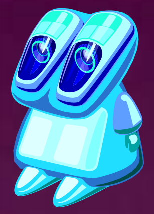
Lecture 8-1

Going Places:

Mapping and Path Planning

The Robotics Primer (Ch. 19)

C.A. Berry



Course Announcements

- Quiz on **Tuesday, 5/05/09** on **Mapping and Path Planning**
- Lab 6 Demo due **Thursday, 5/07/09**
- Lab 6 Memo and code due by midnight on **Friday, 5/08/09**
- Project Demo 1 due **Thursday, 5/14/09**
- Project Demo 2 due **Monday, 5/18/09**
- Competition, **Tuesday, 5/19/09**
- Course wrap up, **Thursday, 5/21/09**
- Report and Code due Friday, **5/22/09**

Quote of the Week

"Making realistic robots is going to polarize the market, if you will. You will have some people who love it and some people who will really be disturbed."

David Hanson, CNN.com, 11/23/06

Map Building

Map Building

Techniques:

- Manual
  - Drawn by hand
  - Static/predictable environment
  - Costly
- Automatically
  - Robot learns environment
  - Dynamically/unpredictable changing
  - Different look due to different perception

Requirements:

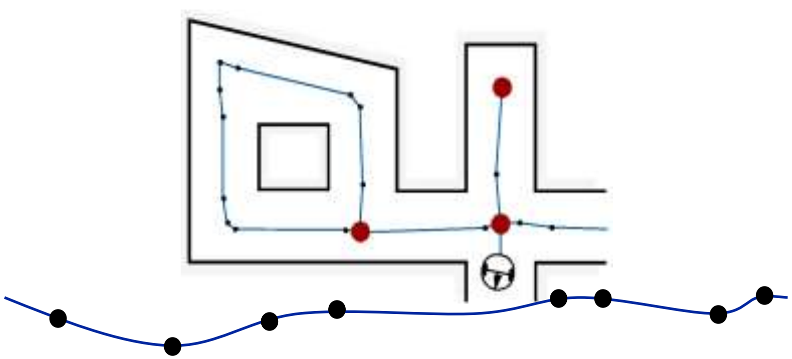
- Incorporates newly sensed information into the existing world model
- Contains information to estimate the robot's position
- Provides Information to do path planning and navigation tasks

Map Building: Measure of Quality

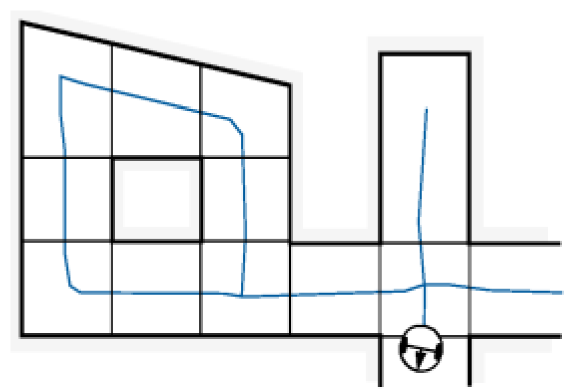
- Most environments are a mixture of *predictable* and *unpredictable* features (hybrid approach)
- The measure of quality is based upon
  - Topological correctness
  - Metric correctness

Road Map, Graph Construction
Cell Decomposition

- *Road Map, Graph construction*
  - Identify a set of routes within the free space
  - Where to put the nodes?
  - Topology-based:
    - at distinctive locations
  - Metric-based:
    - where features disappear or get visible

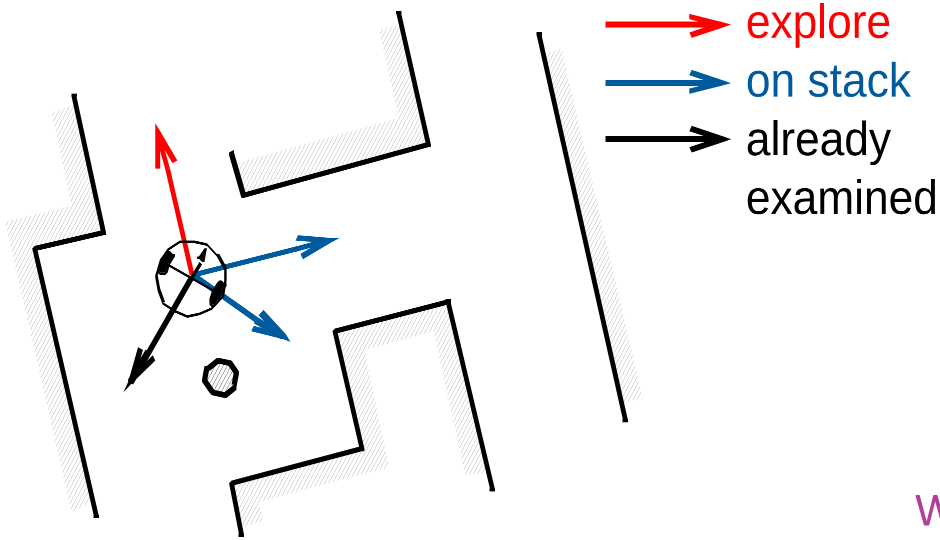


- *Cell decomposition*
  - Discriminate between free and occupied cells
  - Where to put the cell boundaries?
  - Topology- and metric-based:
    - where features disappear or get visible



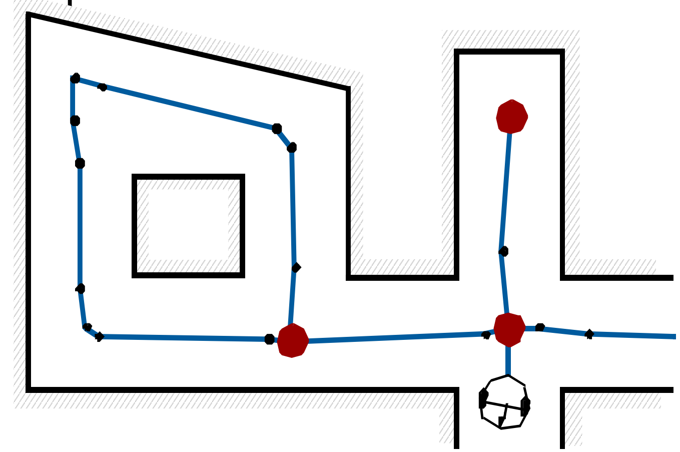
Map Building:
Exploration and Graph Construction

1. Exploration



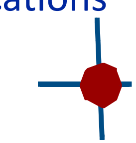
- provides correct topology
- must recognize already visited location
- backtracking for unexplored openings

2. Graph Construction



Where to put the nodes?

- Topology-based: at distinctive locations



- Metric-based: where features disappear or get visible

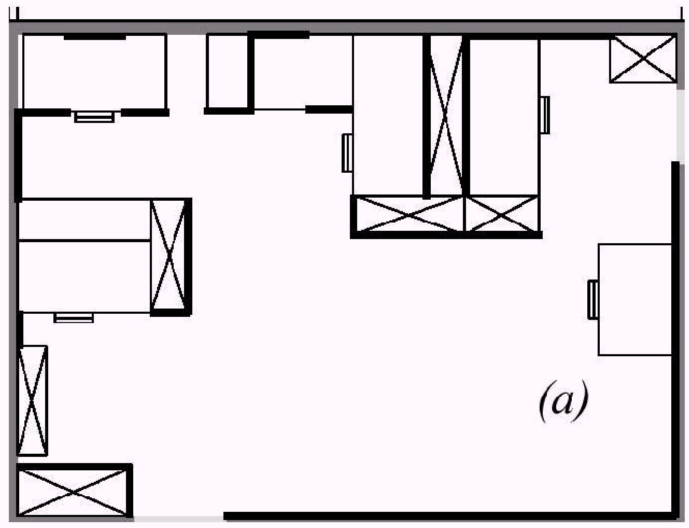
Continuous representation

- A continuous-valued map is one method for exact *decomposition* of the environment
- Continuous maps are only in 2D representations as further dimensionality can result in computational explosion
- Combine the exactness of continuous representation with the compactness of *closed-world assumption*
- The representation will specify all environmental objects in the map

Continuous representation

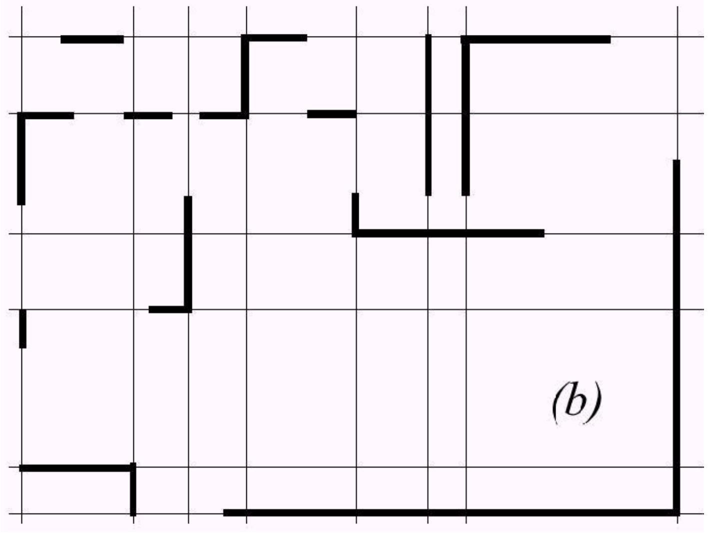
- a low-memory map is a 2D representation in which polygons represent all obstacles
- many simulations run exclusively in the computer memory and polygons are not used to describe a real-world environment
- When real environments must be captured, there are trends for *selectivity* and *abstraction*
  - *The human captures only objects that can be detected by the robot's sensors*
  - *This represents a subset of the features of the real world objects*

Continuous Representation



Architecture map

Infinite line representation

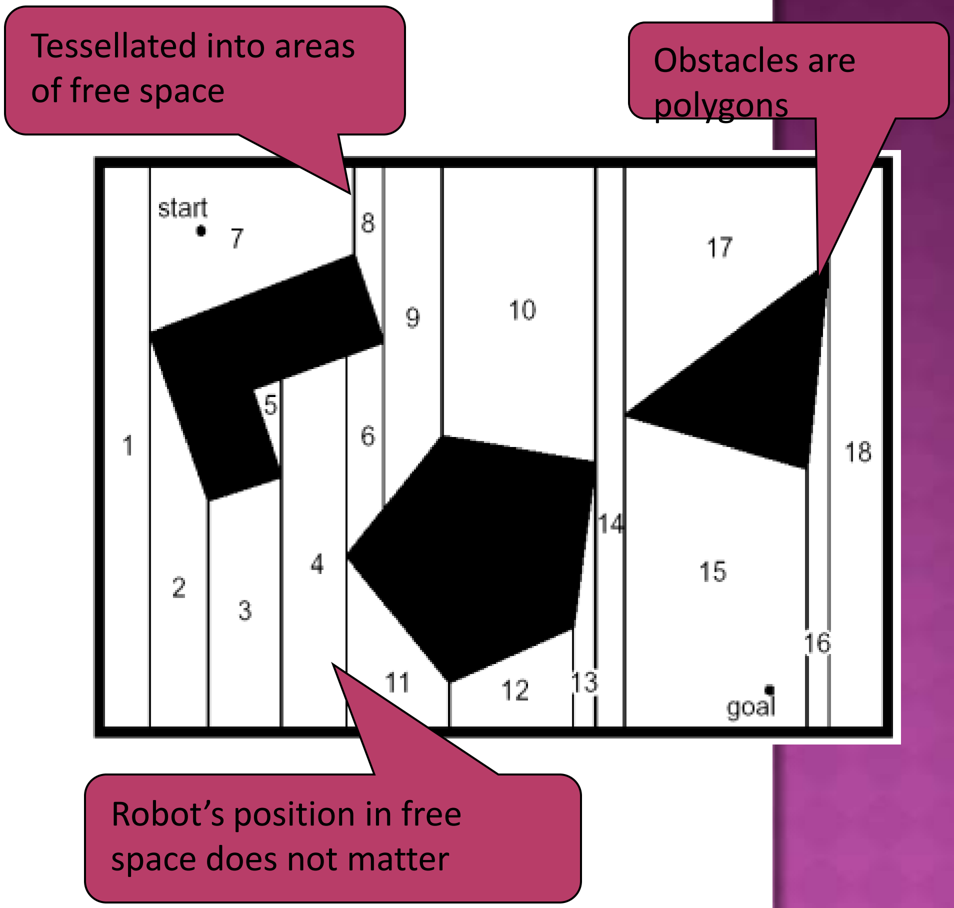


Decomposition strategies

- One method of *simplification* is to approximate the real world environment lines as a set of infinite lines
- A more dramatic form of simplification is *abstraction*
  - *A general decomposition and selection of environmental features*
- The immediate disadvantage is the loss of fidelity between the map and the real world
- It may be useful if planned carefully to capture relevant, useful features of the world while discarding all other features

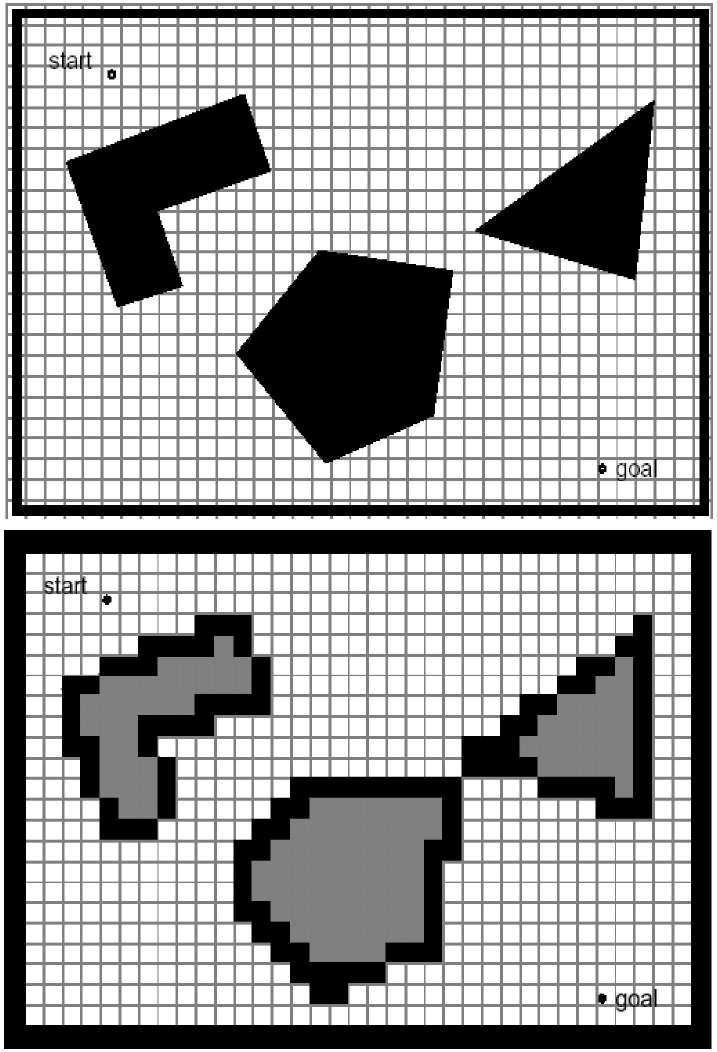
Tessellation Decomposition strategy

- Advantage:
  - the map representation is minimized
  - With hierarchical decomposition, reasoning and planning may be computationally superior to a fully detailed world model
  - A standard, lossless form of *opportunistic decomposition* is termed *exact cell decomposition* selects boundaries between discrete cells based on geometric criticality



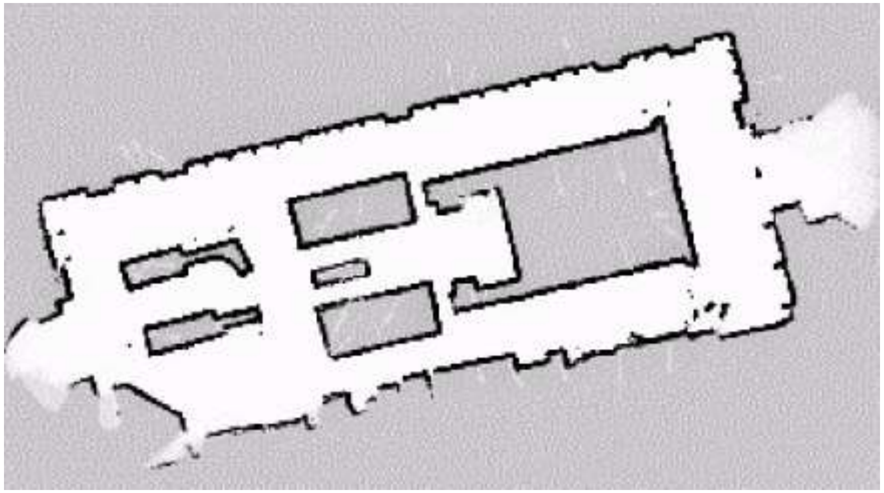
Fixed Cell Decomposition

- *In fixed cell decomposition*, the world is tessellated into a discrete approximation of the continuous map
- The key disadvantage is the inexact nature
- Narrow passages are lost in this transformation



Occupancy Grid Map Representation

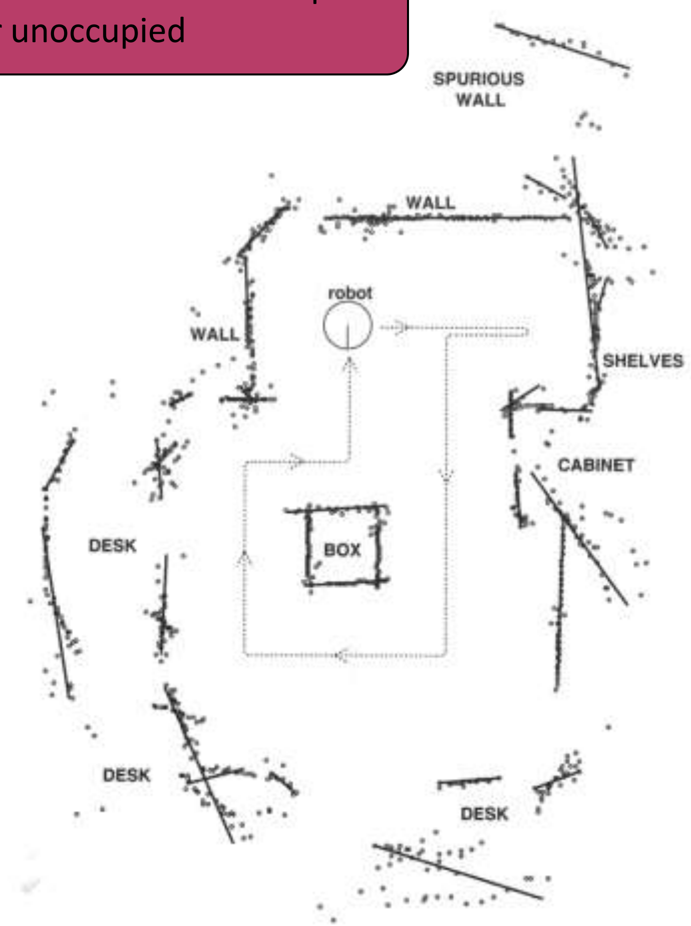
- A counter is used to determine how many times a cell is hit by a ranging sensor
- As the counter is incremented, the cell is deemed an obstacle
- The darkness of the cell is proportional to the value of the counter



Occupancy grid map representation Disadvantages

- The size of the map in robot memory grows with the environment size
- Small cell sizes make the size of the memory untenable
- Not compatible with the closed-world assumption which enables large, sparse environments to have small memory requirements
- Imposes a geometric grid on the world a priori, regardless of environment details

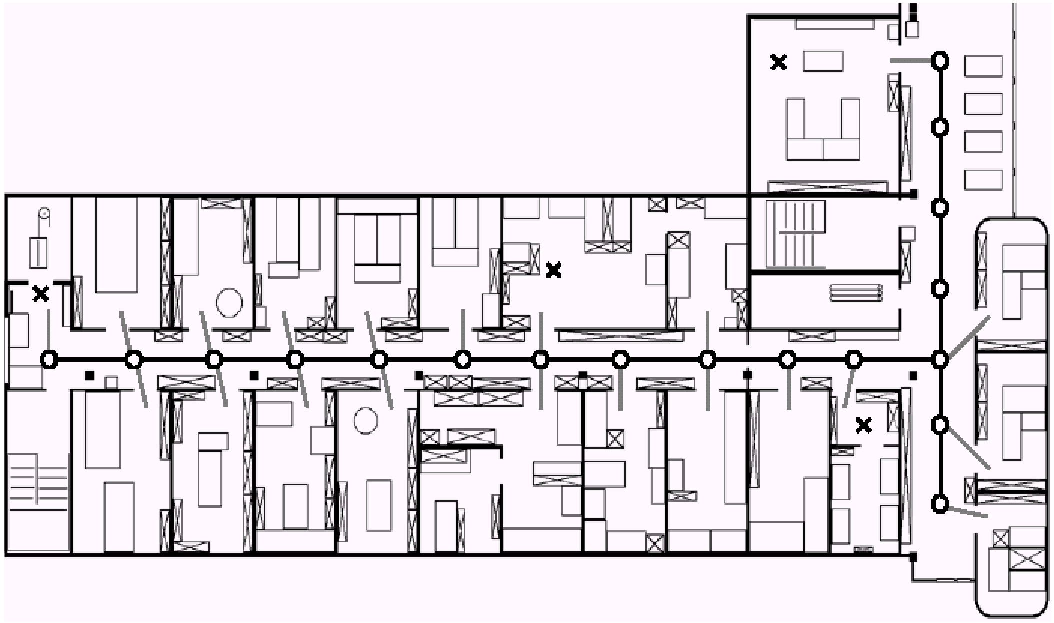
Each cell is either occupied or unoccupied



Made with sonar data

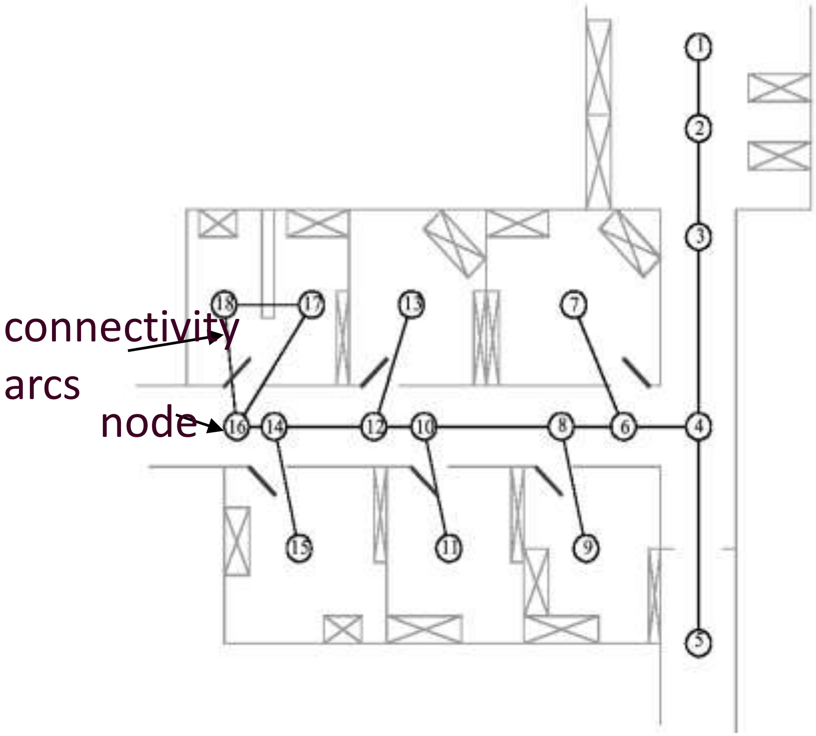
Topological Decomposition

- Avoids direct measurement of geometric environmental qualities
- Concentrates on characteristics that are most relevant to robot localization



Topological Decomposition

- Topological representations is a graph that specifies
  - *Nodes*
    - Areas in the world
  - *Connectivity arcs*
    - Denotes adjacent pairs of nodes
  - Adjacency is at the heart of the topological approach
  - Nodes are not of a fixed size or specifications of free space
  - Nodes document an area based on ay sensor discriminant



Model Complexity

- Some models are very elaborate
  - They take a long time to construct
  - These are kept around for a long time throughout the lifetime of the robot
  - E.g.: a detailed metric map
- Other models are simple
  - Can be quickly constructed
  - In general they are transient and can be discarded after use
  - e.g.: information related to the immediate goals of the robot (avoiding an obstacle, opening of a door, etc.)

Models and Computation

- Using models require significant amount of computation
- *Construction*
  - the more complex the model, the more computation is needed to construct the model
- *Maintenance*
  - models need to be updated and kept up-to-date, or they become useless
- *Use of representations*
  - complexity directly affects the type and amount of computation required for using the model
- Different architectures have different ways of handling representations

Data Association Problem

- SLAM is a difficult problem because it involves having the robot perform two ongoing and related parallel processes
- There is confusion among multiple places that look similar and therefore ambiguous
- This is the *data association problem* of uniquely associating the sensed data with absolute ground truth
- For topological maps
  - the robot has to contend with uniquely identifying landmarks
- For metric maps
  - The robot has to contend with odometry error and other sensor measurements
- It would be nicer if the robot had the map in order to localize or if the robot can localize when building a map

Coverage

- There are two coverage problems based upon with and without a map
- If there is a map
  - The robot searches all navigable spaces until the goal point is found
  - This is a search problem and there are algorithms in computer science and AI that are rather slow
  - Some search algorithms use *heuristics*, rules of thumb that help guide and speed up a search

No map for coverage

- When the robot does not have a map it has to move in a systematic fashion to find what it is looking for
- Mapping the environment first may be a better approach but that takes time
- One heuristic when the map is not known is to follow continuous boundaries or spiral out from a starting point
- The robot may also move randomly and given enough time and a closed environment it may cover the space

Search and Path Planning (Ch. 19)

Search and Path Planning

- There are many possible paths between the start and the goal point for a robot
- The robot finds all of them by searching the map
- To make this efficient, the map is turned into a *graph*, a set of nodes and the lines that connect them
- A path planner looks for the optimal path based upon some criterion (i.e. distance, safest)
- Path planning requires robots to perform higher-level thinking or reasoning

Competencies for Navigation

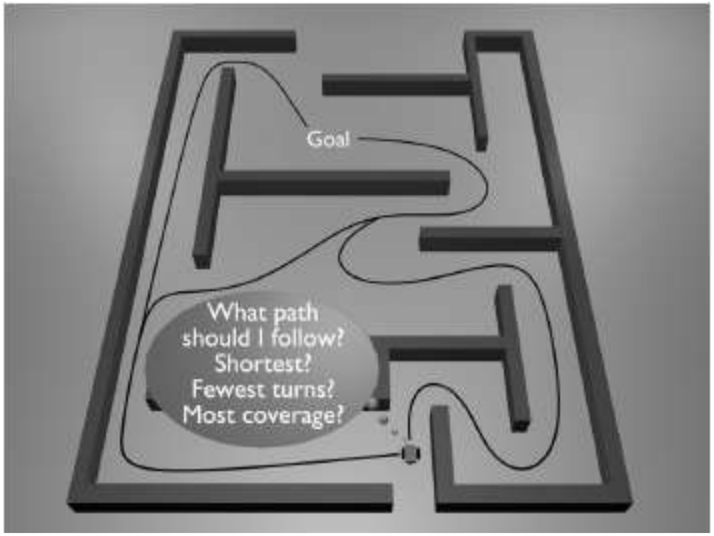
- The robot must incorporate new information gained during plan execution. The planner must incorporate this new information as it is received in order to correct a planned trajectory
- When a planner incorporates every new piece of information in real time, instantly produces a new plan and reacts this is called *integrated planning and execution*
- Robot control can usually be decomposed into global and local behaviors or rules
  - wall following (local)
  - find objects (global)
  - path planning (global)
  - obstacle avoidance (local)

Global Path Planning

- The robot's environment representation can range from a continuous geometric description to a decomposition-based geometric map or a topological map
- Assumption: there exists a good enough map of the environment for navigation.
- Three general strategies for decomposition
  - *road map* - identify a set of routes within the free space
  - *cell decomposition* – discriminate between free and occupied cells
  - *potential field* – impose a mathematical function over the space

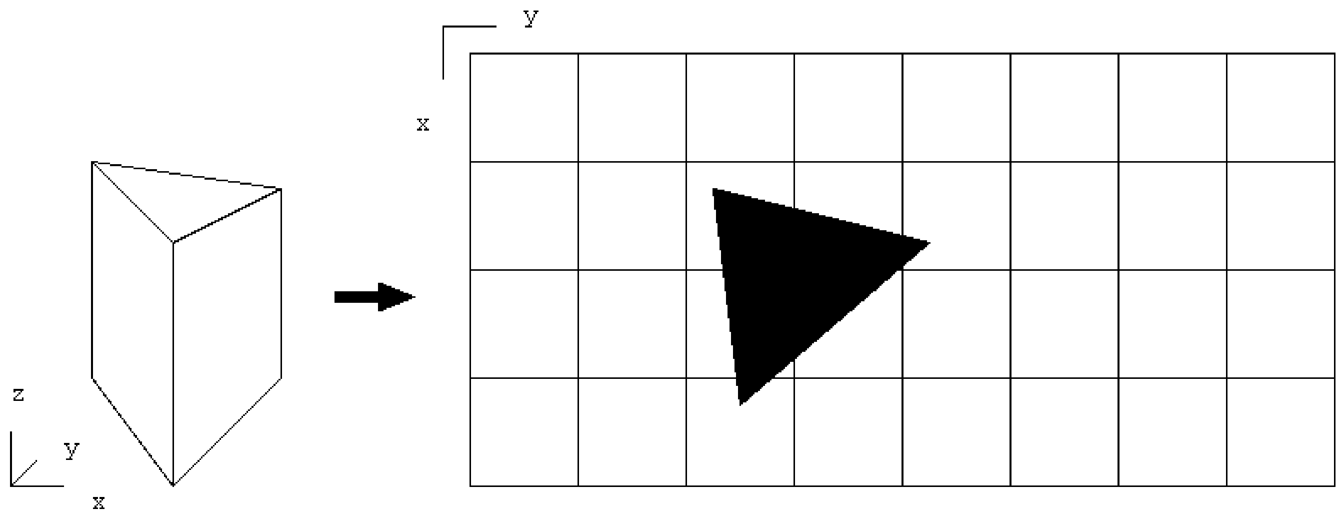
Planner options

- Some planners do not look for optimal paths but use a local map to plan a path and speed up the process
- Other planners look for the first path that gets the robot to the goal
- It requires a great deal of work to represent the environment, plan a path and convert the path to a set of movement commands to the robot



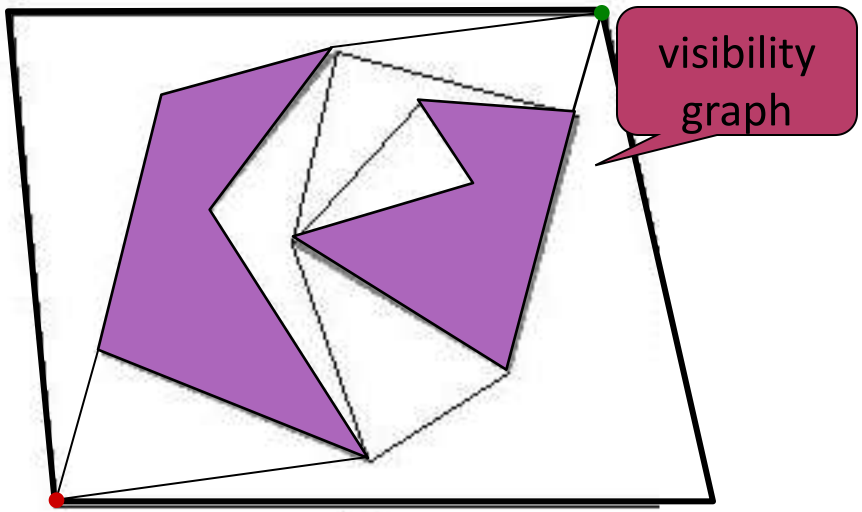
Configuration Space

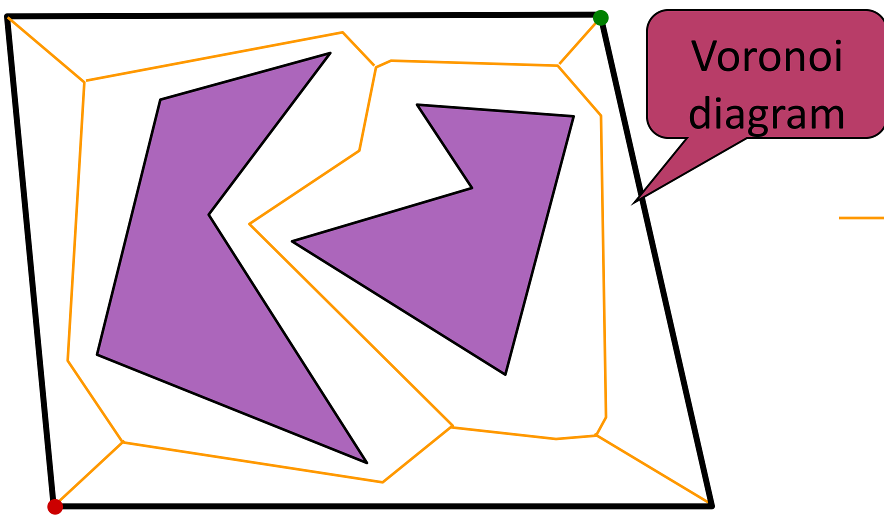
- Metric Maps use *Configuration Space (Cspace)*
- *Cspace* transforms three dimensional space to 2 dimensional space suitable for robots, this is a simplifying assumption
- This is more amenable for storage in computer and for rapid execution of algorithms



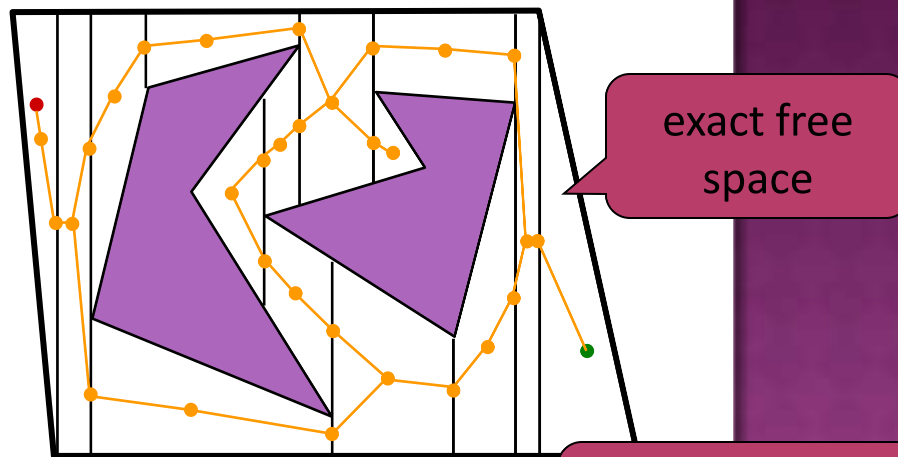
Full-knowledge motion planning

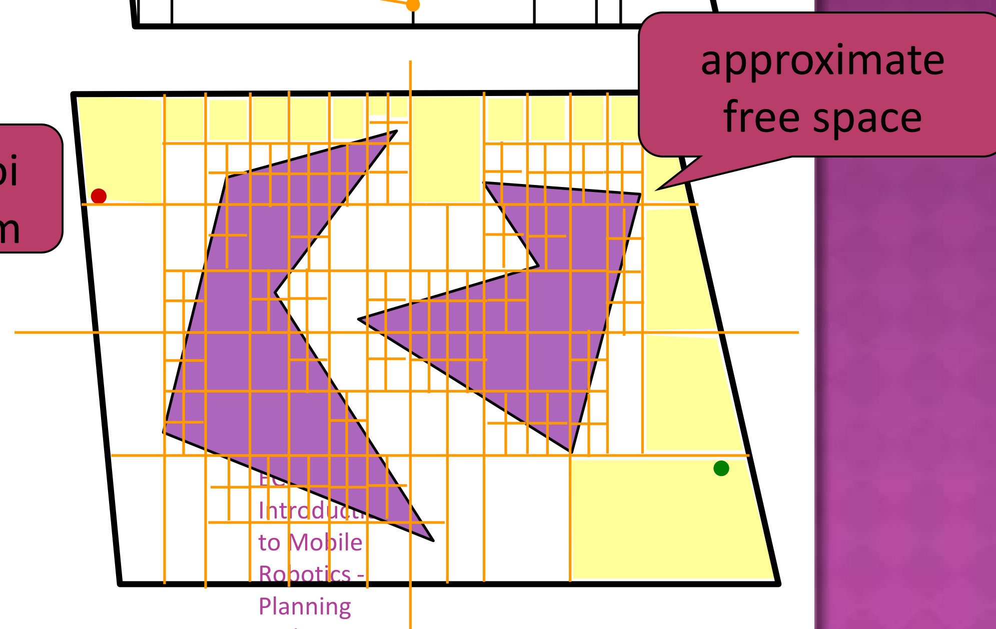
Roadmaps



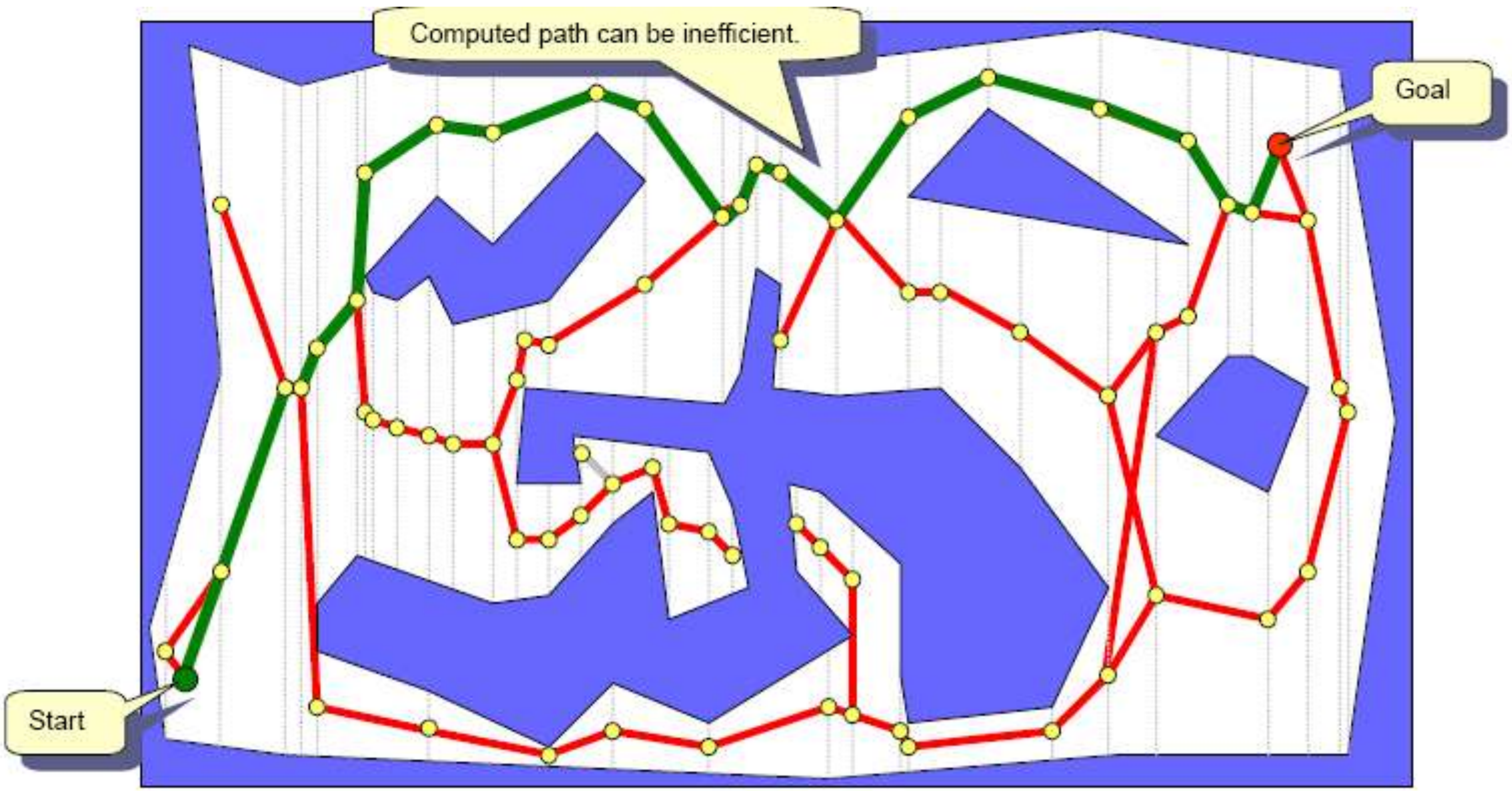


Cell decompositions

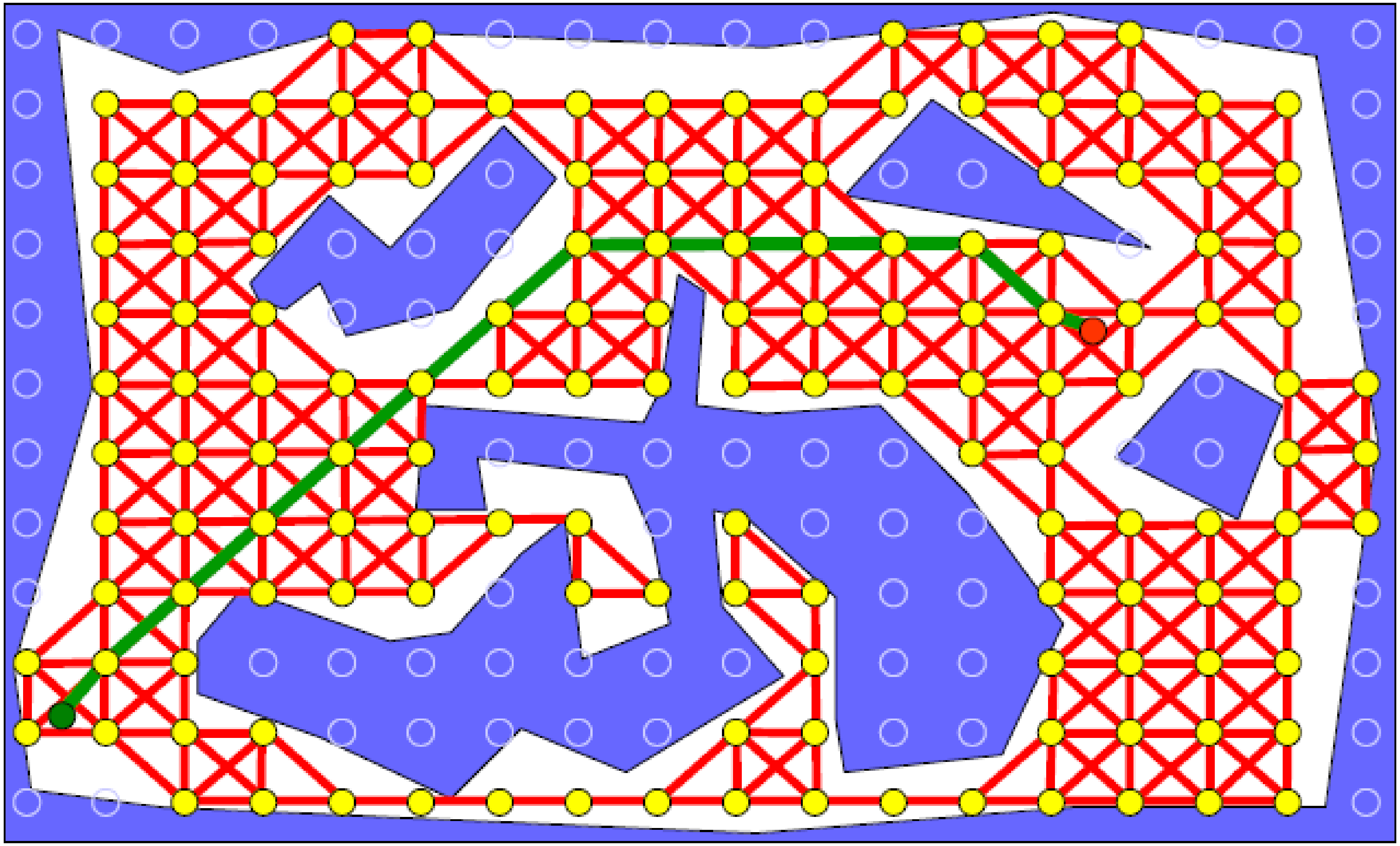




Road map-based path planning

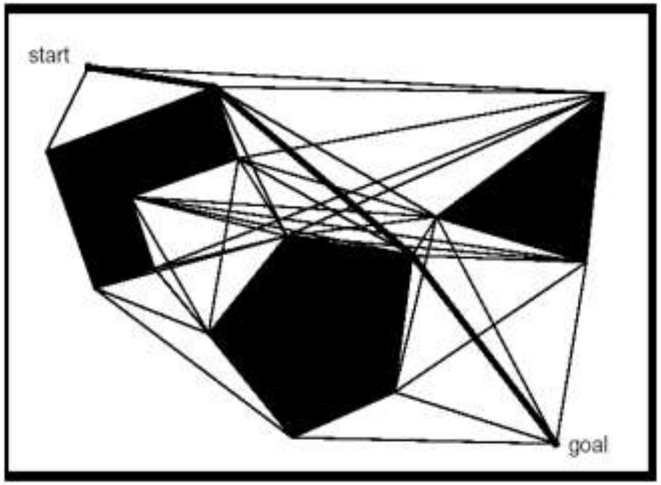


Roadmap-based path planning



Visibility Graph

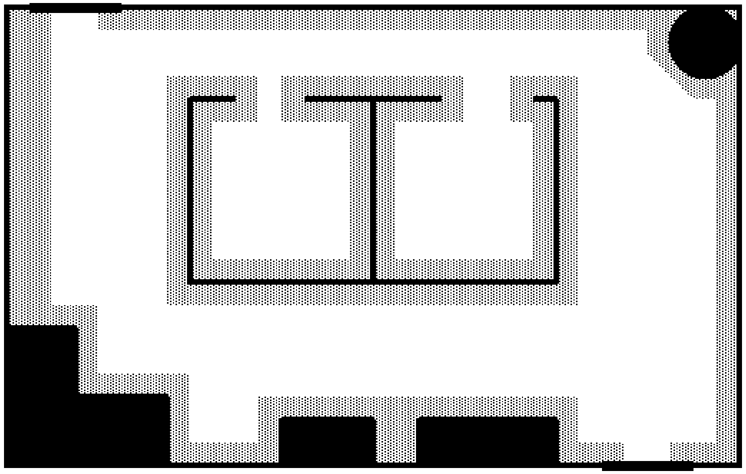
- the *visibility graph* consists of all edges joining vertices that can see each other
- objects in the environment are polygons in either discrete or continuous space
- the size of the representation and the number of edges and nodes increase with the number of polygons
- paths take the robot as close as possible to obstacles on the way to the goal

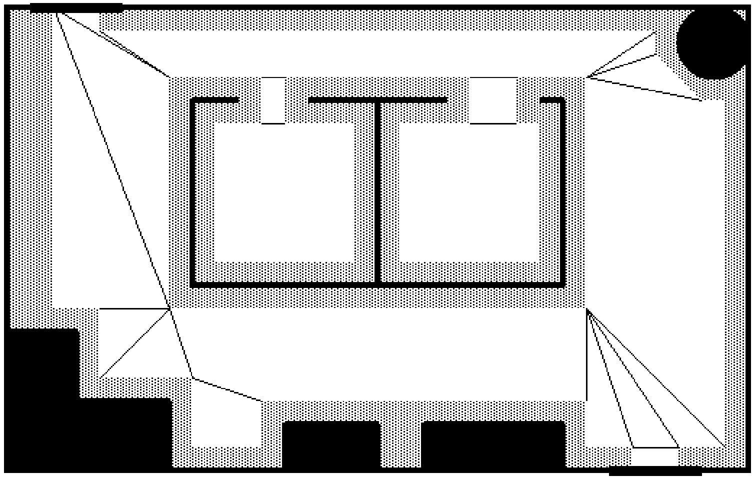


- the length of the solution path is *optimal*
- sense of safety from obstacles is sacrificed for this optimality
- one solution is to grow obstacles by the robot’s radius or modify the solution path

Meadow Maps

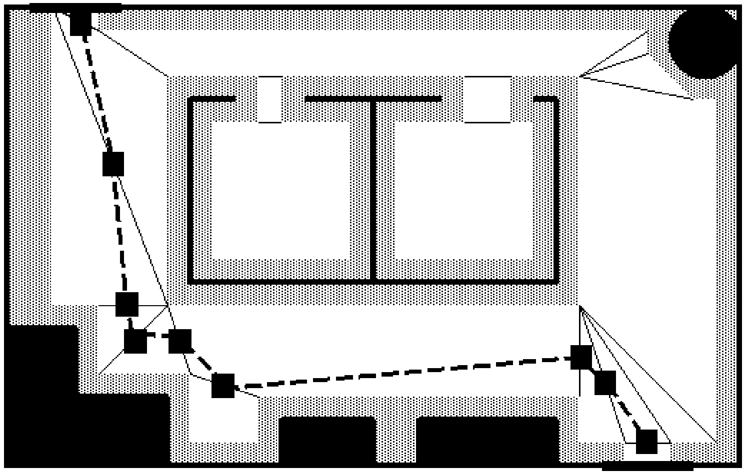
- Visibility graphs are also referred to as Meadow Maps
- The first step is to grow obstacles to be the size of the robot
- Construct convex polygons between pairs of corners or edges

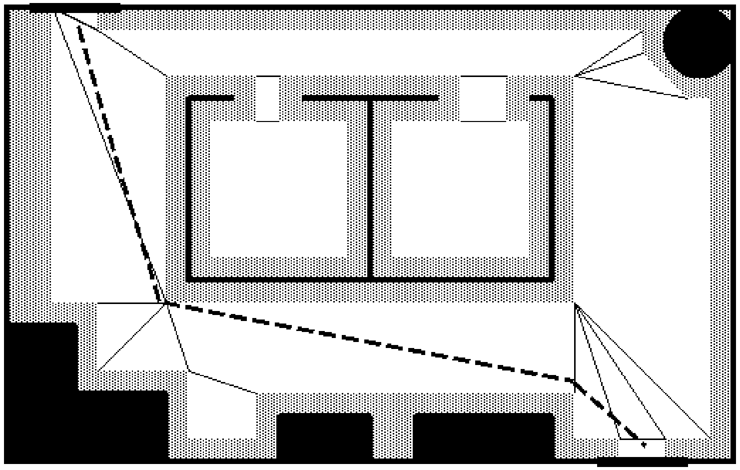




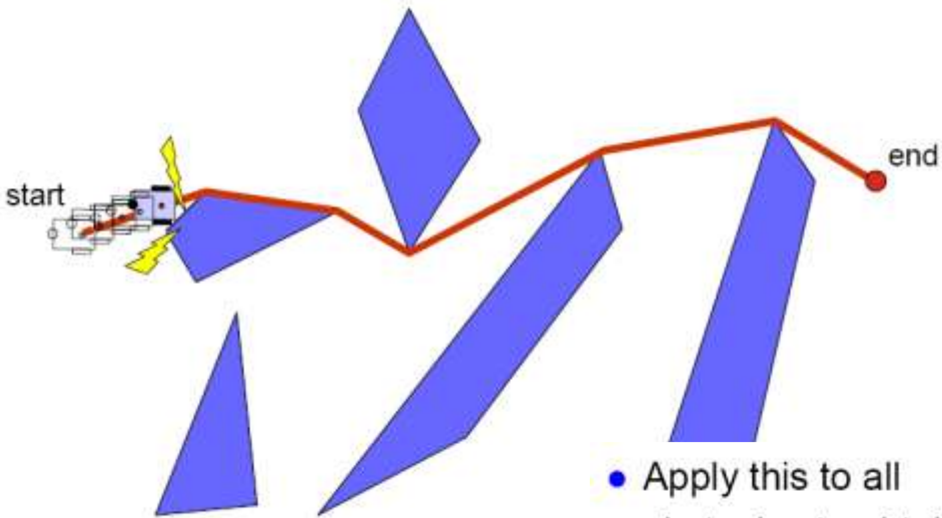
Meadow Maps, Cont.

- Convert the polygons to relational graphs
- Some of the challenges are that there is not a unique set of polygons
- You can’t create this map with sensor data
- The robot cannot recognize corners and edges or the middle for navigation
- Path relaxation may help with some of these challenges

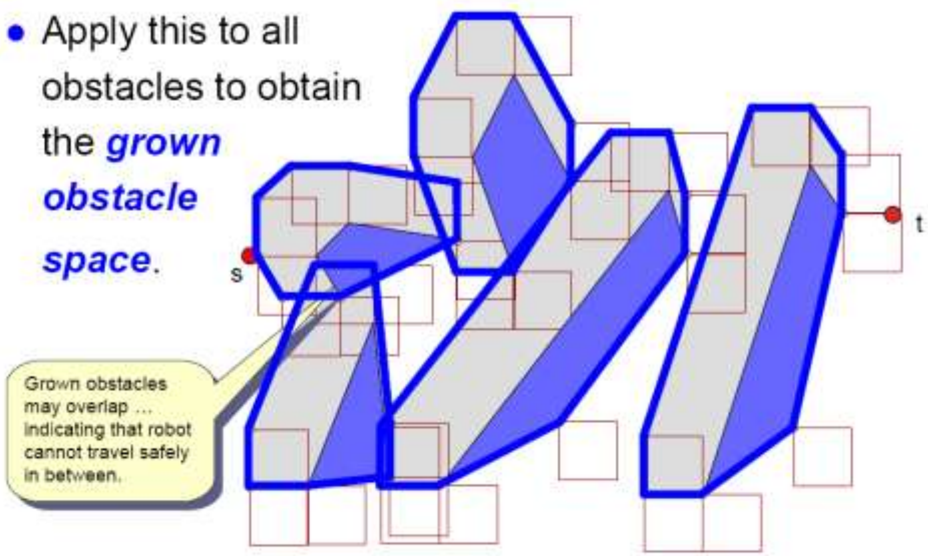




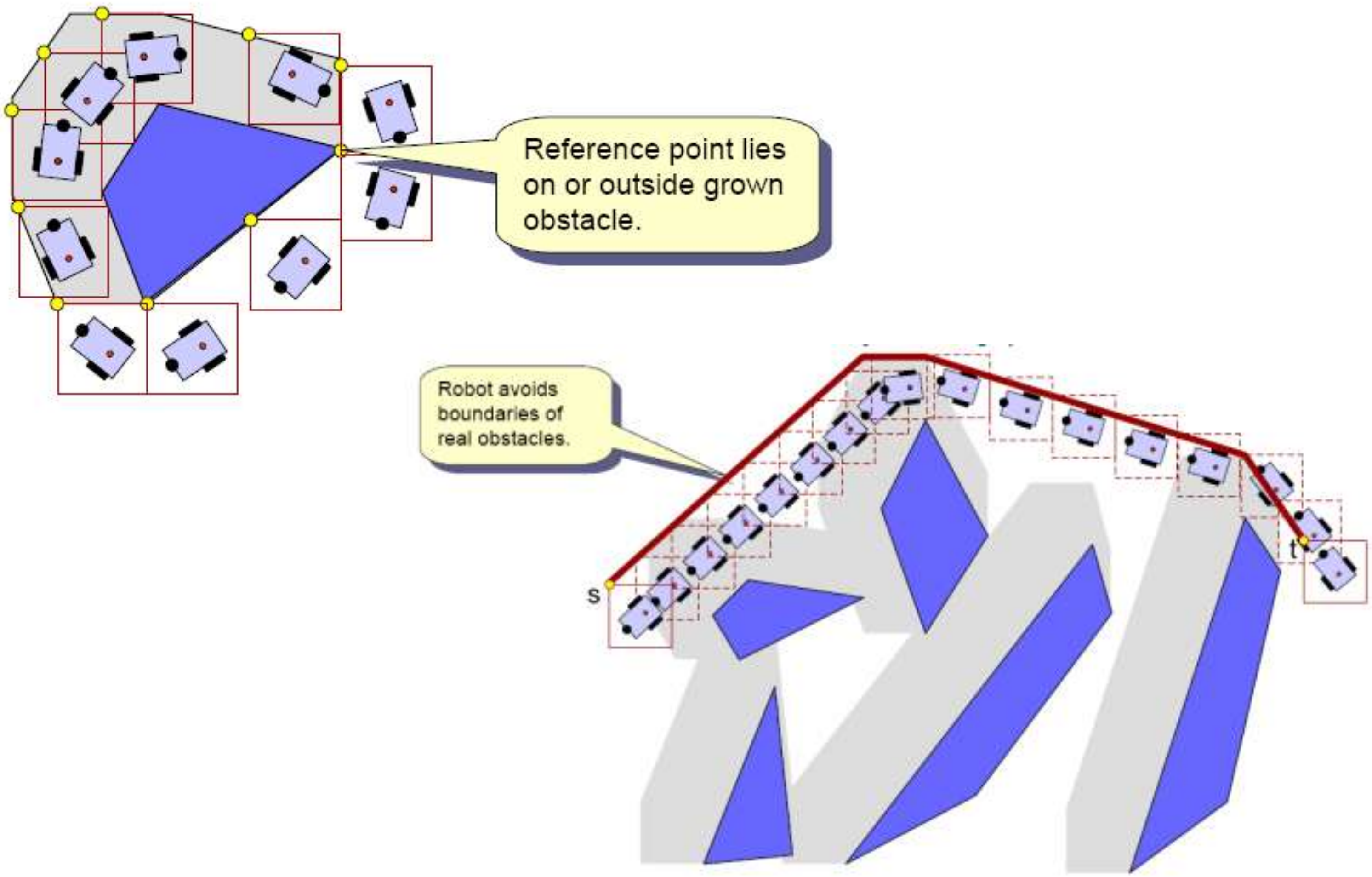
Visibility Graph Paths



- Apply this to all obstacles to obtain the ***grown obstacle space***.

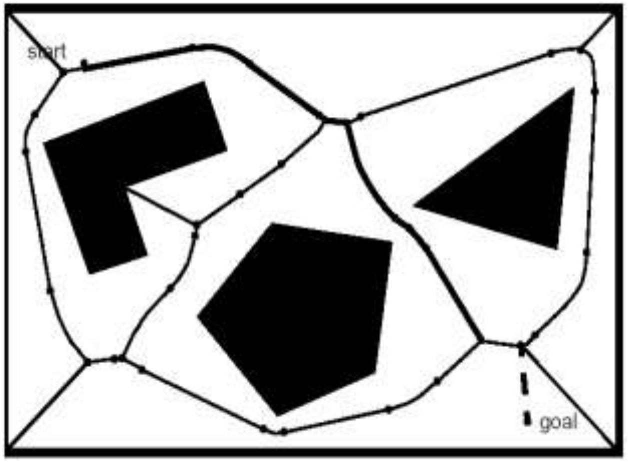


Visibility Graph Paths



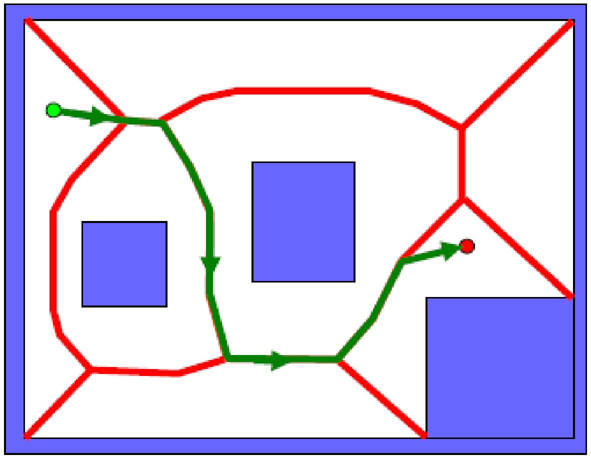
Voronoi Diagram

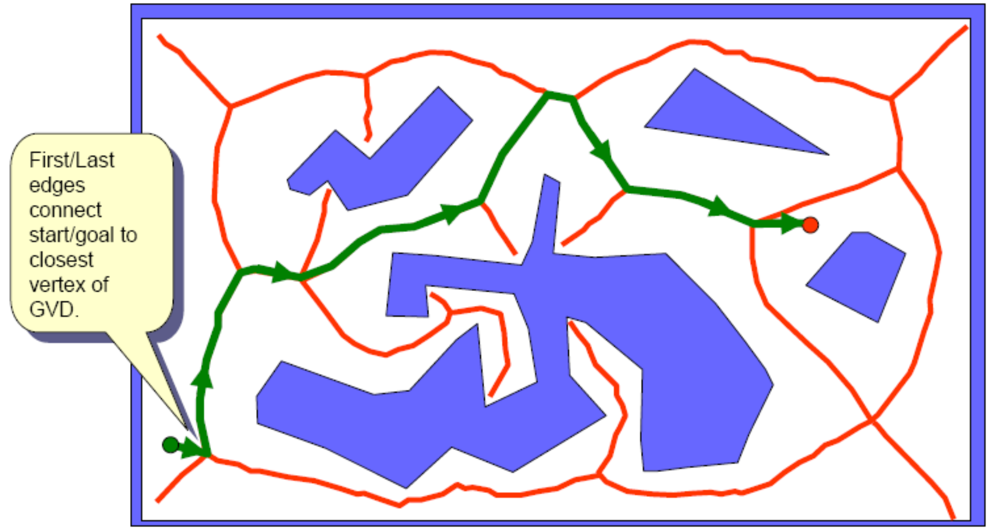
- a *Voronoi diagram* is a complete road map method that tends to maximize the distance between the robot and obstacles
- paths on the Voronoi diagram are usually far form optimal in the sense of the total path length
- one important weakness is in the case of limited range localization sensors. these sensors will be in danger of sensing its surroundings
- the *Voronoi diagram* has the advantage in *executability*

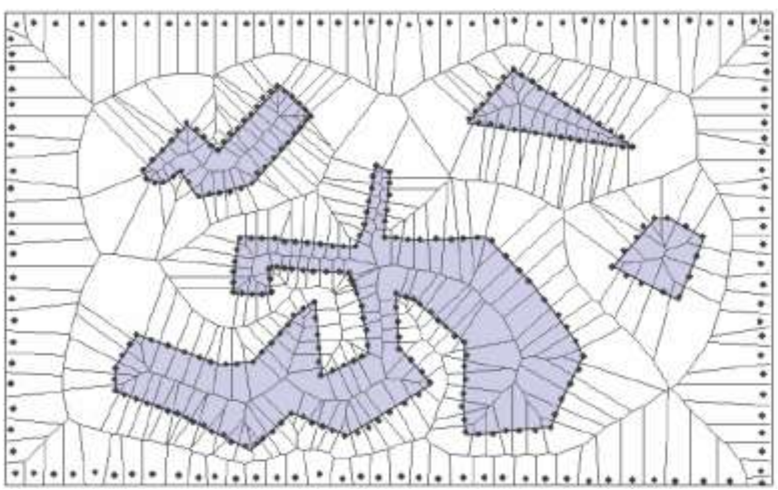


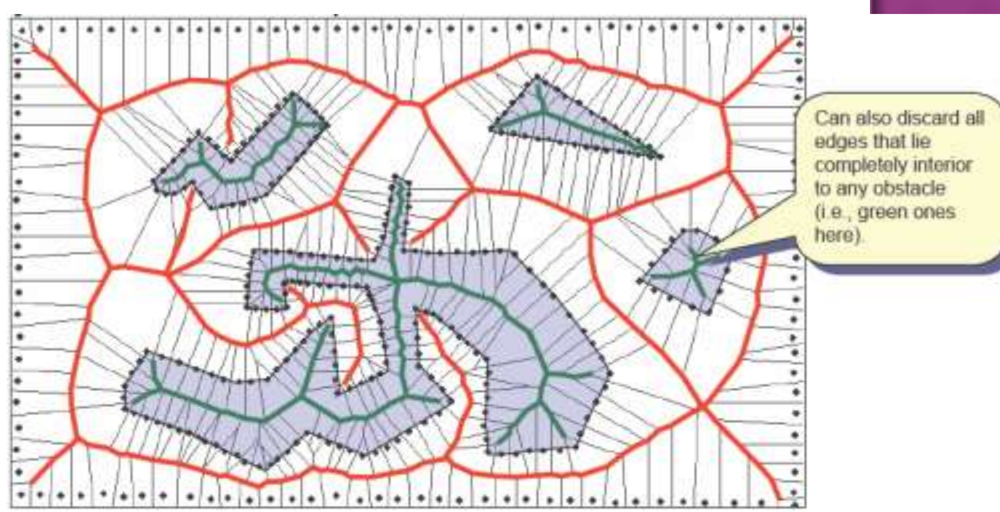
- the robot maximizes the readings of the local minima in its sensor values
- this has been used to conduct automatic mapping by finding and moving on unknown Voronoi edges and then constructing a consistent Voronoi map of the environment

Voronoi Diagram

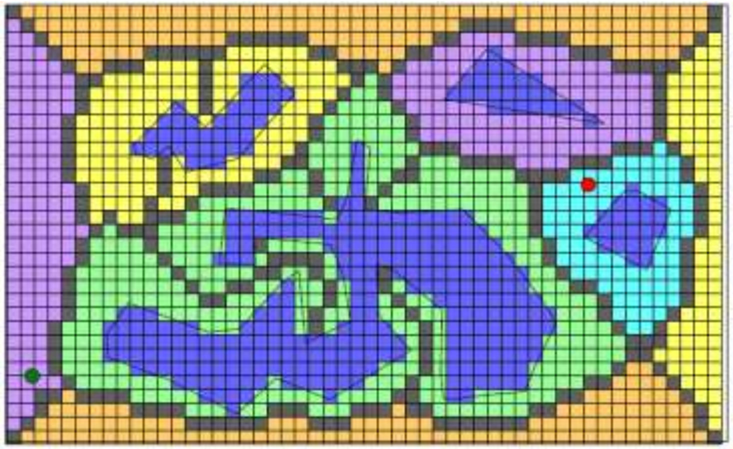


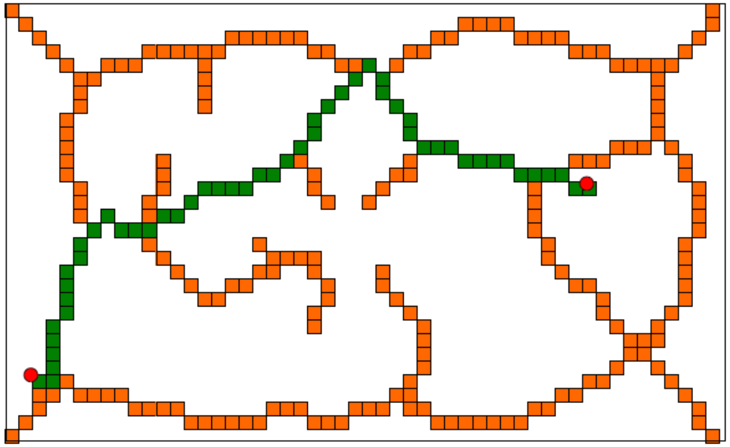






Discretized Voronoi Diagram



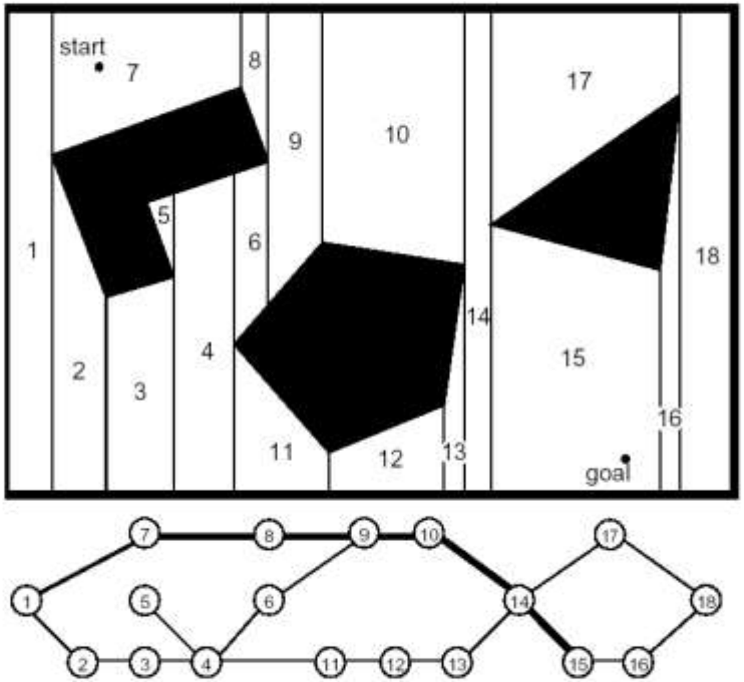


Cell Decomposition path planning

- Use cell decomposition to discriminate between geometric areas, or cells that are free and those that are occupied by objects
- Divide space into simple, connected regions called *cells*
- Determine which open cells are adjacent and construct a *connectivity graph*
- Find cells in which the initial and goal configuration (state) lie and search for a path in the connectivity graph to join them.
- From the sequence of cells found with an appropriate search algorithm, compute a path within each cell.
  - e.g. passing through the midpoints of cell boundaries or by sequence of wall following movements.

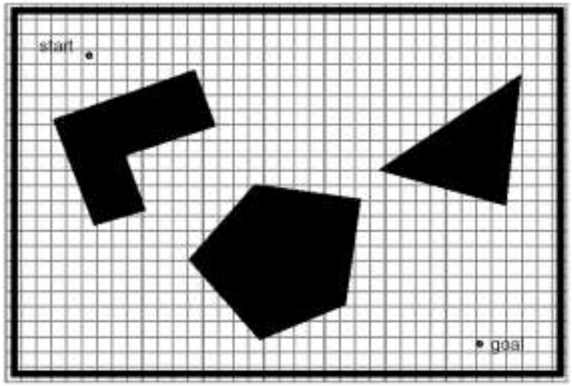
Cell Decomposition Path Planning

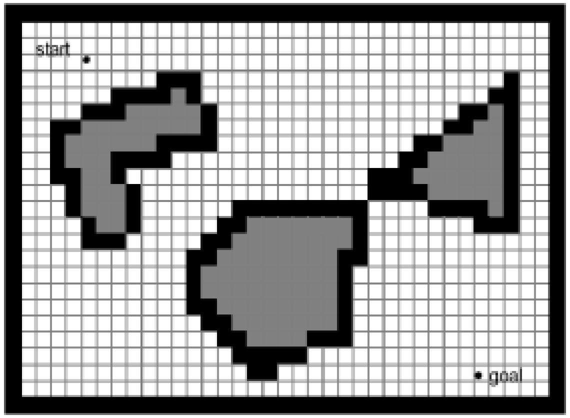
- An important aspect of *cell decomposition* is the placement of the boundaries between the cells
- if the boundaries are placed as a function of the structure of the environment then the method is *exact cell decomposition*
- if the decomposition is an approximation of the actual map, the system is an *approximate cell decomposition*



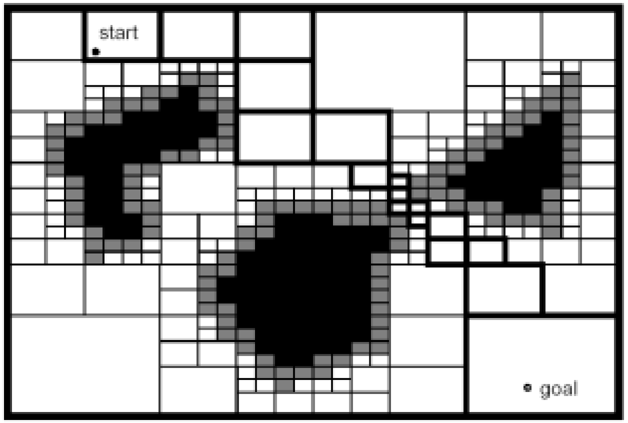
Exact Cell Decomposition

- the boundaries of cells is based on geometric criticality
- the cells are completely free or occupied
- what matters is the robot's ability to traverse from each free cell to adjacent free cells
- efficient computation in that case of large, sparse environment
- used rarely in mobile robot applications due to complexities in implementation





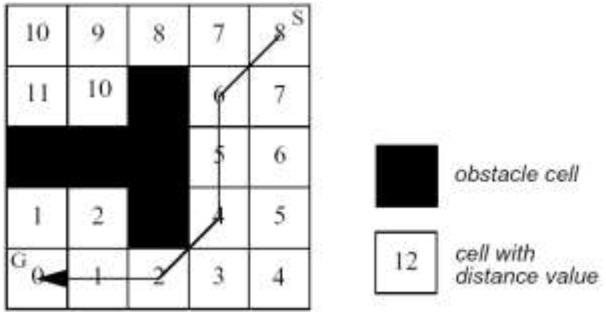
Adaptive Cell Decomposition Quadtree



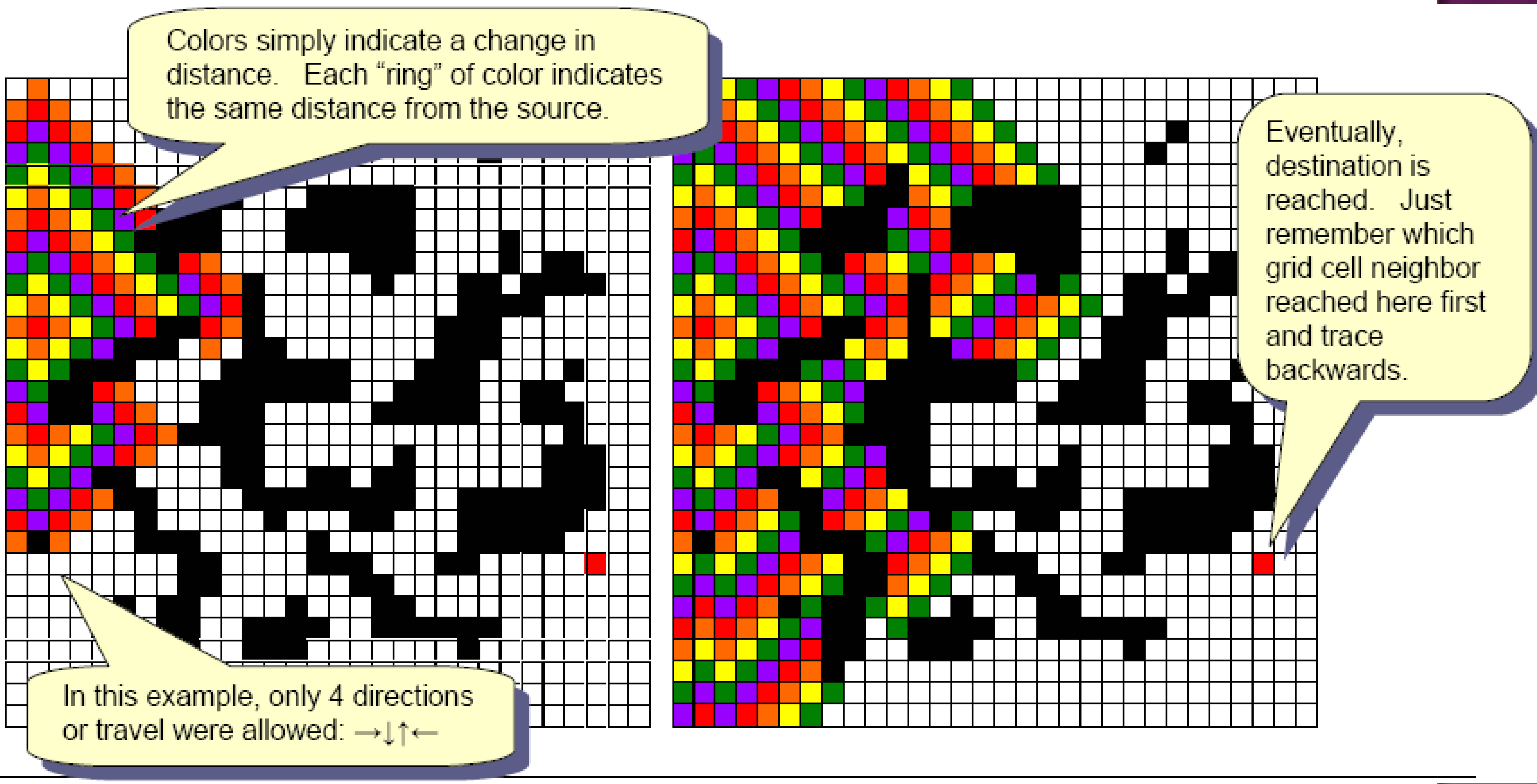
- one of the most popular techniques for mobile robot path planning
- cell size is not dependent upon objects in an environment so narrow passageways may be lost
- low computational complexity for path planning
- the fundamental cost is memory because the grid must be represented in entirety
- sparse environments contain few cells consuming dramatically less memory

Approximate Cell Decomposition

- *Wavefront expansion* or *grassfire* is an efficient and simple to implement technique for finding routes in fixed-size cell arrays
- employs wavefront expansion from the goal position outward, marking each cell's distance to the goal
- this continues until the wave reaches the initial position
- the planner can then estimate the robot's distance to the goal as well as recover a specific solution trajectory by linking together adjacent cells that are always closer to the goal



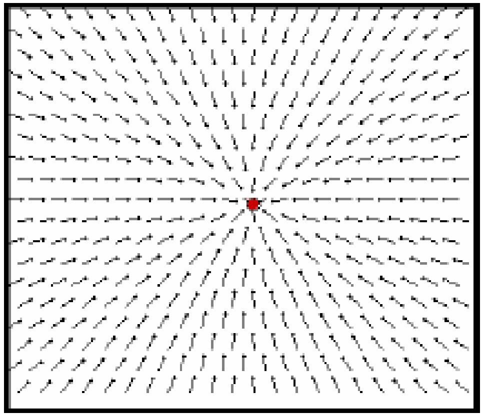
Wavefont propagation



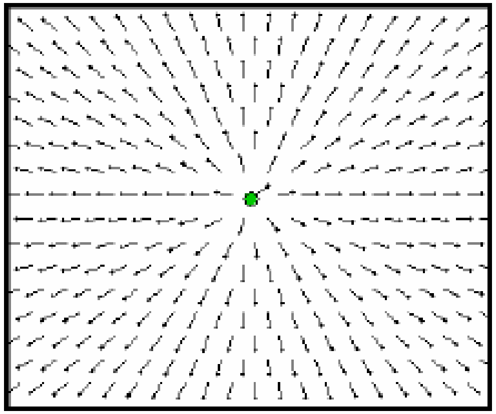
Potential field path planning

- *Potential field path planning* creates a field, or gradient, across the robot's map that directs the robot to the goal position from multiple prior positions
- Robot is treated as a *point under the influence* of an artificial potential field.
  - Generated robot movement is similar to a ball rolling down the hill
  - Goal generates attractive force
  - Obstacles are repulsive forces
  - the superposition of all forces is applied to the robot
  - artificial potential field smoothly guides the robot toward the goal while simultaneously avoiding obstacles

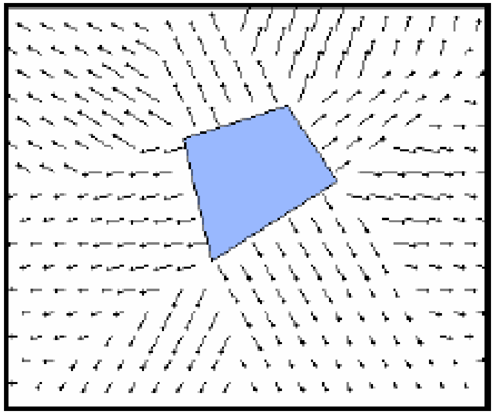
Potential field path planning



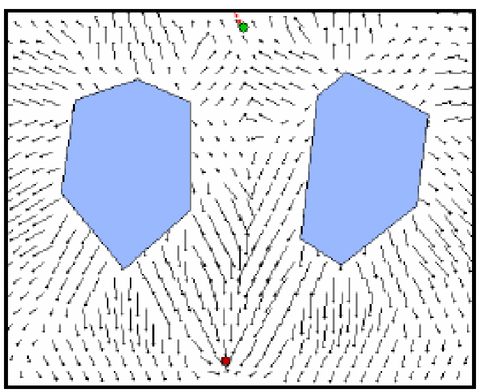
Attract to goal



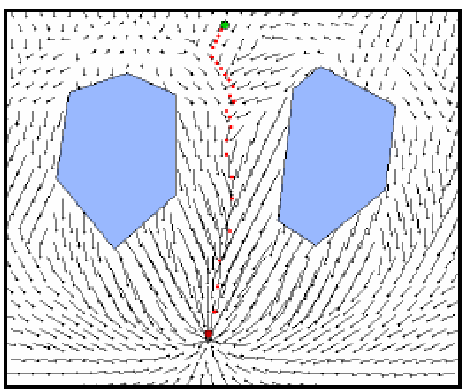
Repel from source



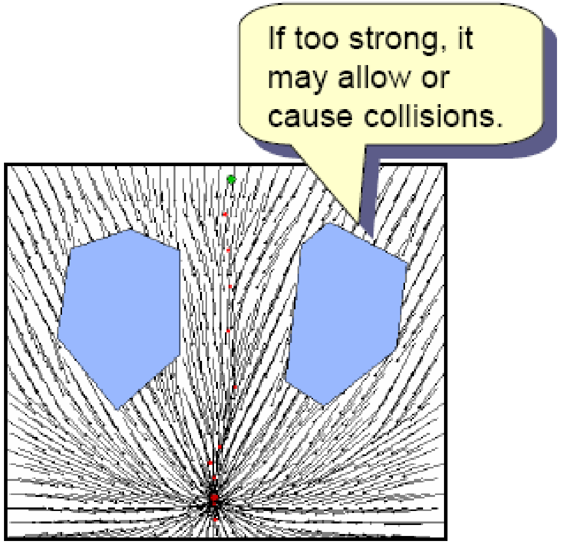
Repel from obstacle



weak goal attraction

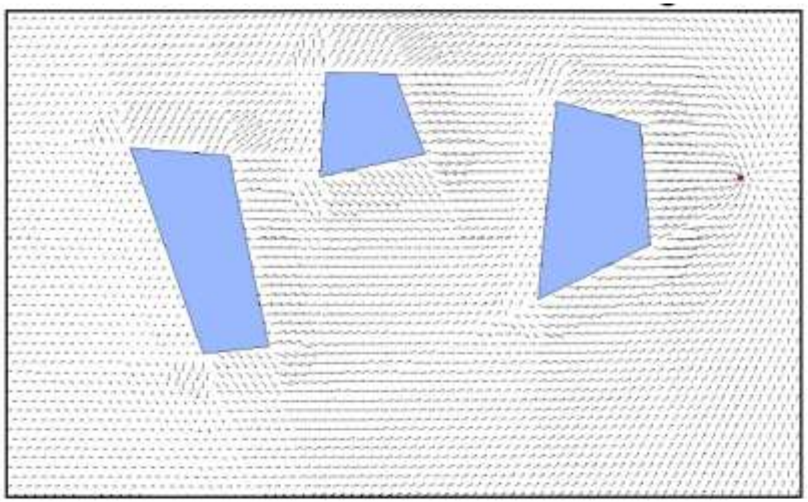


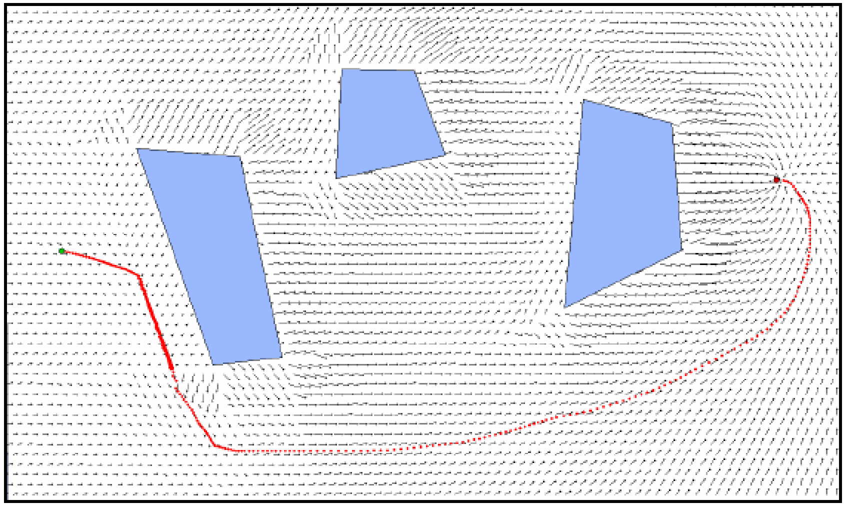
medium goal attraction



strong goal attraction

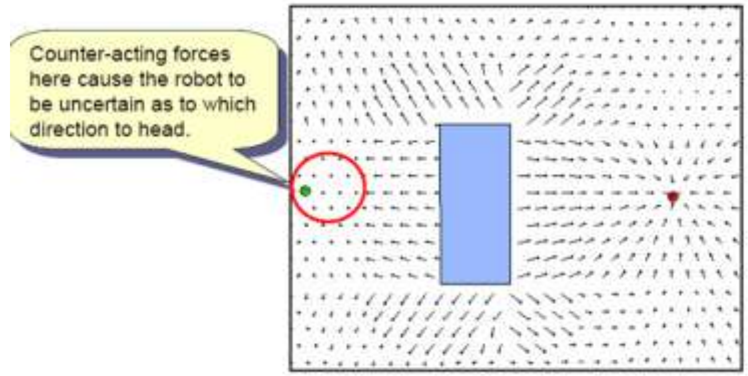
Potential field path planning

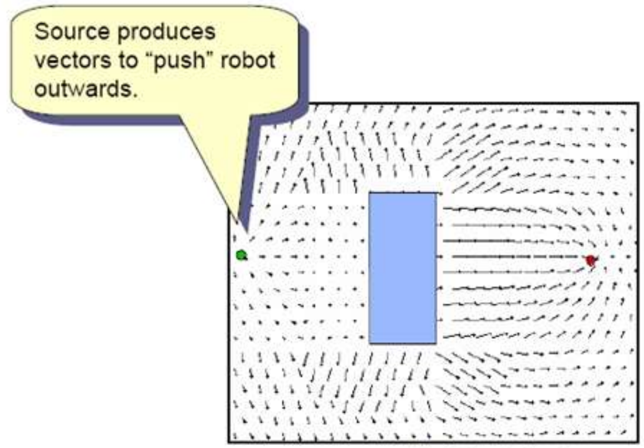




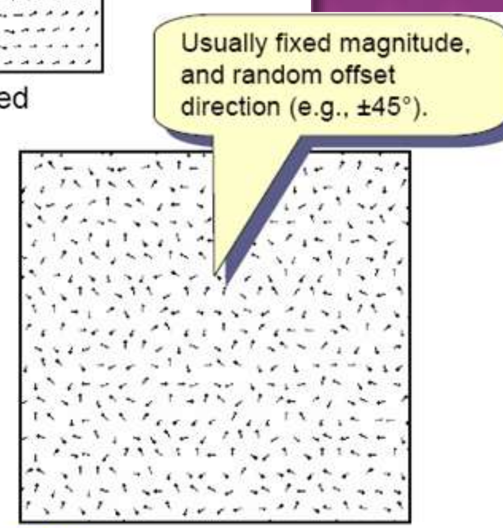
Local Minima

- A local minima occurs when the robot gets stuck due to counteracting forces
- To overcome a local minima problem
  - Add a field from the source to push away from it
  - Introduce noise into the environment
  - The outward source may still lead to local minima but noise may overcome this



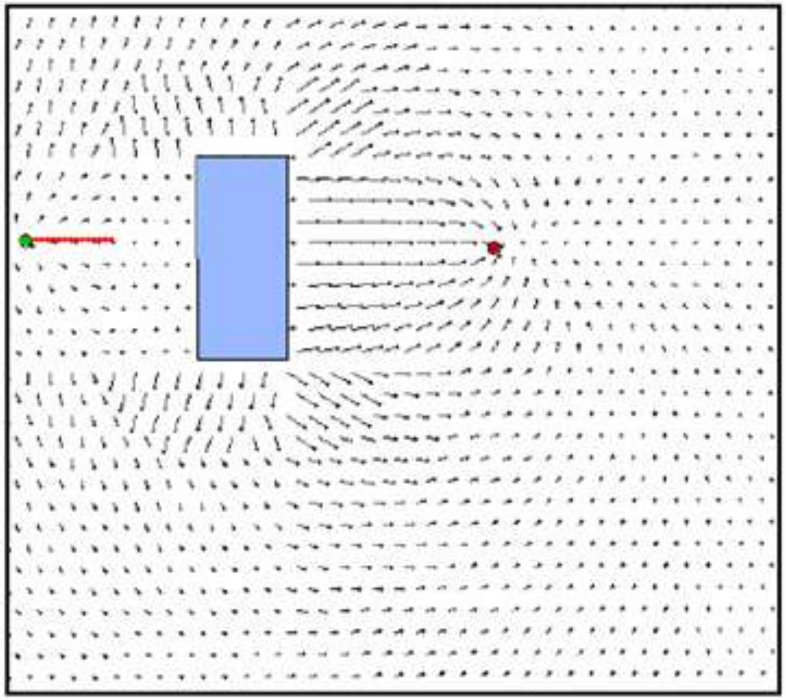


source field added

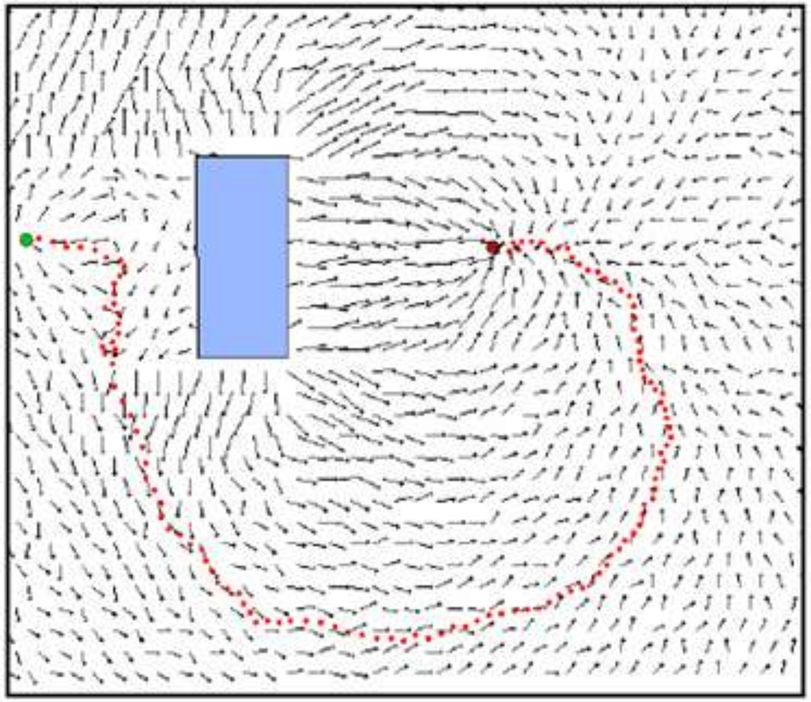


noise (i.e., random vectors)

Local Minima



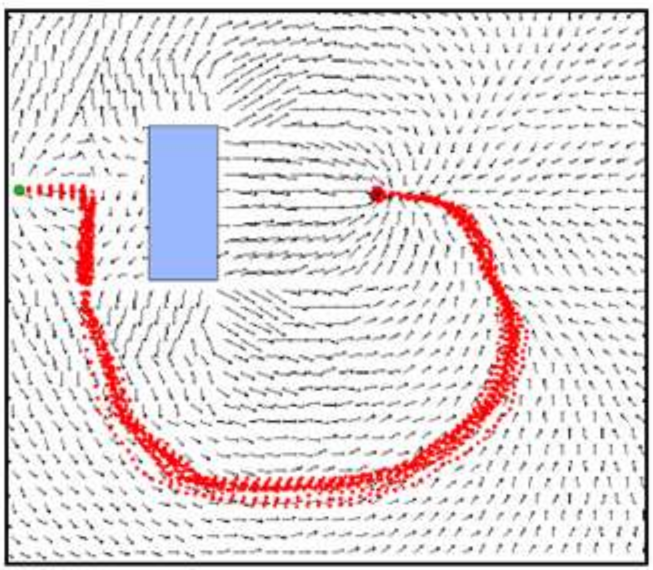
without noise, no path



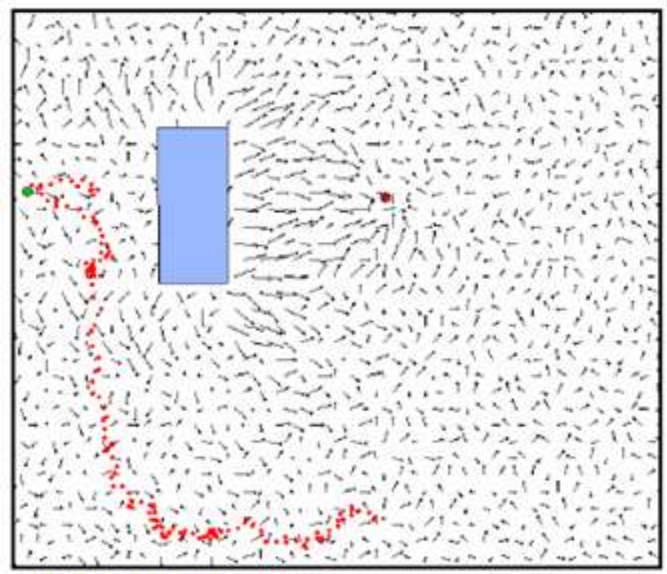
with noise, path found

Local Minima – Random Noise

- The path will vary depending on the random values of the noise vector
- Too much noise will not work and no path will be found

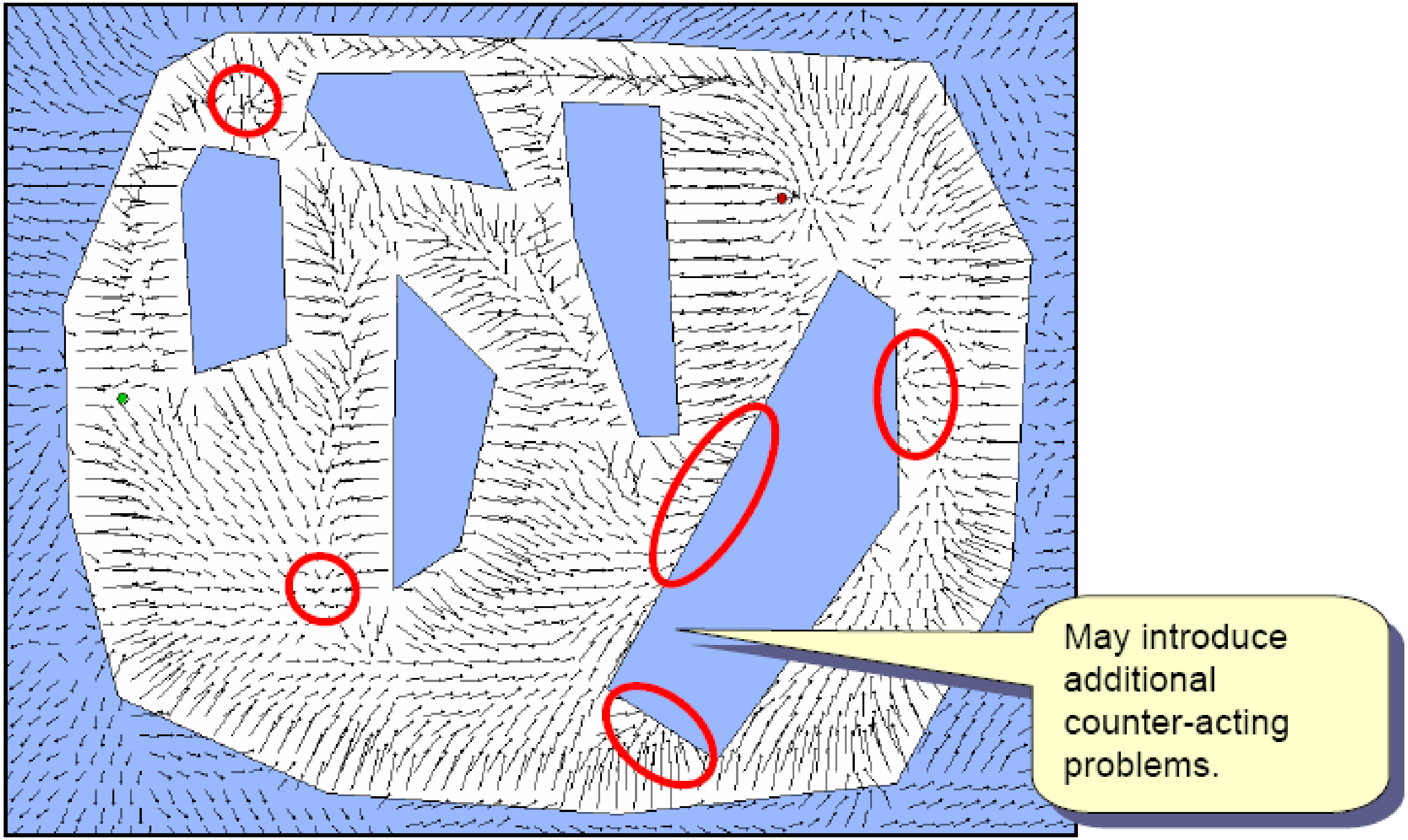


multiple iterations

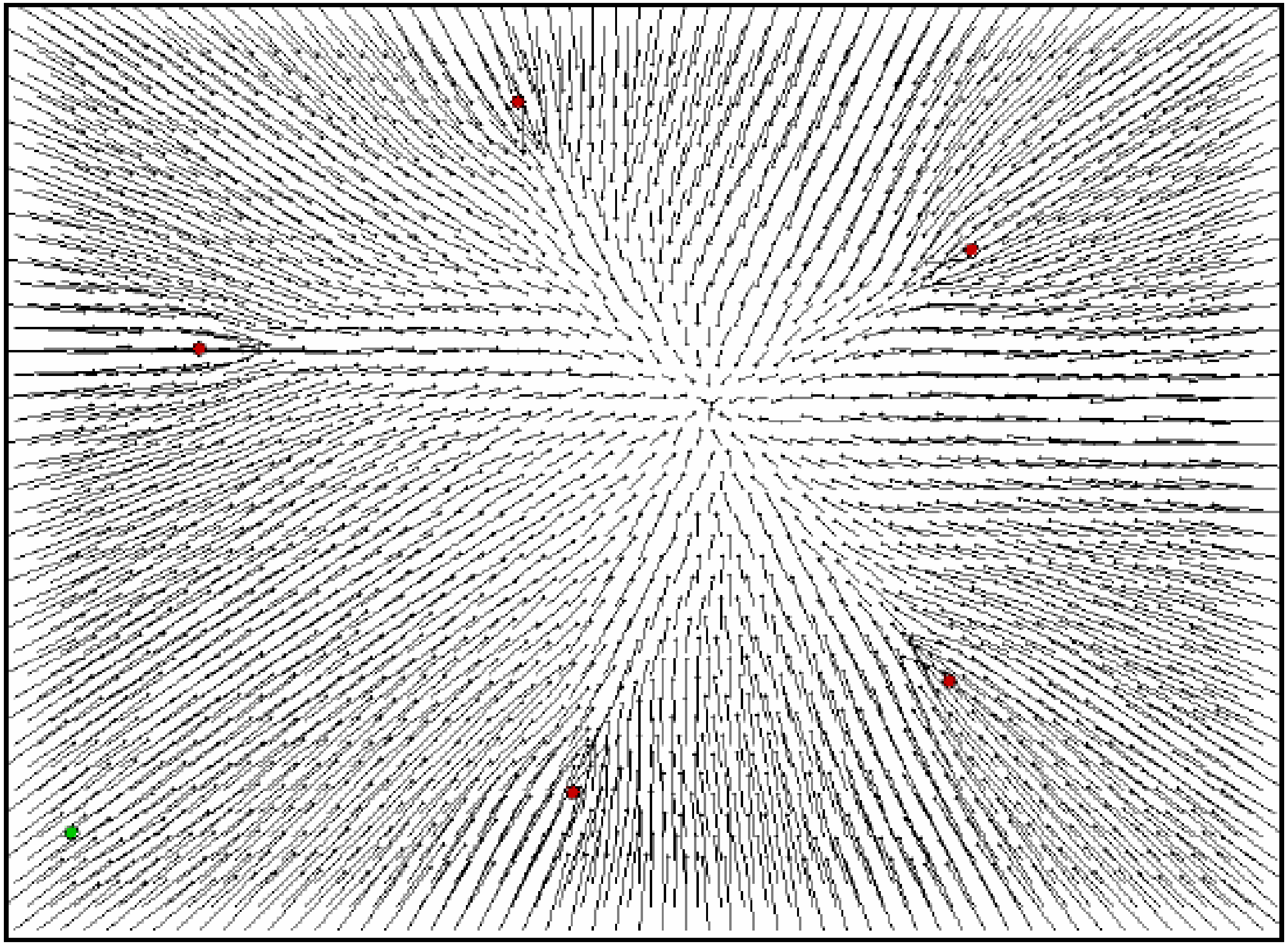


too much noise, no path found

Counteracting fields
Environment Boundary and Obstacles

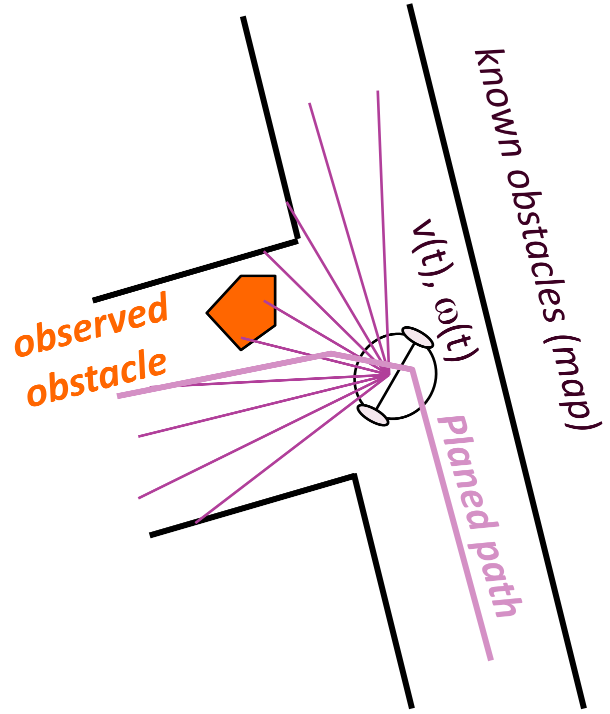


Potential fields cannot handle multiple goals



Obstacle Avoidance with Path Planning

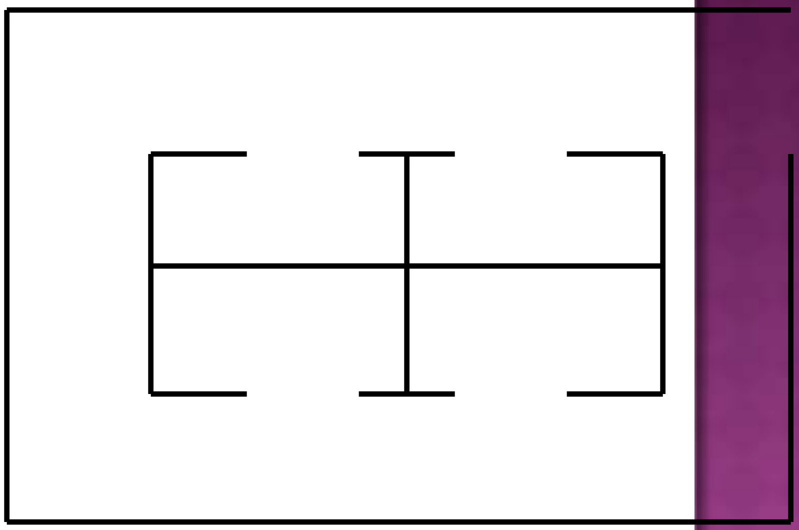
- Local obstacle avoidance will avoid collisions by changing the robot's trajectory as informed by sensor readings and the relative location to the goal position.



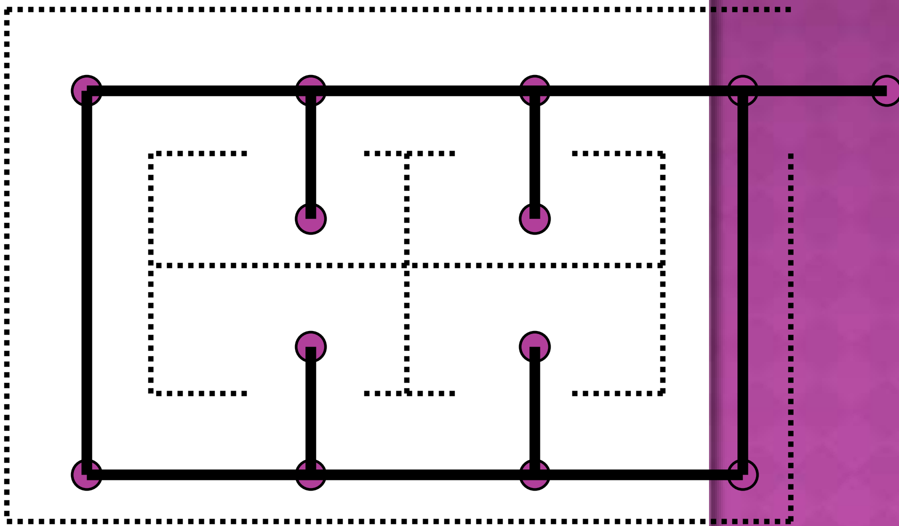
Topological Path Planning

- A *relational graph* has nodes which represent landmarks, gateways and goal locations
- The *gateways* are opportunities for the robot to change the path heading
- The edges of the relational graph represent a navigable path

floor plan

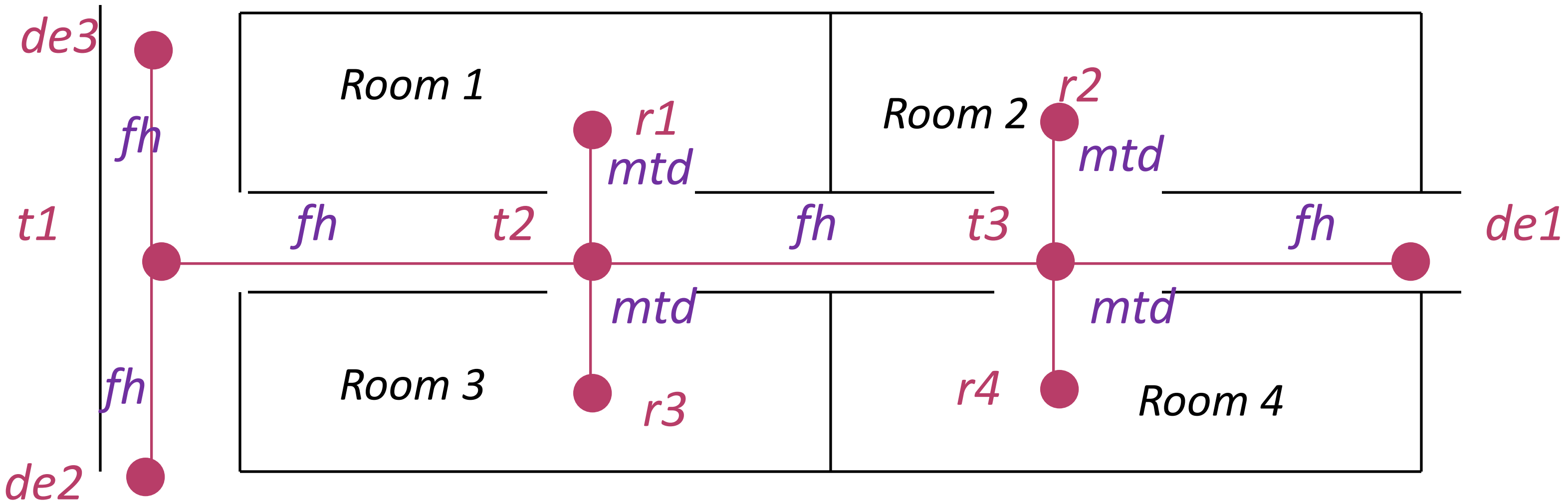


relational graph



Control Scheme for Relational Graph

- The following floor plan has been made into a relational graph
- Each edge should be labeled with the appropriate localization control scheme
  - mtd: move through door
  - fh: follow hall
- Each node should be labeled with the type of gate way
  - t: t – junction
  - de: dead end
  - r: room

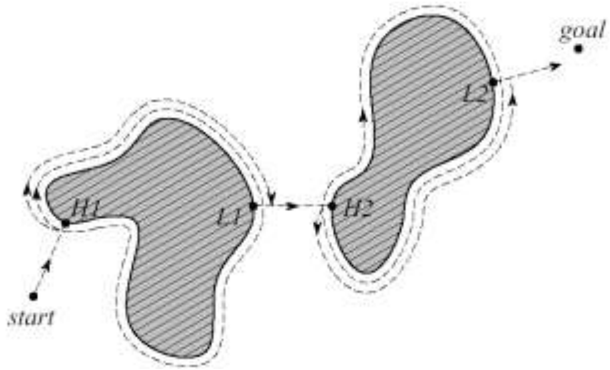


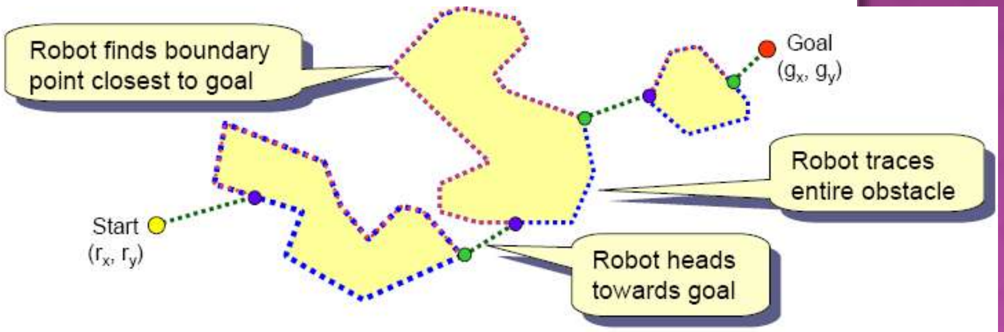
Transition Table for Relational Graph

	Room	T-junction	Dead End
Room	Undefined	Move through doorway	Undefined
T-junction	Move through doorway	Follow hall	Follow hall
Dead End	Undefined	Follow hall	Undefined

Obstacle Avoidance
Bug Algorithm

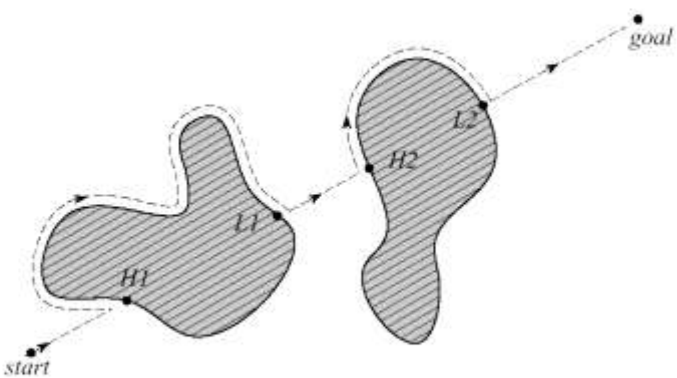
- This is the simplest algorithm
- Follow the contour of each obstacle until it is fully circled before it is left at the point closest to the goal
- Very inefficient but it guarantees that the robot will reach any reachable goal

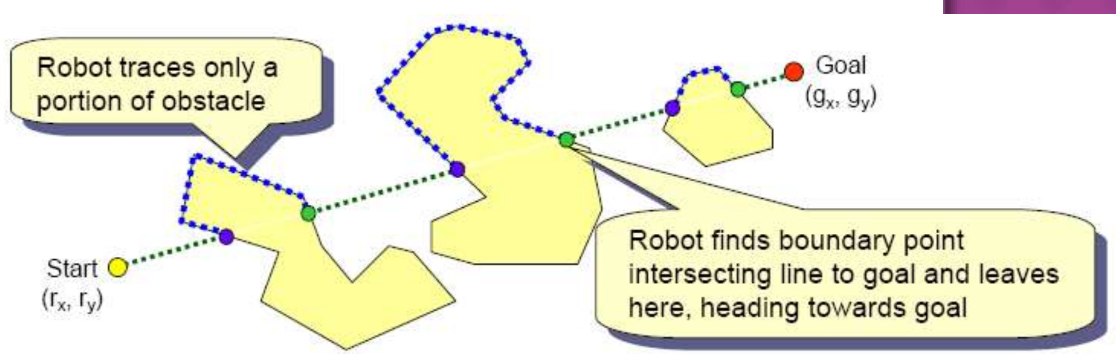




Obstacle Avoidance
Bug2 Algorithm

- Follows the obstacle always on the left or right side
- Leaves the obstacle if the direct connection between start and goal is crossed
- Has significantly shorter total robot travel





Obstacle Avoidance Tangent Bug

- The tangent bug adds range sensing and a local environmental representation termed the *local tangent graph (LTG)*
- The *LTG* approaches globally optimal paths