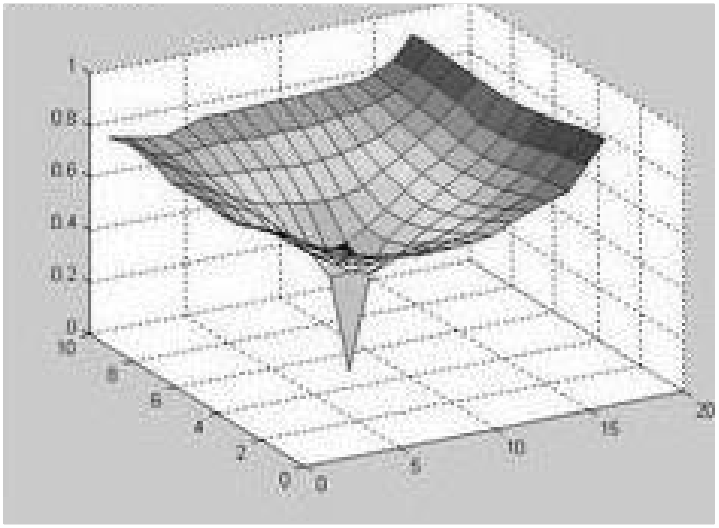
Maps and Planning

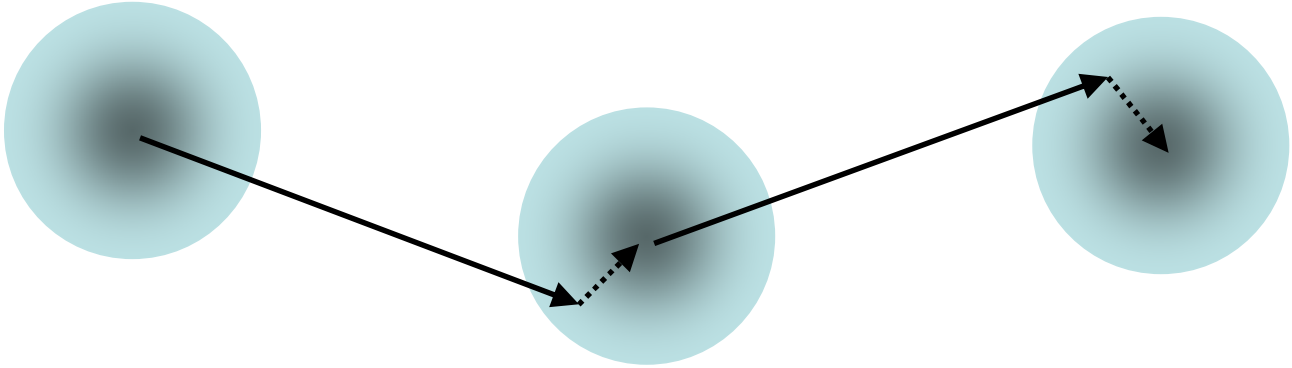
IAR Lecture 6

Barbara Webb

From local strategies to routes to maps

- (From lecture 4) Beacon or memory of home view allows navigation from a surrounding catchment area
- Multiple beacons or memories can be linked together to form routes





From local strategies to routes to maps

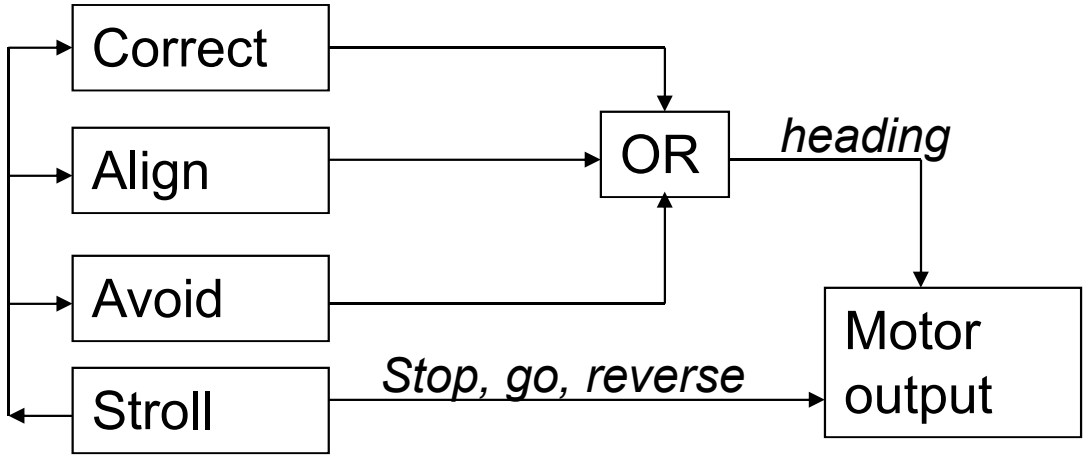
- If we can combine routes (or close loops in routes) by recognising overlapping locations we create a graph representation of the world
- Problem is to reliably recognise when encounter the same location:
  - From different approach directions
  - With possible alterations to appearance (e.g. lighting)
  - In ‘wrong place’ according to odometry
- But not to confuse locations that are different but look similar
- And ideally, to do this with an efficient algorithm.

Topological maps

- Represent known locations and the connections between them as nodes and edges in a graph
- The edges could represent simple adjacency, the raw actions needed to get from one node to the next; or direction, distance, path convenience etc.
- Can determine a possible (or even the optimal) route by standard graph search methods

Example: “Toto”

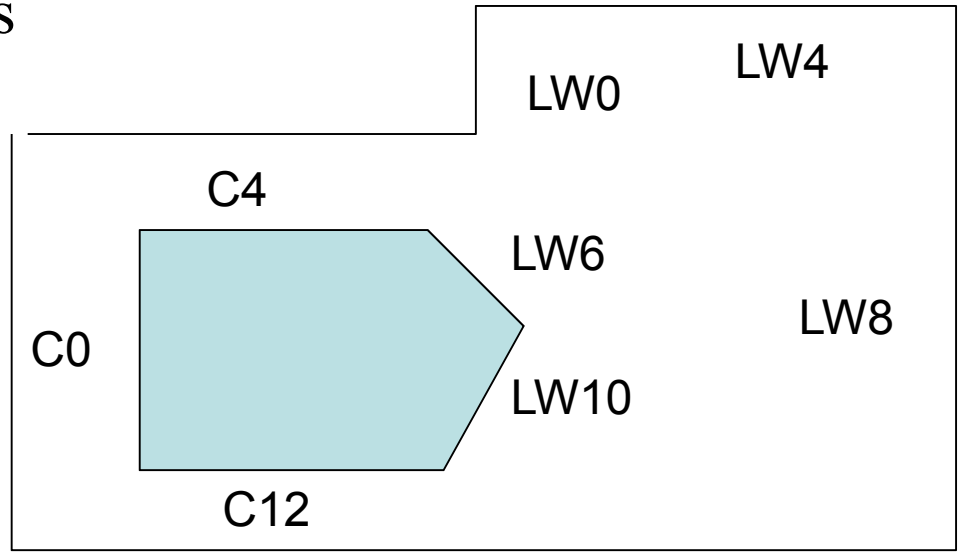
- Mataric & Brooks (1990)
- Robot with compass and 12 sonar sensors
- Subsumption architecture to follow boundaries:

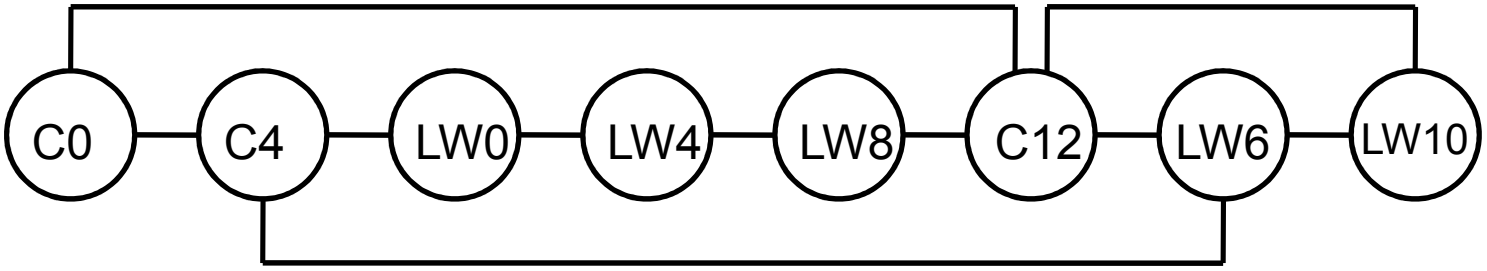


- Then add layers for landmark recognition, storing, and navigation

Example: “Toto”

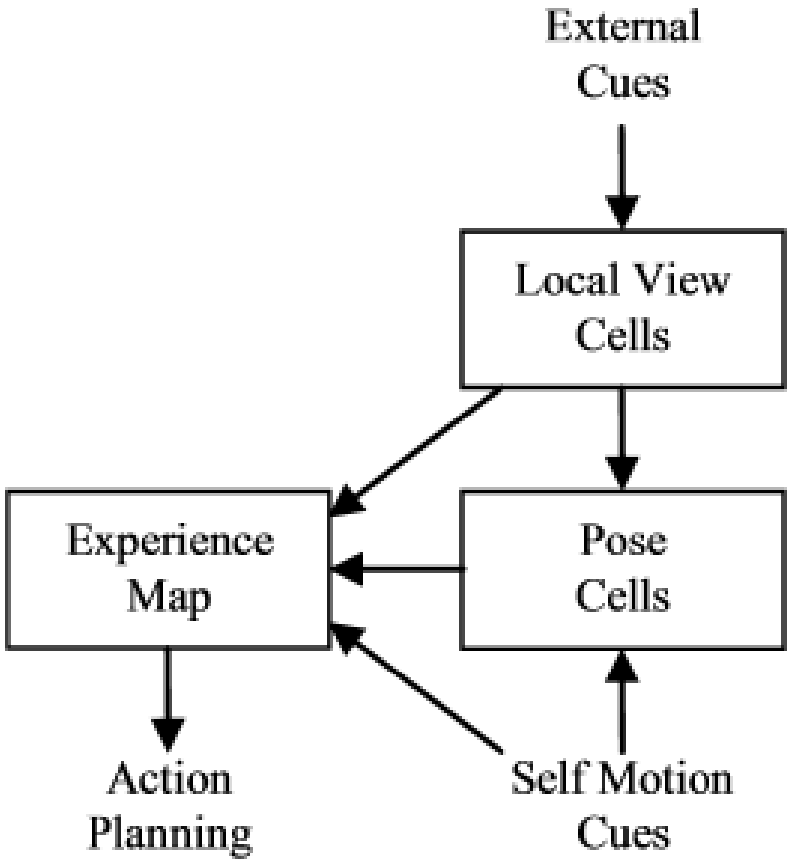
- Identify ‘wall’, ‘corridor’ or ‘junk’ as dynamic landmarks (i.e. pattern of sensor input and movement over time)
- Stored in simple graph structure reflecting the sequence of landmarks encountered when exploring
- Can find shortest paths by activity propagation



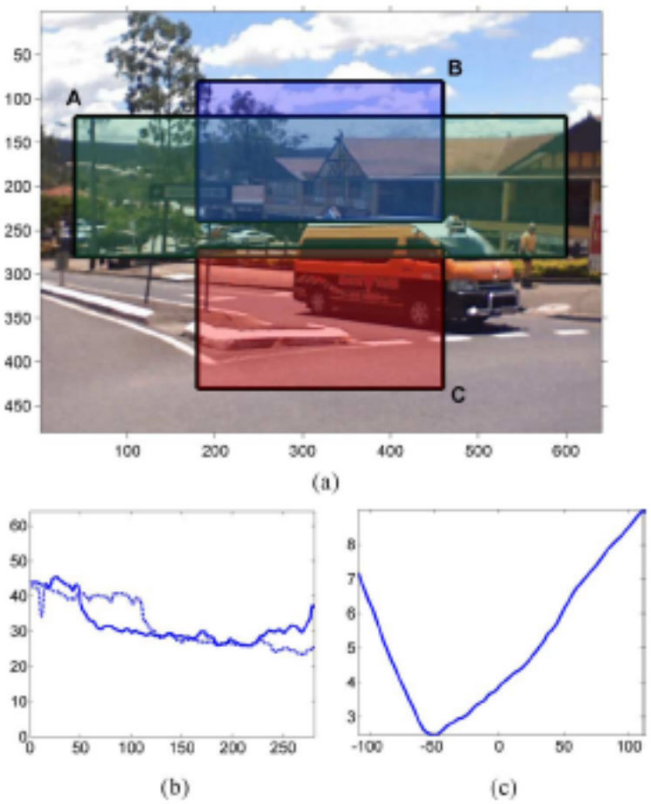


Example: RAT-SLAM

- For each new local view and/or pose, store an ‘experience’ node, linked to the previous node by a transition derived from the self motion (experience nodes are like rat place cells)
- Recognise when same view and pose occur to close loops in the experience map
- When closing loops, align the transitions and poses for geometric consistency to correct for drift



RAT-SLAM



- Both local views and self-motion are derived from vision
- Use different parts of visual field for:

A: Local view: compare to previously stored templates; either recognise or store as new template

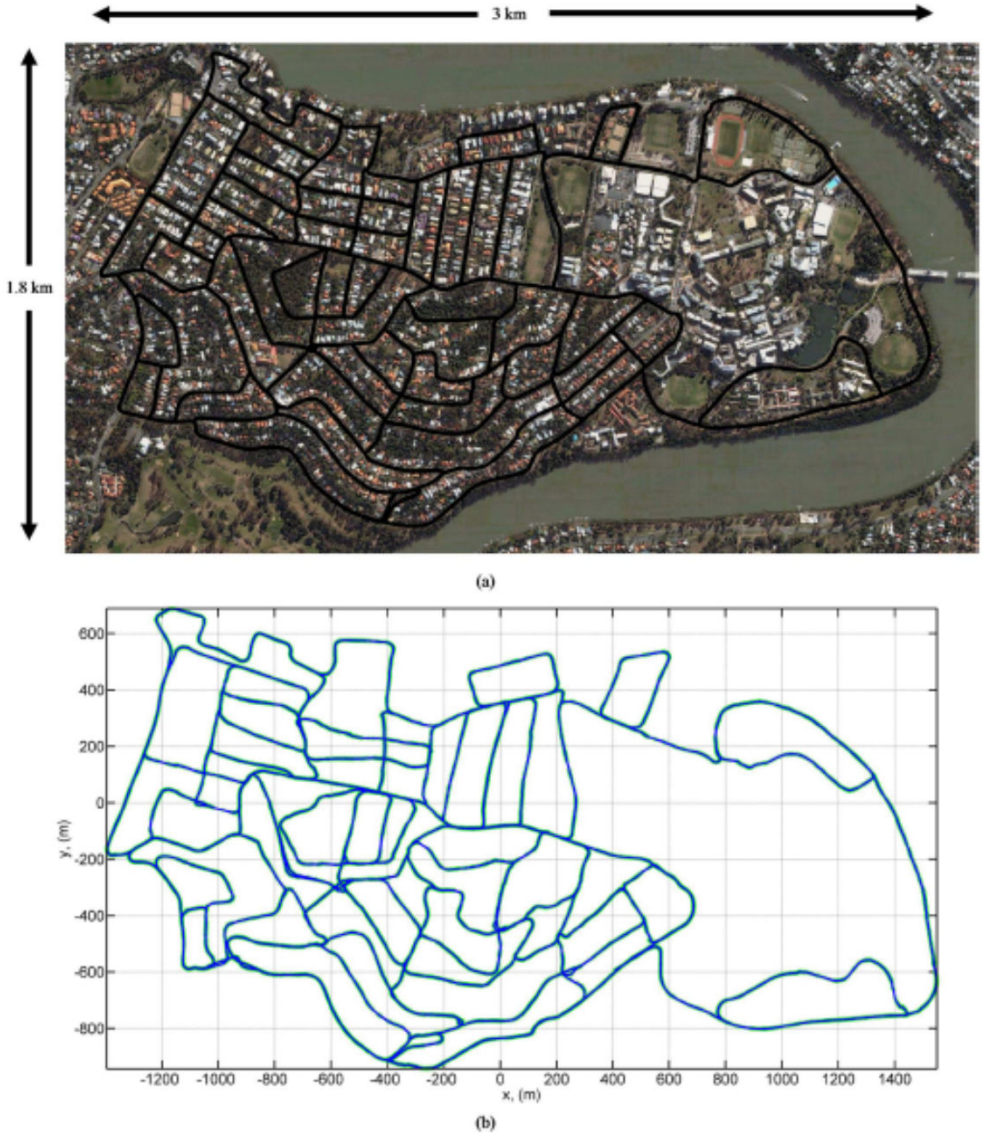
B: Rotation estimate: find sideways pixel shift that produces best match.

C: Speed estimate: for best rotation, take image difference.

- In each case reduce image to one-dimensional scanline of normalised intensity across columns.

RatSLAM Results 2008

- Using built-in laptop webcam, drove for 100 minutes through 3km by 1.6km area of Brisbane
- Visualising the ‘experience map’ shows method produces a fairly accurate map that could be used for navigation

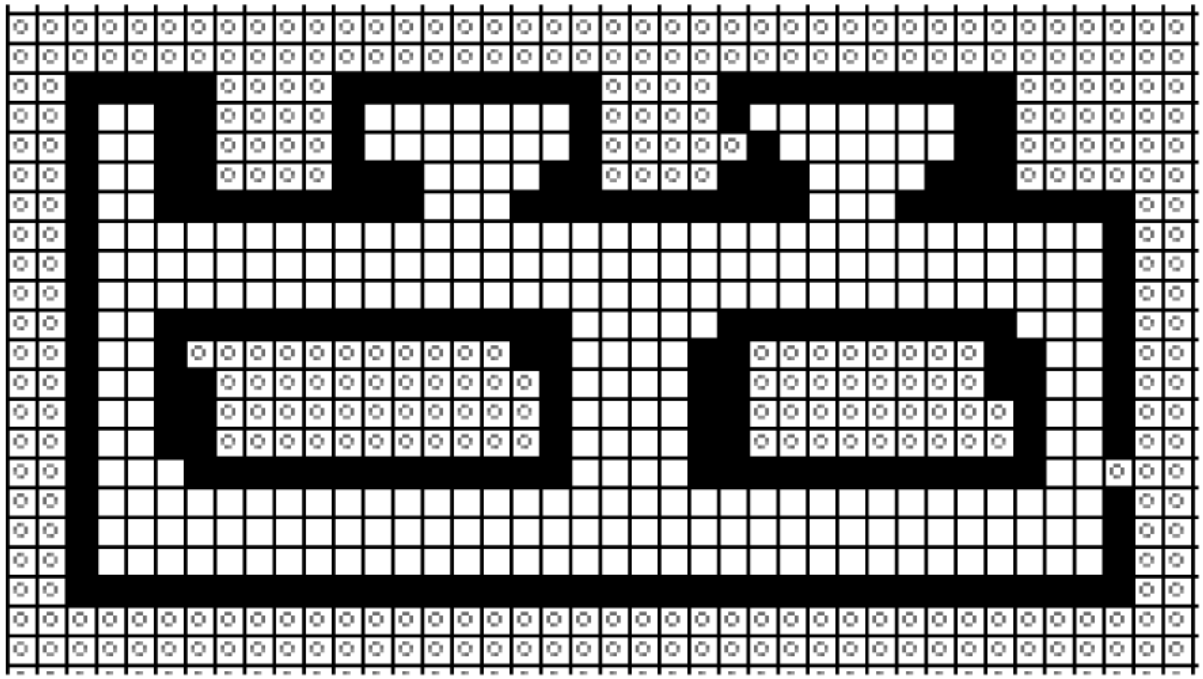


From topological to metric maps

- A graph in which edges represent the distances and directions between locations in consistent global co-ordinates is effectively a metric map.
- We describe the location of the robot and objects in its world in some kind of absolute coordinates (e.g. cartesian or polar)
- For simple robot, moving on ground plane and able to rotate on the spot, could consider this as 2 degree of freedom *configuration space* (i.e., where the robot can move to)*:*
  - Only obstacle location matters (not identity)
  - Assume robot is holonomic or can rotate on spot
  - Treat robot as a single point and expand obstacle boundaries by robot's maximum dimension

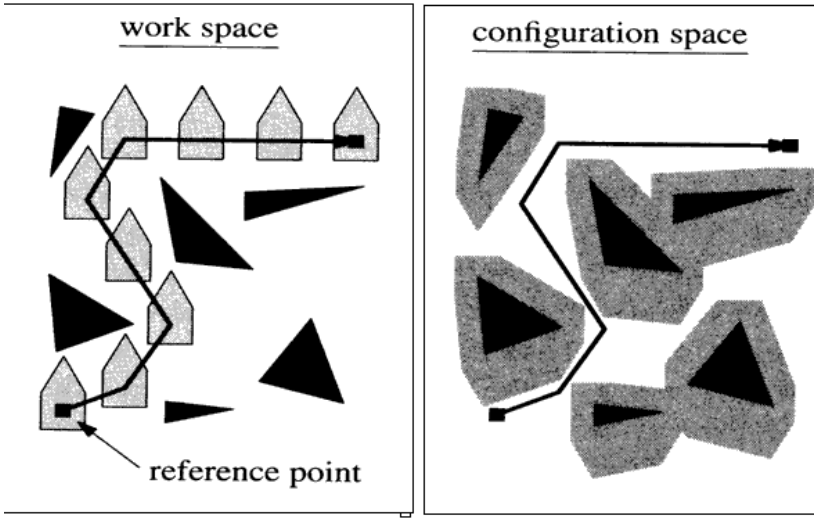
Example: Occupancy grid representation

- Divide space into a regular rectangular grid at some specific resolution
- Mark each grid square as either ‘occupied’, ‘empty’ or ‘unknown’

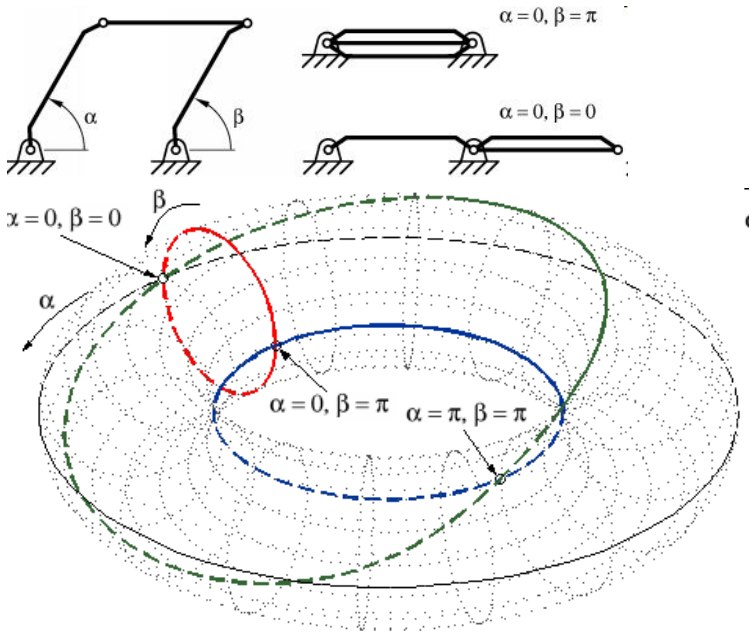


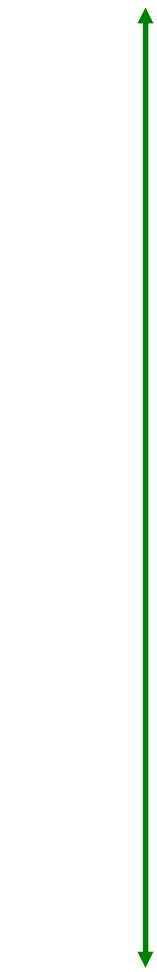
Configuration space:

- Given the kinematic description of the robot (e.g. holonomic or not, 3-d motion, articulation)
- Describes the possible state space of the robot (each point in the space is a possible pose or configuration of the robot)
- Can map kinematic obstacles into the C-space
- And thus represent the possible connected trajectories that can be taken



N.B. Here would need a corresponding C-space for every orientation of the robot to represent pose





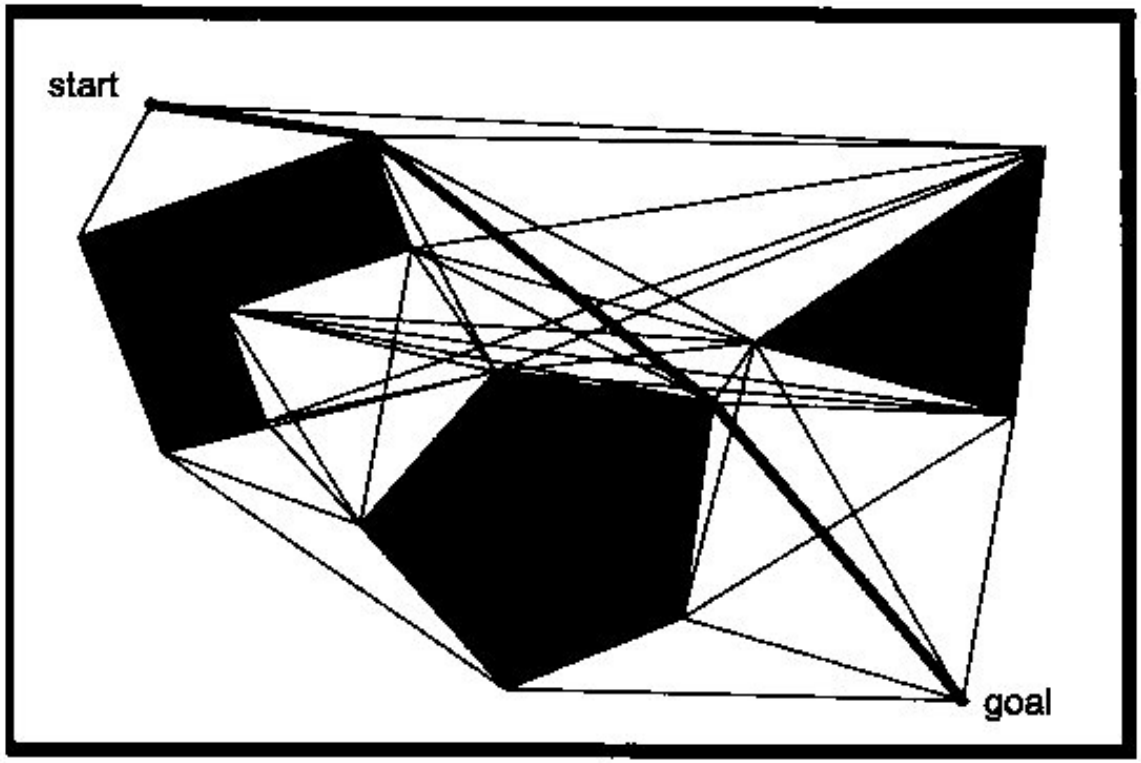
Raw sensor data	Store few locations	Connect by raw motor action
Extract features	Store continuous links	Connect with some metric information
Use sensor model	Make inferences between nodes	Nodes have global position
Convert to spatial data	Continuous representation	Absolute metric

From metric to topological maps

- For planning a route, one popular approach is to convert a metric map to a topological map
- Several different methods:
  - Visibility graph
  - Voronoi diagram
  - Cell decomposition
  - Or treat each empty grid square as a node
- Can then apply standard graph search e.g. A*

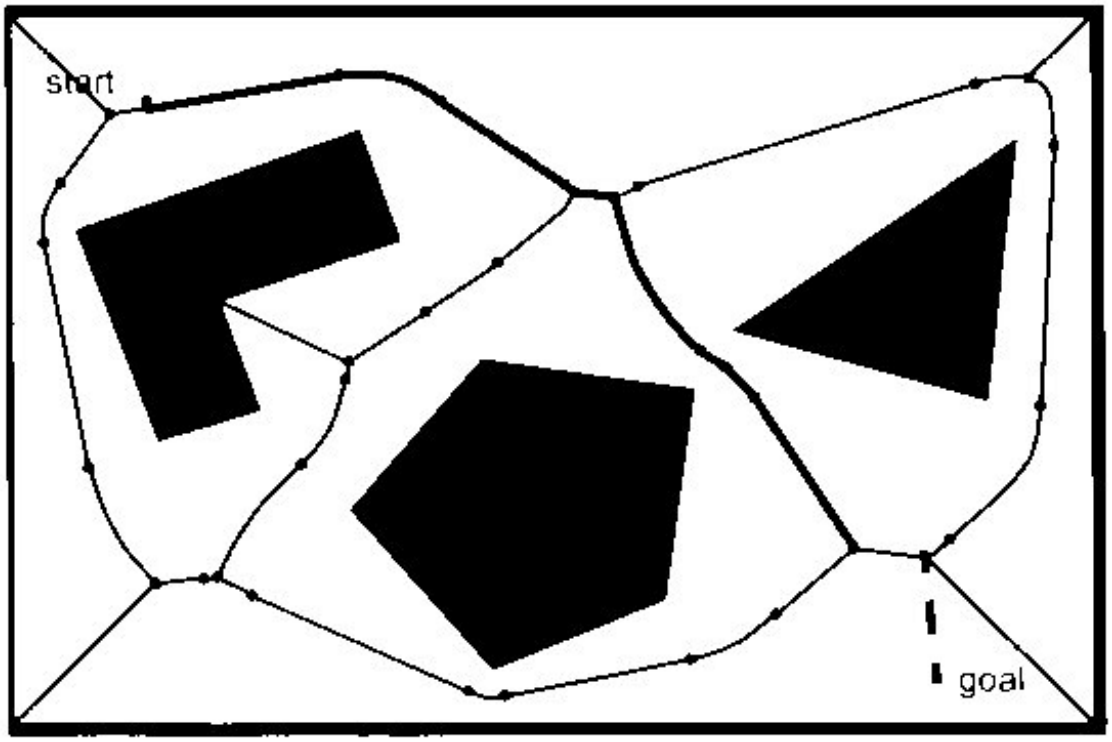
From metric to topological maps

Visibility graph: edges join all vertices that can ‘see’ each other. Defines shortest possible paths.



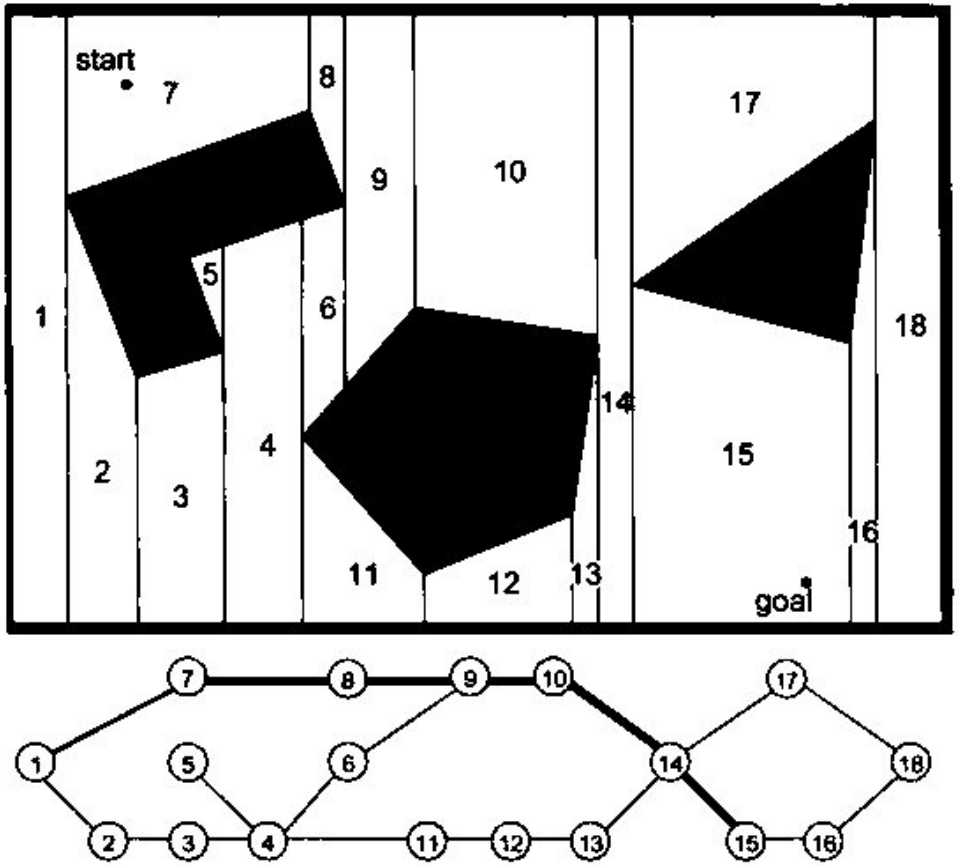
From metric to topological maps

- Voronoi graph – generate edges that are equidistant from obstacles, meeting at vertices.



From metric to topological maps

- Cell decomposition: define free and occupied geometric areas and determine which are adjacent



Path planning

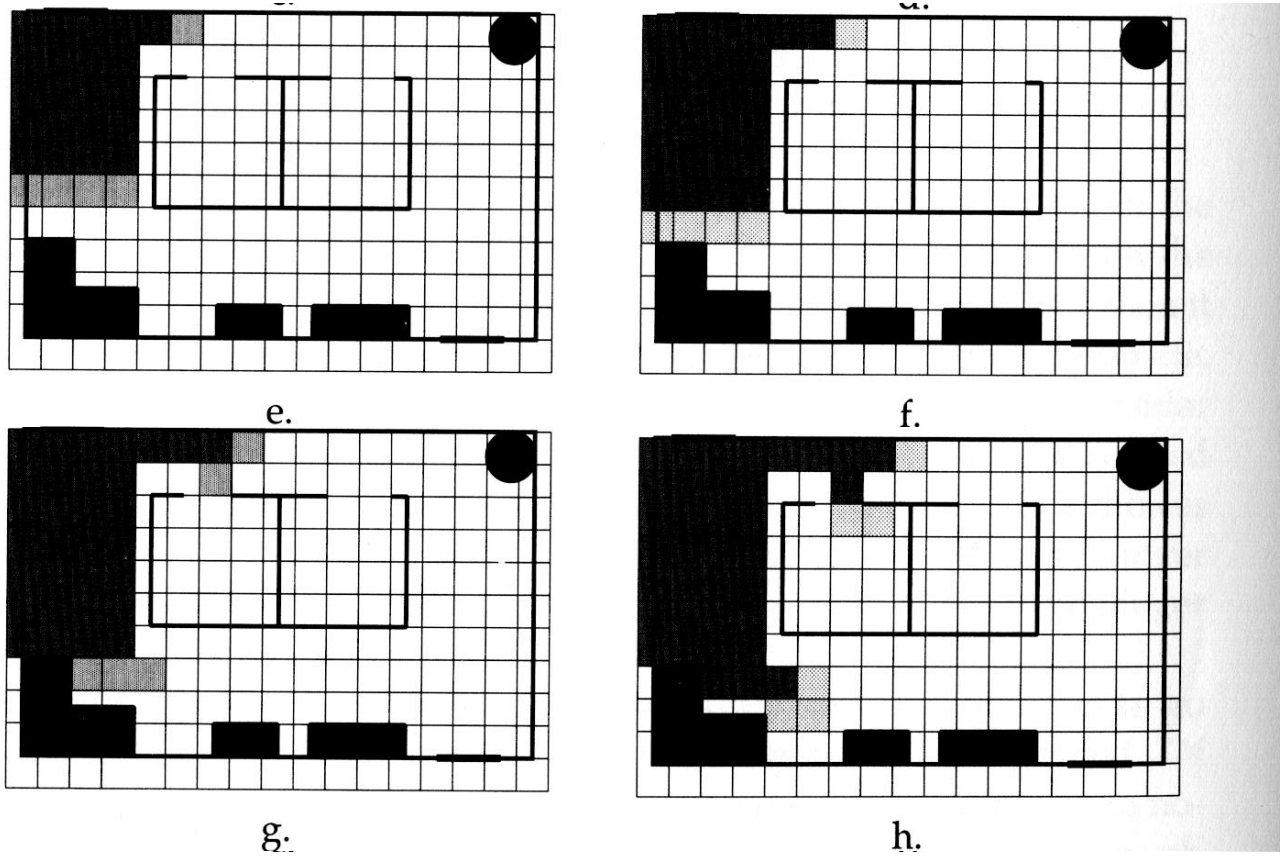
- For a graph with nodes connected by edges

$$\mathbf{f(n)=g(n)+\varepsilon h(n)}$$

- f(n) is the “goodness” of the path via node n
- g(n) is the “cost” of going from the Start to node n
- h(n) is the cost of going from n to the Goal
- c(n,n’) is cost from node n to adjacent node n’

Breadth-first search

If **ε=0**, and c(n,n') is constant for all n (e.g. in grid) then breadth first search will find optimal route.



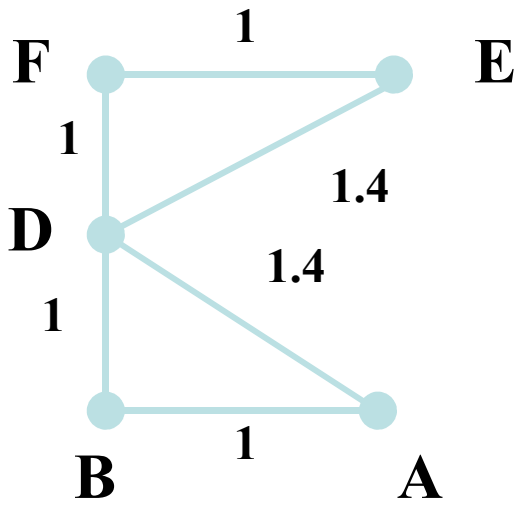
A* Heuristic Function

If **ε=1**, and c(n,n') is not constant, A* is a more efficient search. Like breadth-first except always expand the 'best' (least cost) node first (note same method with **ε=0** is Dijkstra's algorithm)

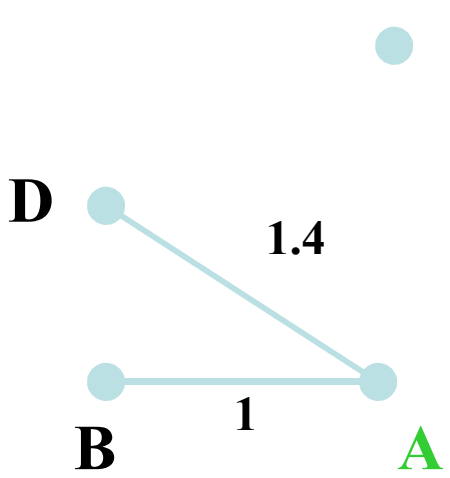
$$\mathbf{f^*(n)=g^*(n)+h^*(n)}$$

- g*(n) is easy: just sum up the path costs to n
- h*(n) is tricky
  - But if began with metric map, may know the *direct* distance between any two nodes, even if not what path is needed to get between them.
  - Thus a minimal estimate of the remaining cost we can use for h*(n) is the direct distance between n and Goal

Example: A to E



- But since you're starting at A and can only look 1 node ahead, this is what you see:



Summary

- Local navigation strategies suffice for some robot tasks
- Local strategies can be linked to form routes
- Routes can be linked to form maps:
  - A map is needed to plan novel routes (exception?)
- Choice of map representation has many consequences:
  - How can it be acquired?
  - How much information must be stored?
  - How can it be used to find novel routes?

References:

Maja J Mataric (1990) “Navigating With a Rat Brain: A Neurobiologically-Inspired Model for Robot Spatial Representation” in *Proceedings, From Animals to Animats: First International Conference on Simulation of Adaptive Behavior (SAB-90)*, J-A. Meyer and S. Wilson, eds., MIT Press, 169-175.

M. J. Milford, G. Wyeth (2008) ”Mapping a Suburb With a Single Camera Using a Biologically Inspired SLAM System,“ *IEEE Transactions on Robotics*, 24: 1038-1052

Chapter 6 of Seigwart, R. and Nourbakhsh, I.R. `Introduction to Autonomous Mobile Robots', 2nd edition, MIT Press 2011