problem?

Local Minima!

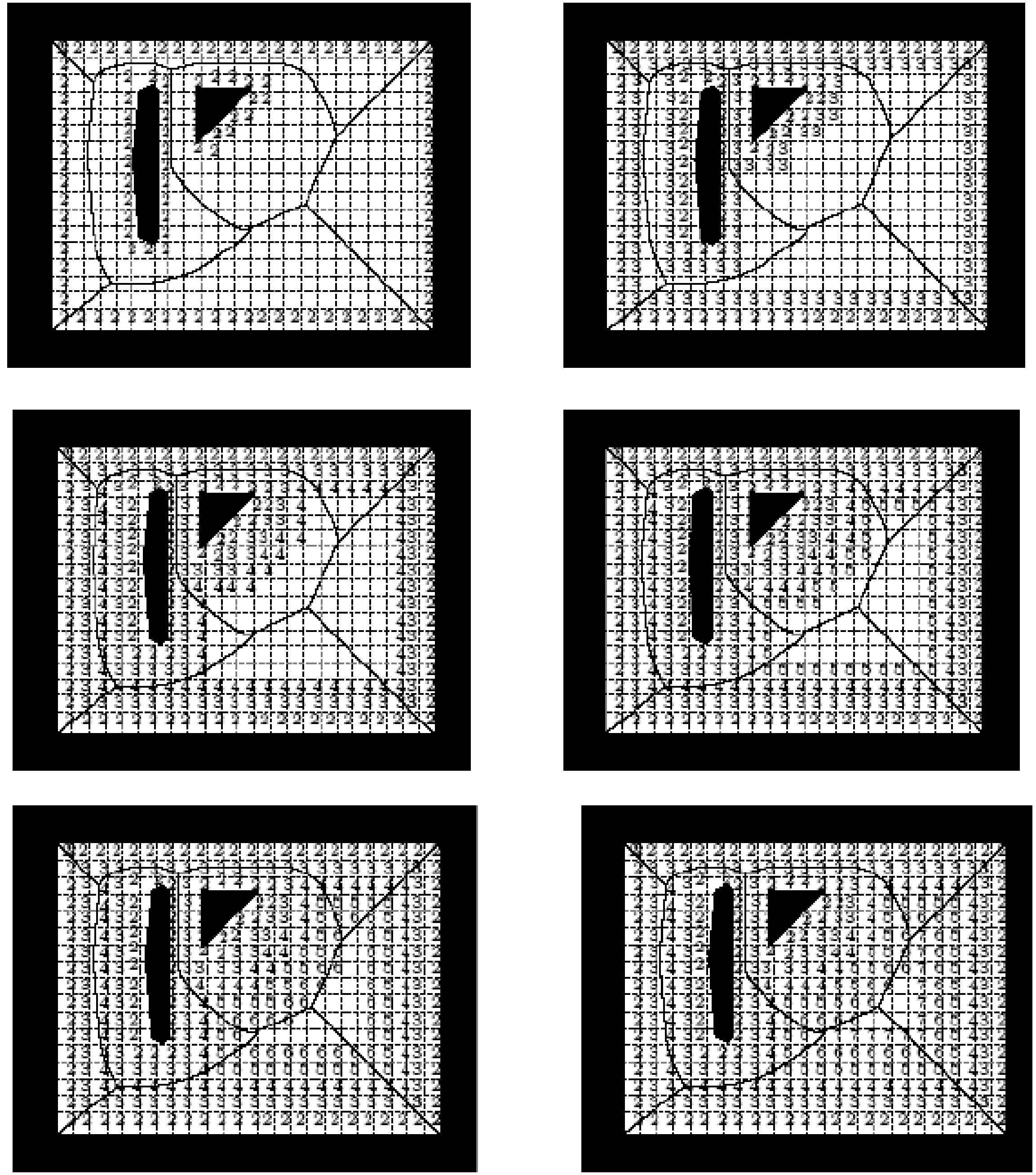


Simulated Annealing

- Every so often add some random force

Known Map

Brushfire Transform



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

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 3)

- Repeat

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

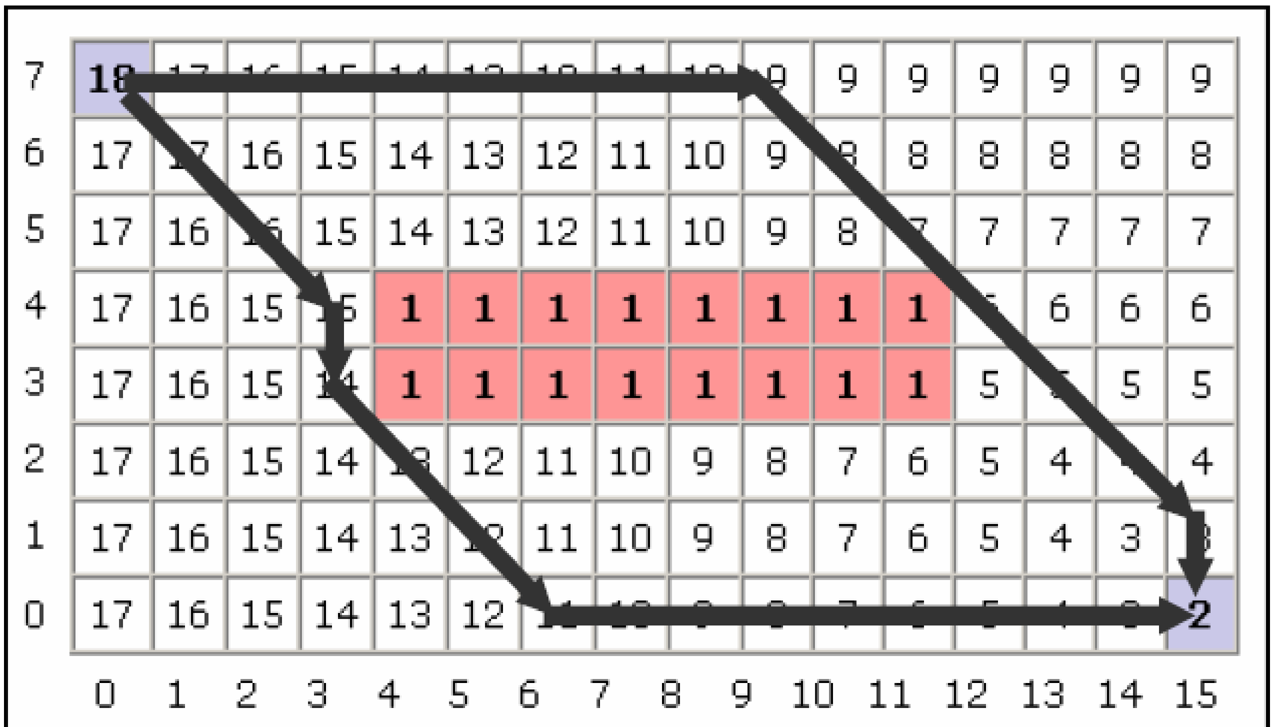
The Wavefront in Action (Part 3)

- Until Done
  - 0's would only remain in the unreachable areas

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 in Action

- 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

Introduction to Mapping

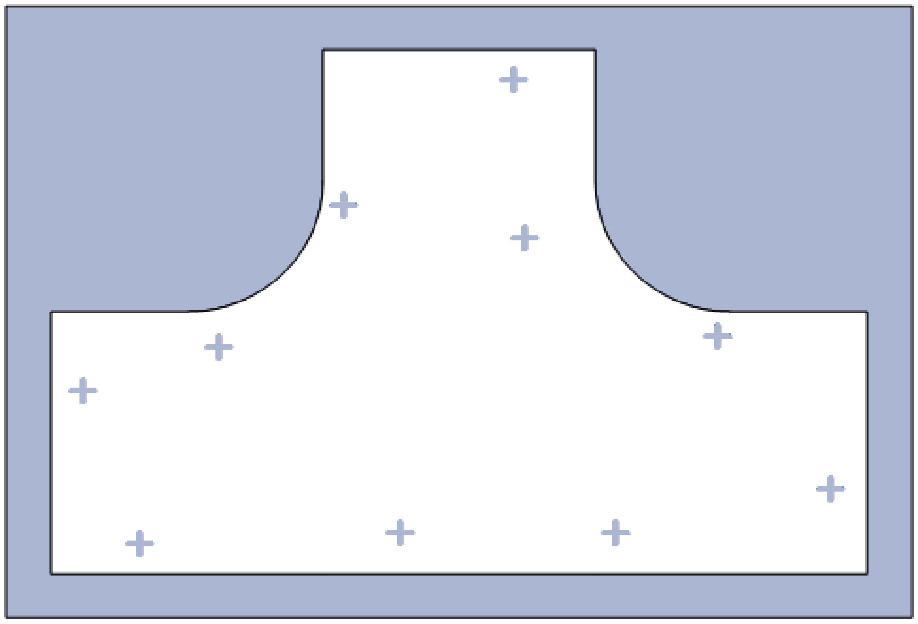
- What the world looks like?
- Knowledge representation
  - Robotics, AI, Vision
- Who is the end-user?
  - Human or Machine
- Ease of Path Planning
- **Uncertainty!**

Simultaneous Localization And Mapping

SLAM is the process of building a map of an environment while, at the same time, using that map to maintain the location of the robot.

- Problems for SLAM in large scale environments:
  - Controlling growth of uncertainty and complexity
  - Achieving autonomous exploration

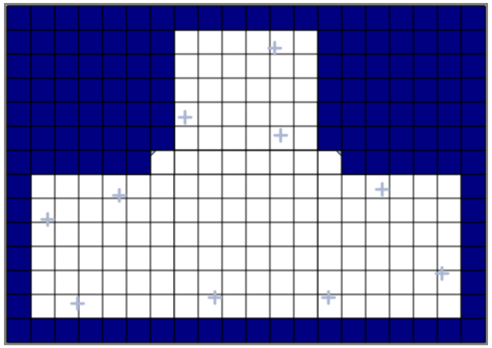
Consider this Environment:

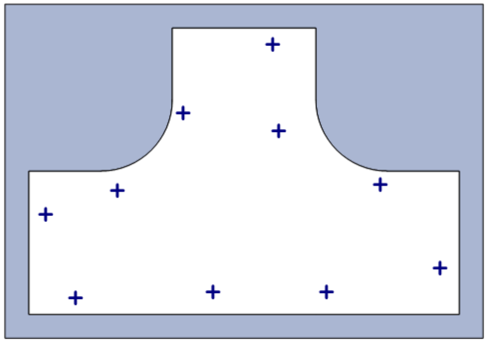


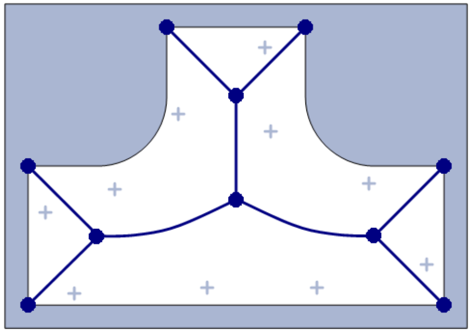
Three Basic Map Types

Topological:

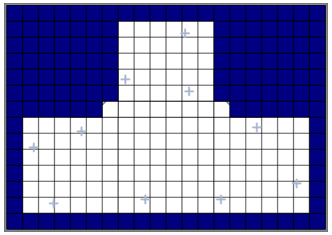
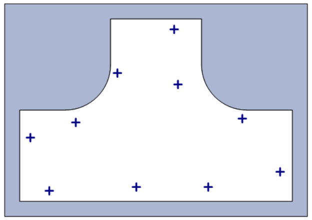
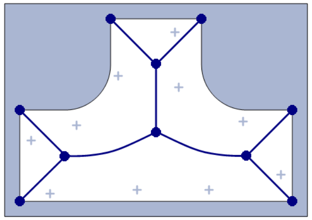
Collection of nodes and their interconnections







Three Basic Map Types

	Grid-Based	Feature-Based	Topological
Construction	Occupancy grids	Kalman Filter	Navigation control laws
Complexity	Grid size *and* resolution	Landmark covariance (N^3)	Minimal complexity
Obstacles	Discretized obstacles	Only structured obstacles	GVG defined by the safest path
Localization	Discrete localization	Arbitrary localization	Localize to nodes
Exploration	Frontier-based exploration	No inherent exploration	Graph exploration

Other Maps

	Appearance Based	Geometry Based	Mesh Based
Construction	Images	Lines, planes, etc	Mesh
Path Planning	N/A	Geometry based	Graph based
Localization	Arbitrary localization	Arbitrary localization	Arbitrary localization