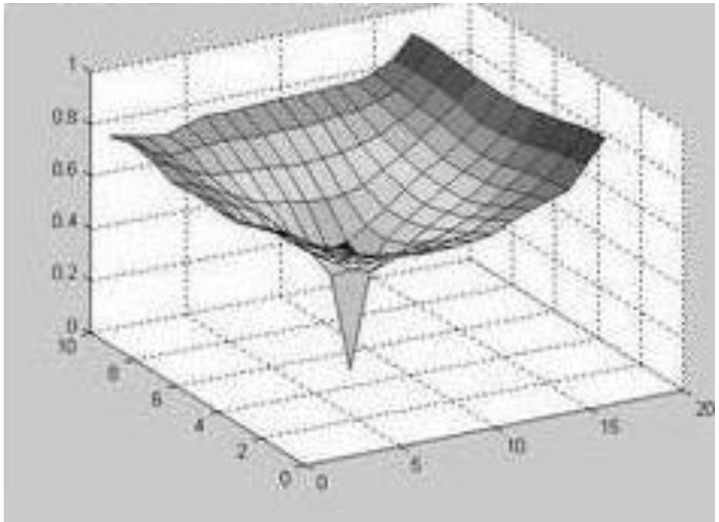
Maps and Planning

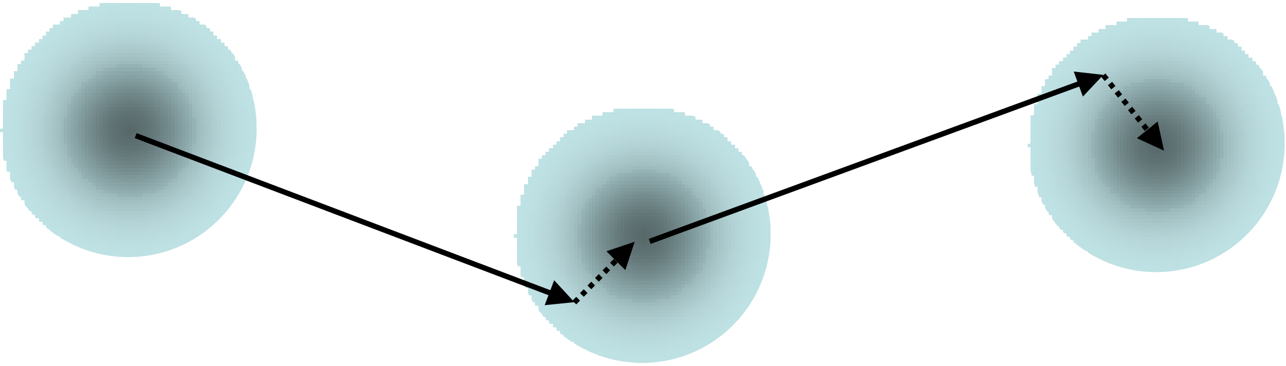
IAR Lecture 8

Barbara Webb

From local strategies to routes to maps

- (From lecture 2) Beacon or memory of home view allows navigation from a surrounding catchment area
- Multiple beacons or memories can be linked together to form routes

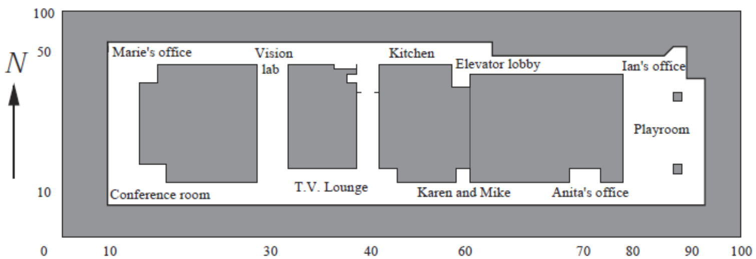


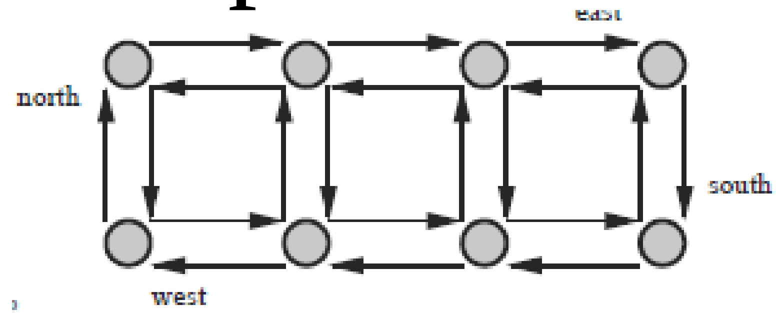


From local strategies to routes to maps

- 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





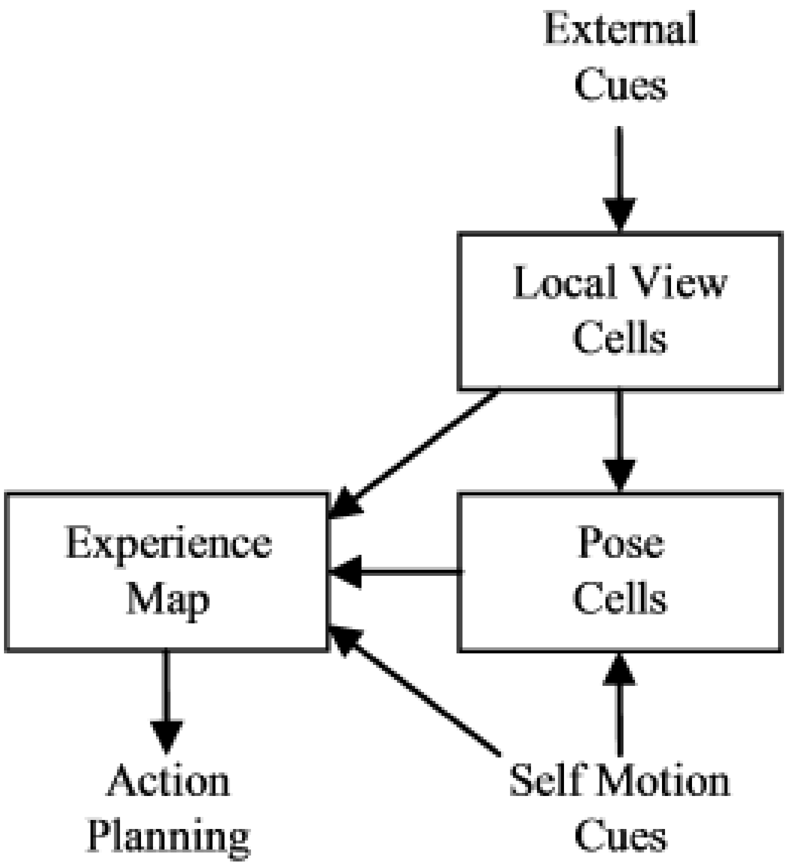
- Represent known locations and the connections between them as nodes and edges in a graph
- The edges could represent simple adjacency, the raw actions needed to get from one node to the next; or direction, distance, path convenience etc.
- Can determine a possible (or even the optimal) route by standard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

Example: RAT-SLAM

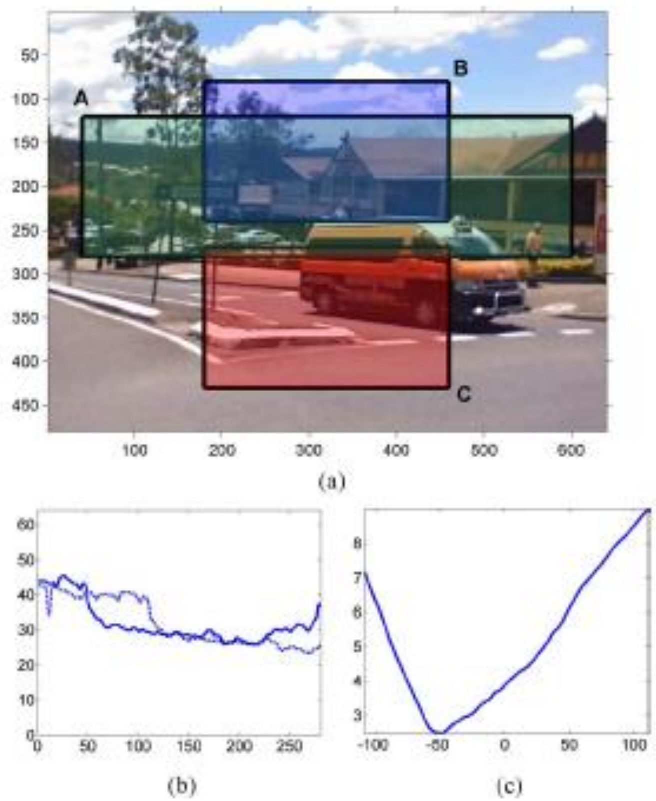
- For each new local view and/or pose, store an ‘experience’ node, linked to the previous node by a transition derived from the self motion (experience nodes are like rat place cells)

- 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:

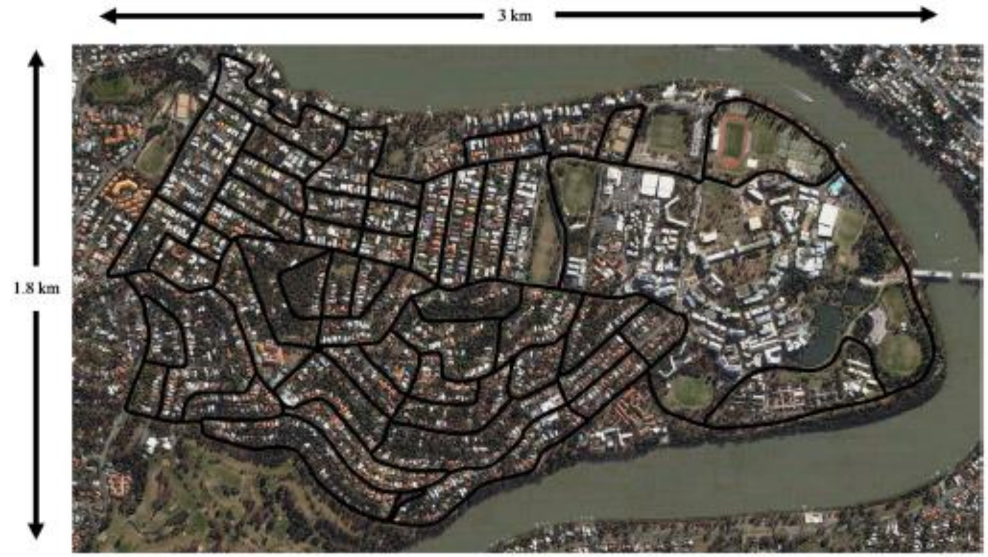
A: Local view: compare to previously stored templates; either recognise or store as new template

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

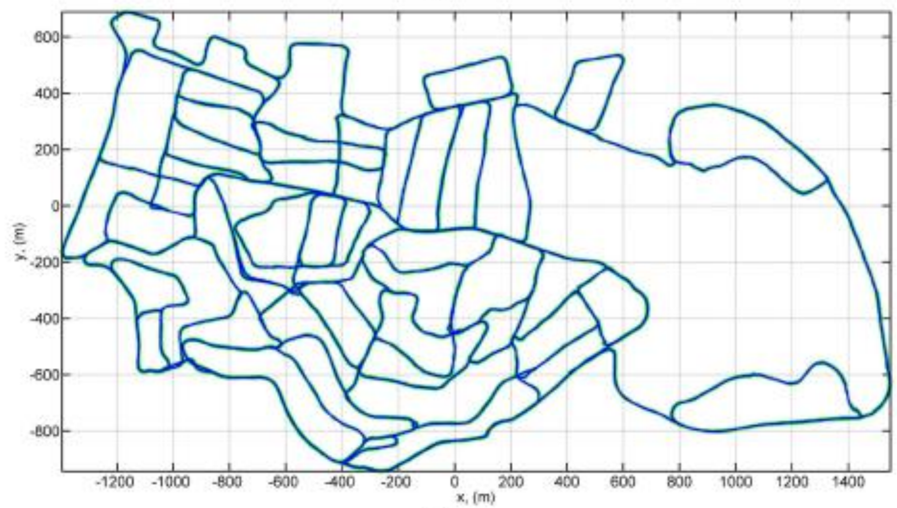
C: Speed estimate: for best rotation, take image difference.

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

RatSLAM Results 2008



(a)

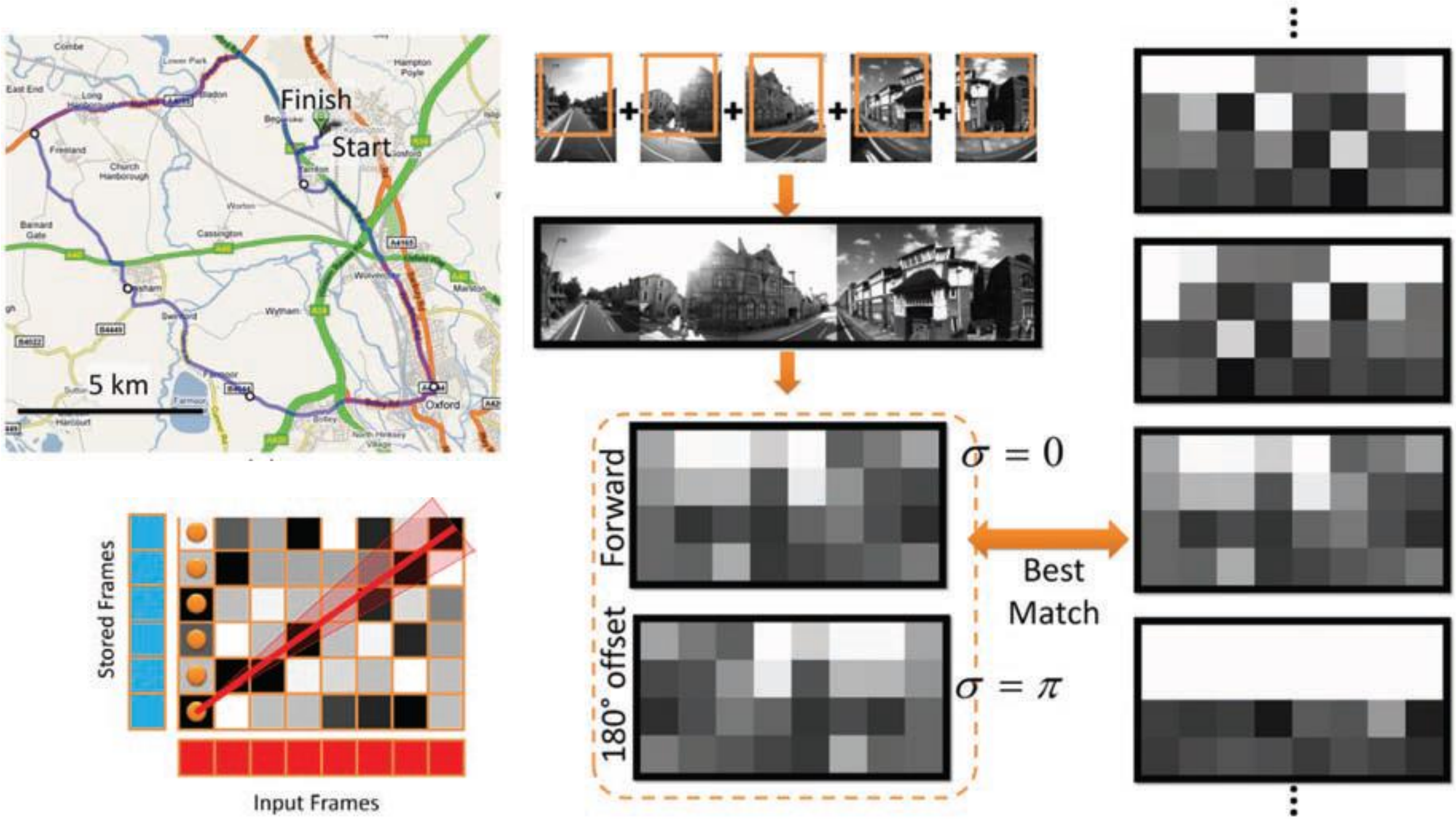


(b)

- 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

Sequence SLAM (Milford 2013)
Improve matching by looking for sequences

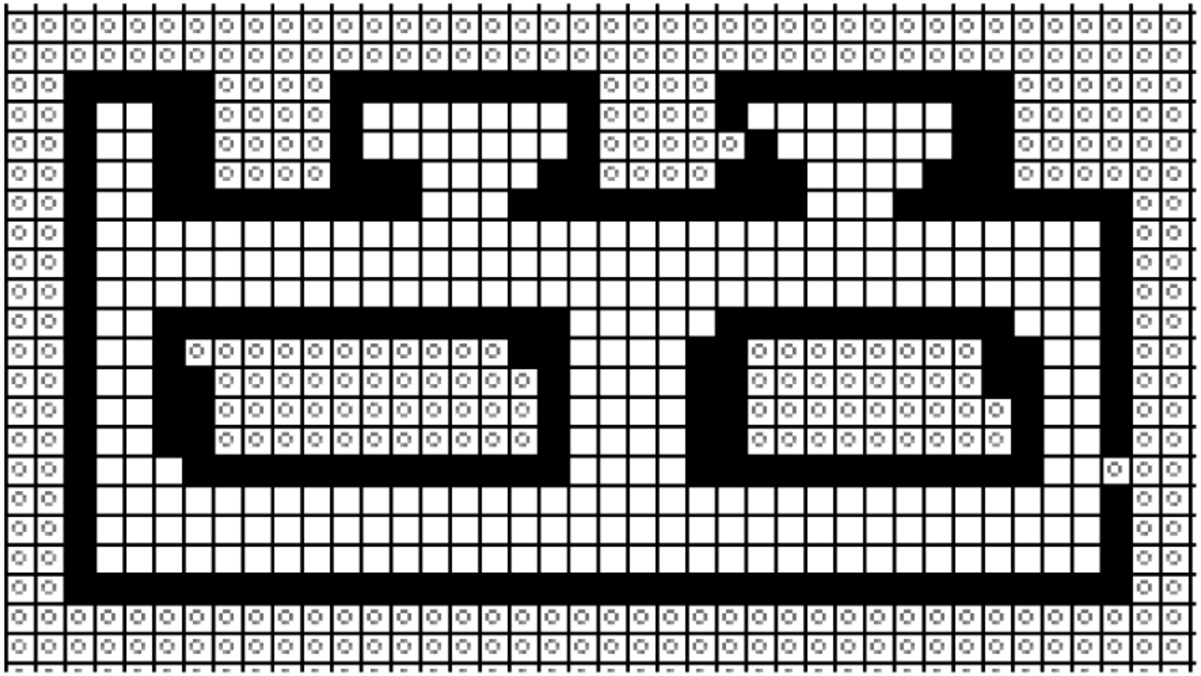


From topological to metric maps

- 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’



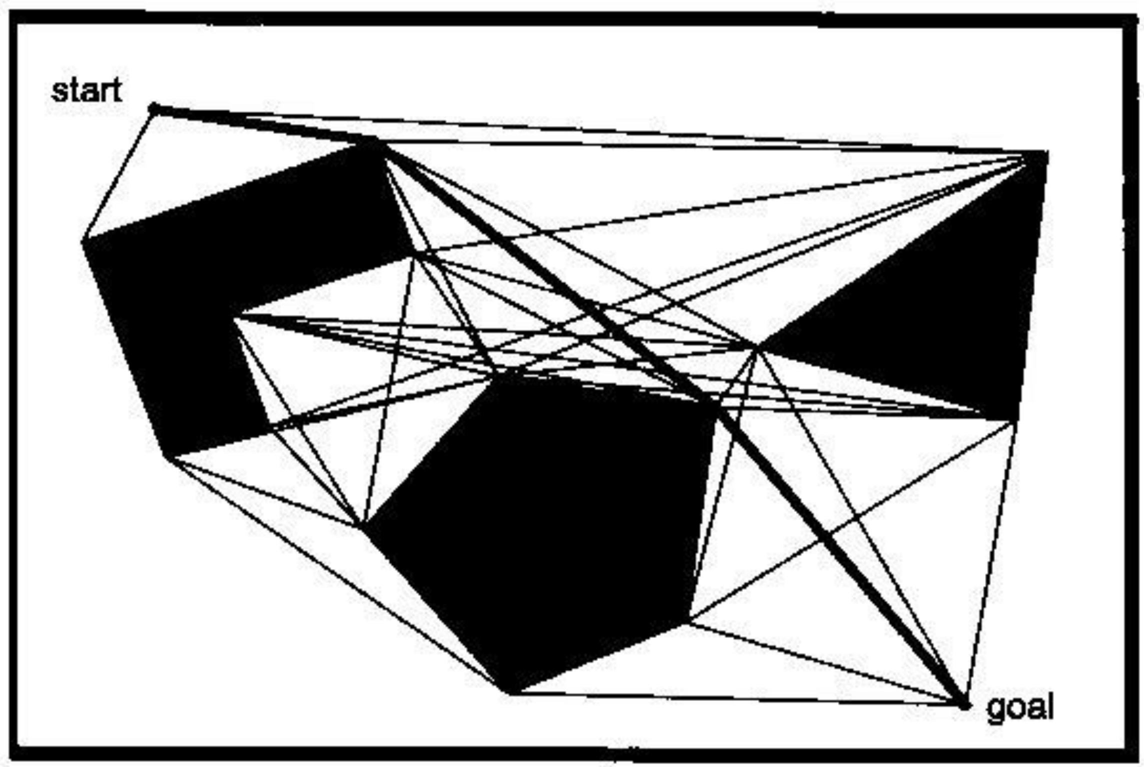
Topological	Raw sensor data	Store few locations	Connect by raw motor action
	Extract features	Store continuous links	Connect with some metric information
	Use sensor model	Make inferences between nodes	Nodes have global position
Metric	Convert to spatial data	Continuous representation	Absolute metric

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*

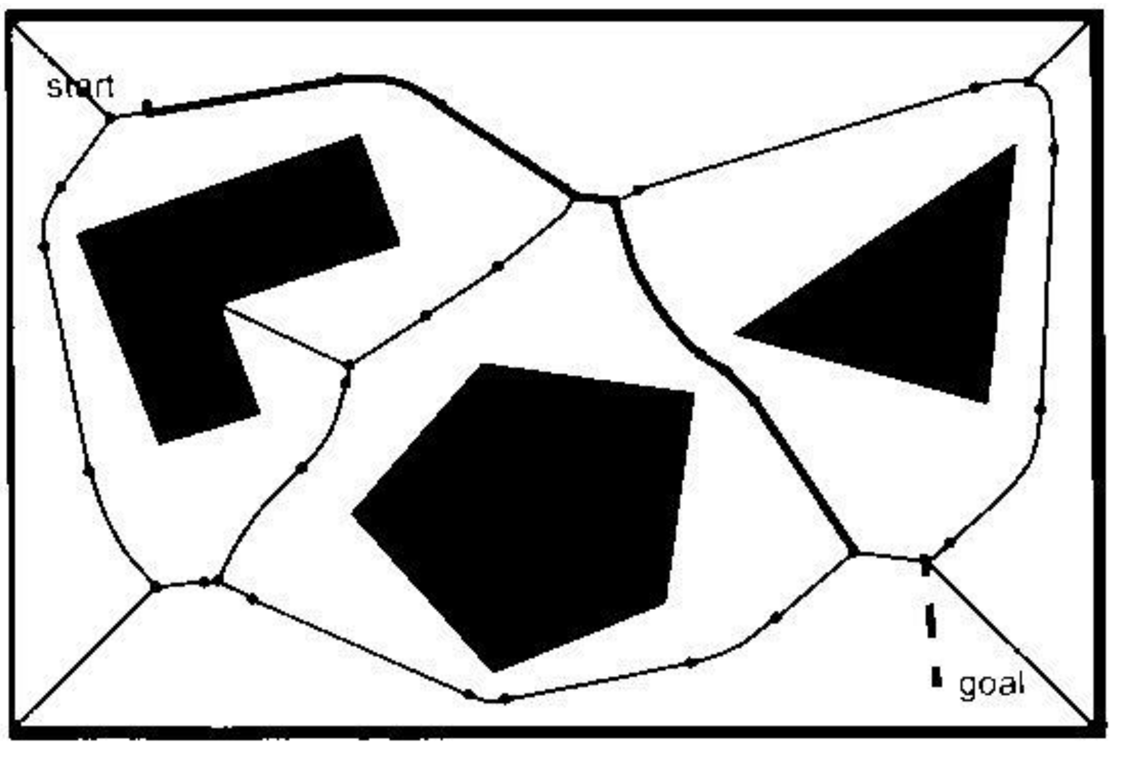
From metric to topological maps

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



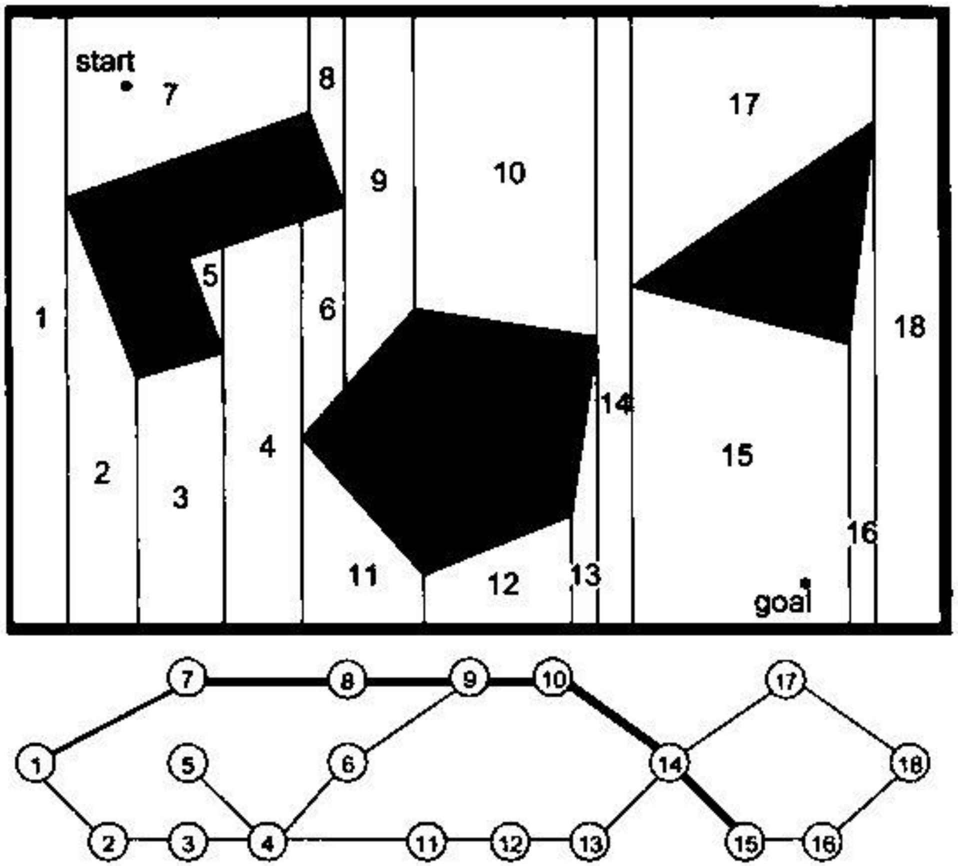
From metric to topological maps

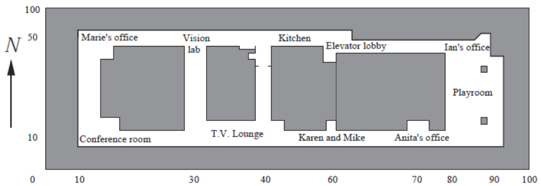
- Voronoi graph – generate edges that are equidistant from obstacles, meeting at vertices.



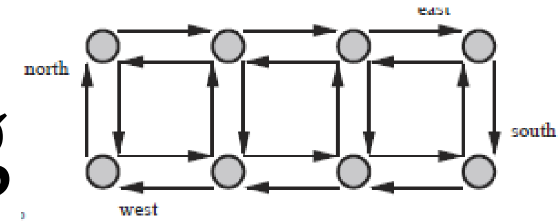
From metric to topological maps

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





Path planning



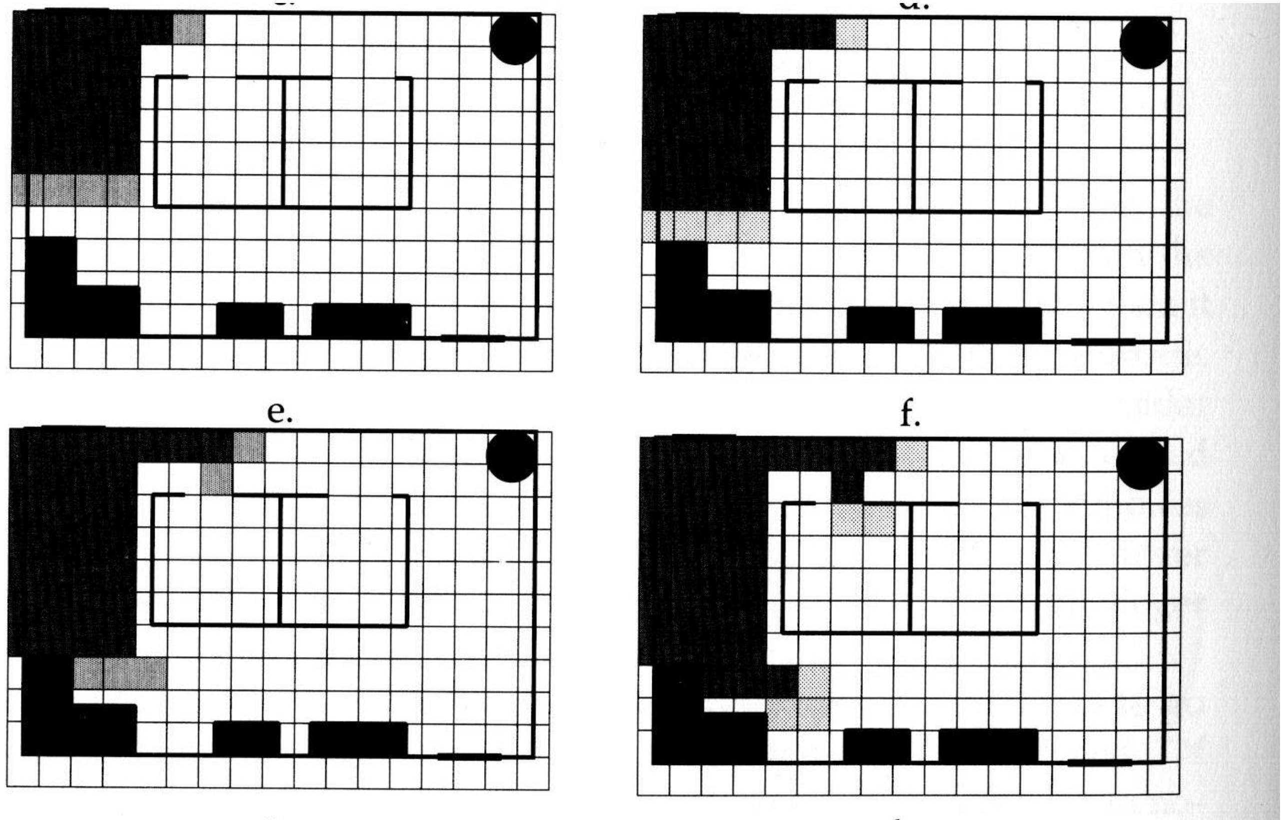
- For a graph with nodes connected by edges

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

- 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
- **ε** is relative weighting of these costs
- Use c(n,n’): 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.



e. f.

g. h.

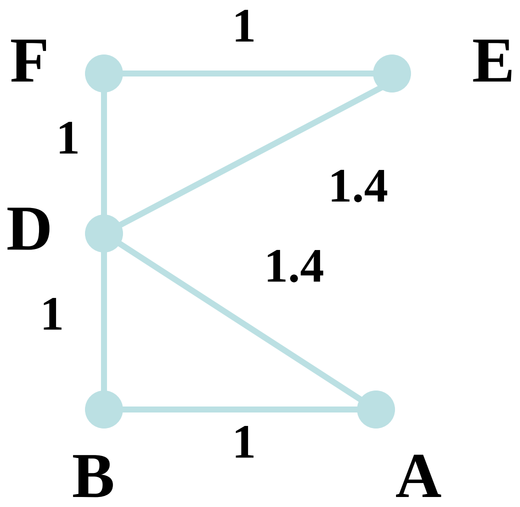
A* Heuristic Function

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)

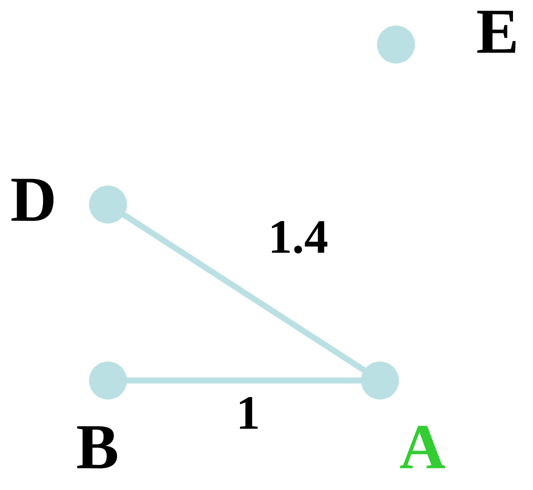
$$\mathbf{f^*(n)=g^*(n)+h^*(n)}$$

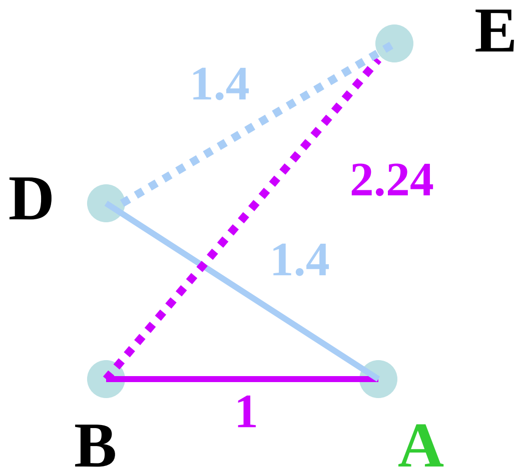
- g*(n) is easy: just sum up the path costs to n
- h*(n) is tricky
  - But if began with metric map, may know the *direct* distance between any two nodes, even if not what path is needed to get between them.
  - Thus a minimal estimate of the remaining cost we can use for h*(n) is the direct distance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

Example: A to E



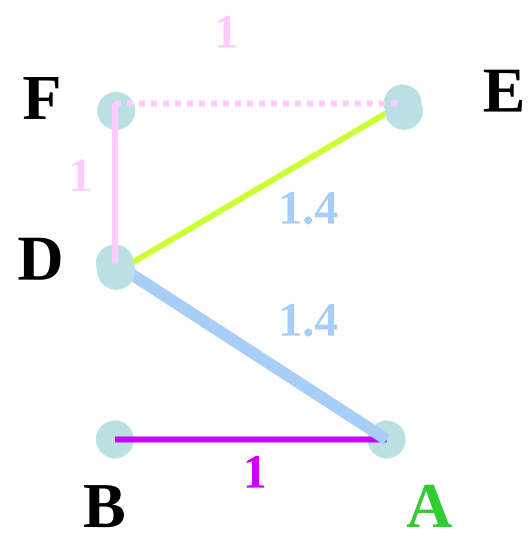
- But since you're starting at A and can only look 1 node ahead, this is what you see:





- Two choices for n: B, D
- Do both
  - f*(B)=1+2.24=3.24
  - f*(D)=1.4+1.4=2.8
- Expand the most plausible path first => A-D-?-E

- A-D-?-E
  - “stand on D”
  - Can see 2 new nodes: F, E
  - f*(F)=(1.4+1)+1=3.4
  - f*(E)=(1.4+1.4)+0=2.8



- Three paths
  - A-B-?-E >= 3.24
  - A-D-E = 2.8
  - A-D-F-?-E >=3.4

- A-D-E is the winner!
  - Don’t have to look farther because expanded the shortest first, others couldn’t possibly do better without having negative distances, violations of laws of geometry…

Planning as optimisation

- The problem of planning can be formulated as an optimisation problem: in state *x* what is best action *u*?
  - Robot has goals, but actions have costs
  - Express both as a single ‘pay-off’ function, e.g.,

$$r(x,u) = \begin{cases} +100 & \textit{If reach the desired state} \\ -1 & \textit{Otherwise, i.e., cost for each time step} \end{cases}$$

Aim is to maximise the cumulative expected pay-off:

$$R_T = E\left[\sum_{\tau=1}^{T} \gamma^{\tau} r_{t+\tau}\right]$$

Where $0<\gamma<1$ is a discount factor, making distant reward less attractive.

Optimal Policy

- 1-step optimal policy:

$$\pi_1(x) = \operatorname*{argmax}_u \; r(x,u)$$

- Value function of 1-step optimal policy:

$$V_1(x) = \gamma \max_u \; r(x,u)$$

- T-step optimal policy is defined recursively:

$$\pi_T(x) = \operatorname*{argmax}_u \left[r(x,u) + \sum_{i=1}^{N} V_{T-1}(x_i)\, p(x_i \mid u, x) \right]$$

N is number of possible states

Value of the ***next*** *state*

State transition probabilities

- In theory this can sometimes be solved; in practice usually obtain the value function by iteration till the approximation converges

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:

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

M. J. Milford (2013) Vision-based place recognition: how low can you go? The International Journal of Robotics Research 32 (7), 766-789

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