6.141: Robotics systems and science Lecture 8: Control Architectures Motion Planning

Lecture Notes Prepared by Daniela Rus

EECS/MIT

Spring 2011

Thanks to Rod Brooks, Vijay Kumar

Reading: Chapter 3, and Craig: Robotics

http://courses.csail.mit.edu/6.141/

Challenge: Build a Shelter on Mars

Last lecture block we saw

- Camera as a sensor
- Software engineering and Carmen

Today

- Robot control architectures
- Deliberative control: motion planning
- Applications: industrial assembly, exploration, drug design
- Reading: chapter 6

Controlling in the large

- We have seen feedback control
- How do we put together multiple feedback controllers?
  - in what order?
  - with what priority?
- *How do we generate reliable and correct robot behavior?*

Control Architecture

- A control architecture provides a set of principles for organizing a robot (software) control system.
- Like in computer architecture, it specifies building blocks
- It provides:
  - structure
  - constraints

Control Architecture Types

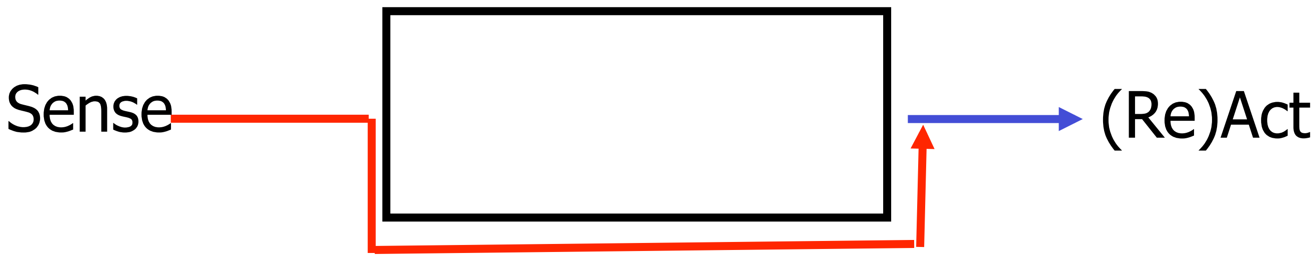
- Deliberative control
- Reactive control
- Hybrid control
- Behavior-based control

Deliberative Architecture

- Maps, lots of state
- Look-ahead

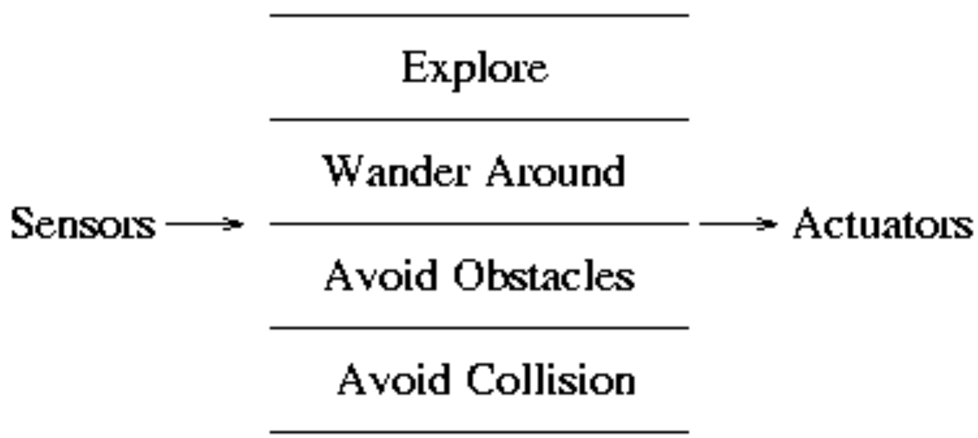
Reactive Architecture

- No maps, no state
- No look ahead



Behavior-based Architecture

- Some state
- Look ahead only while acting
- Reactive + state



Hybrid architectures

- State
- Look ahead but react
- Combines long and short time scales

Criteria For Selection

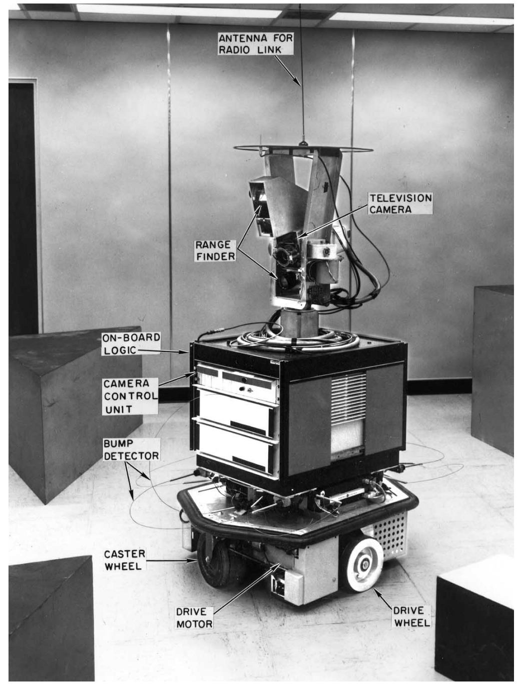
	deliberative	reactive	behavior
Task and environment			
Run-time constraints			
Correctness/ Completeness			
Hardware			

Motion Planning

How do we command the robot to move from A to B despite complications?

Complications: error in maps, sensing, control, unexpected obstacles, etc.

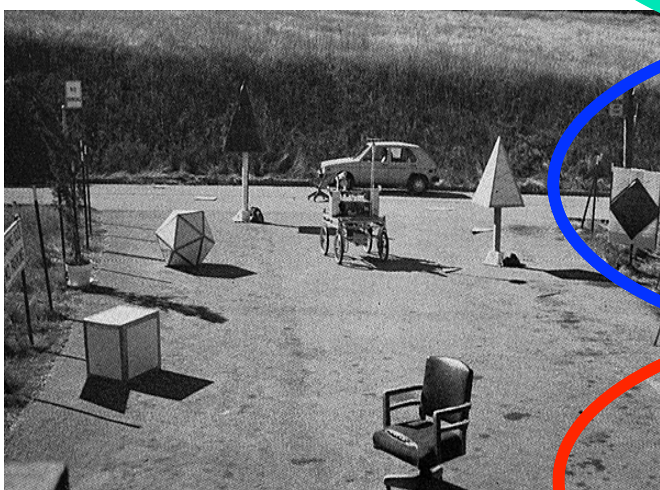
Spatial Planning: Shakey and Stanford Cart (1969)

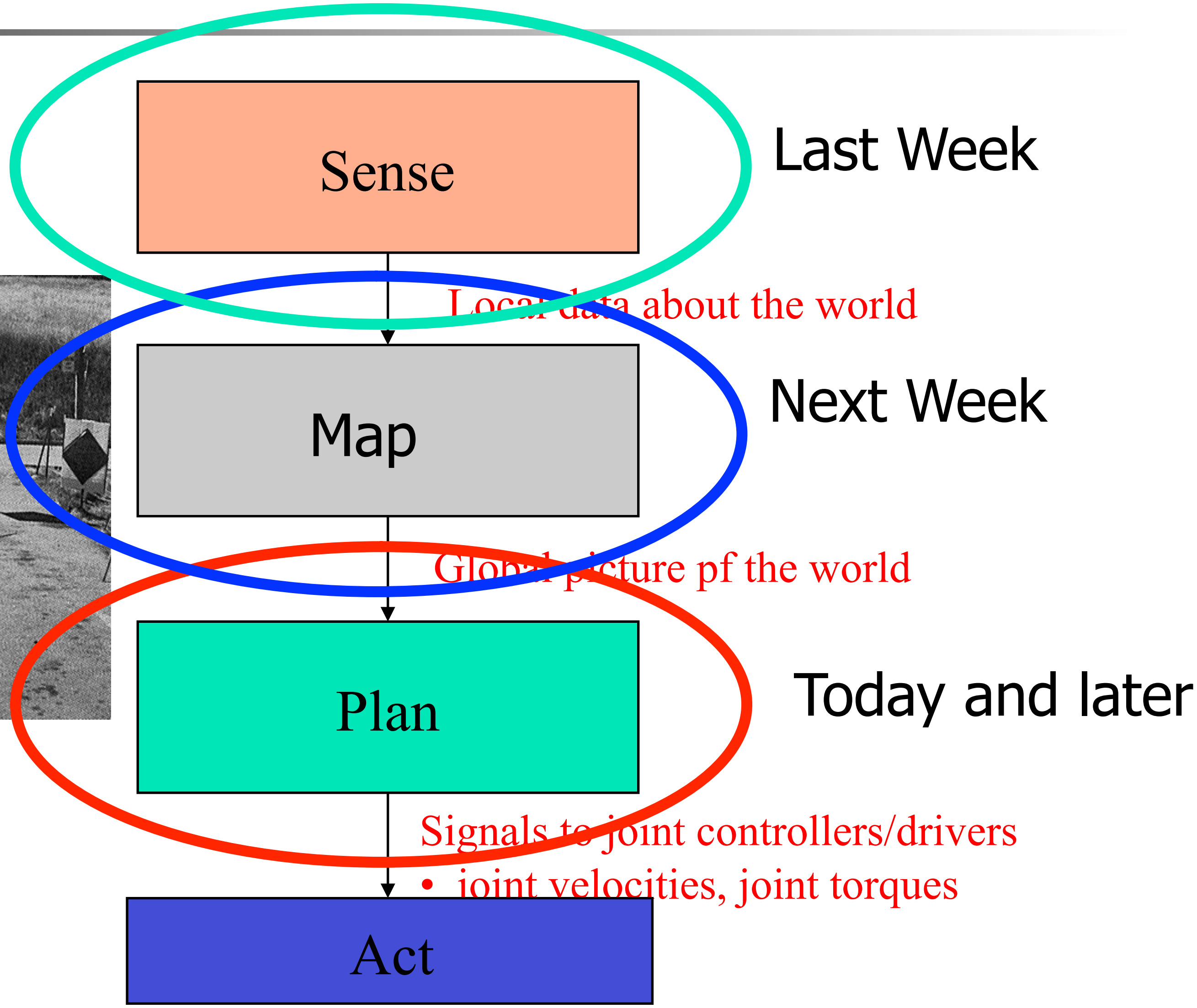


TV camera
Triangulating range finger
Bump sensors
DEC PDP-10, PDP-15 via radio
(192K 36-bit)

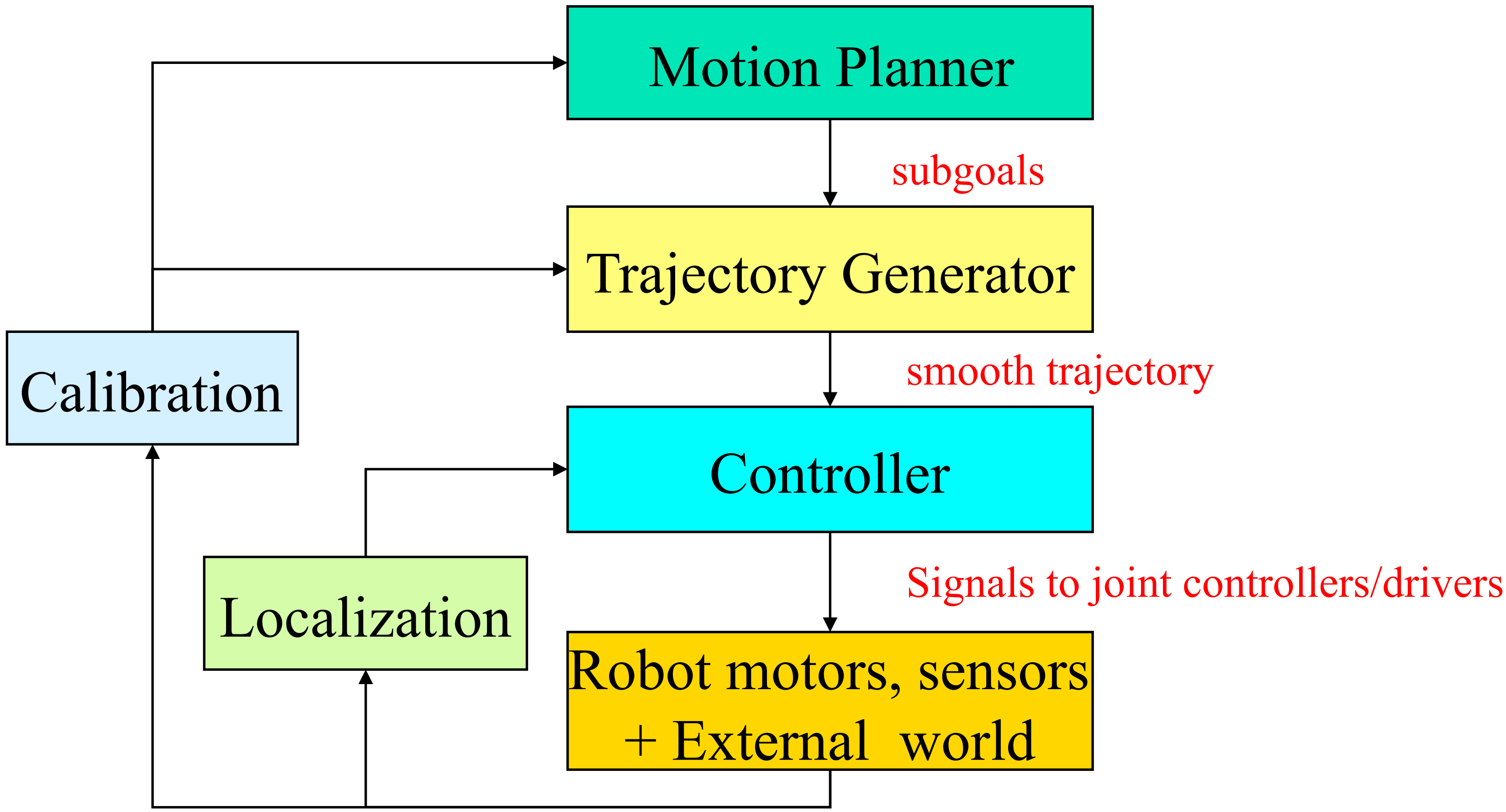
15 mins processing for video planning per meter of travel

Deliberative Architecture

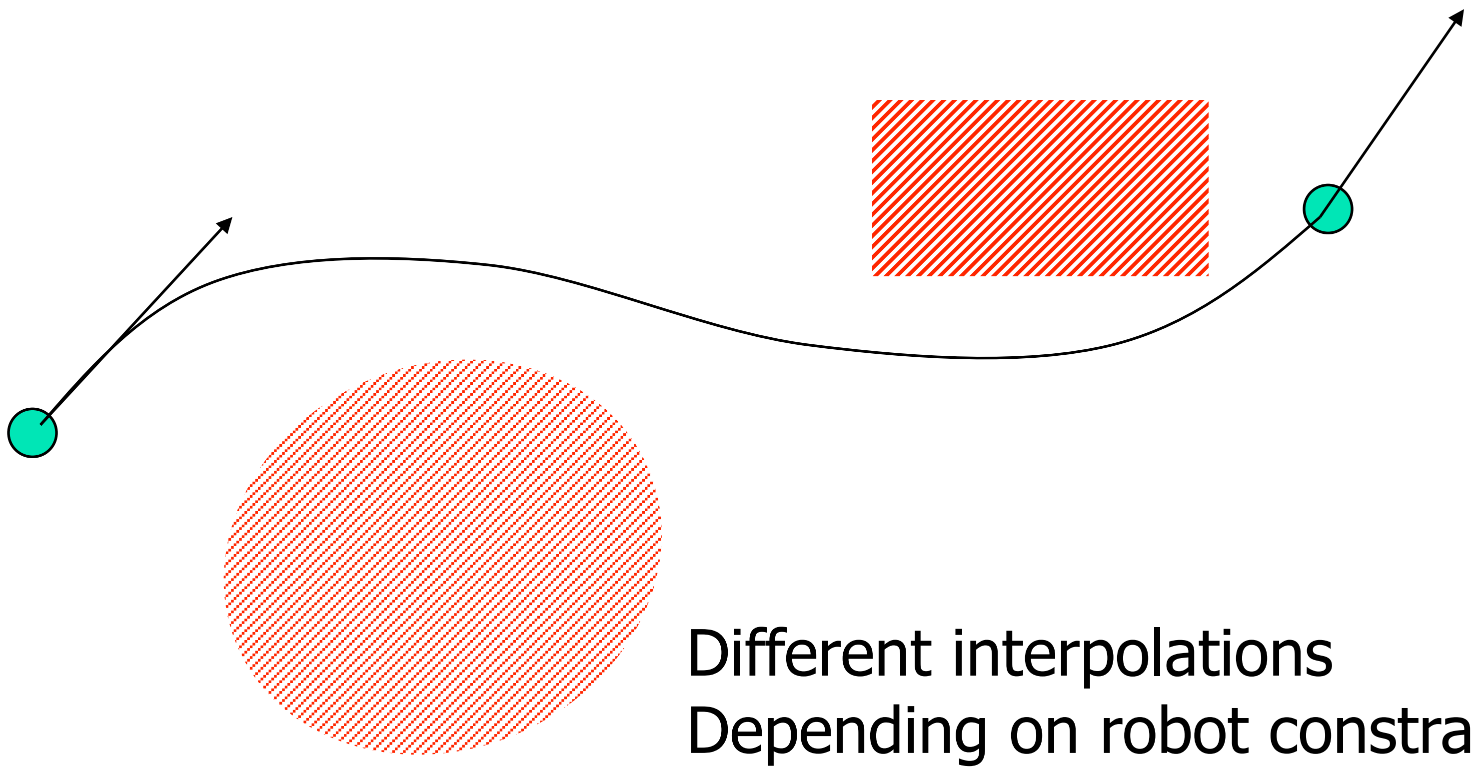




Motion Planning



Trajectory generation from waypoints



Different interpolations
Depending on robot constraints

Motion Planning

- Known Environments (Model)

- Unknown Environments (No Model)

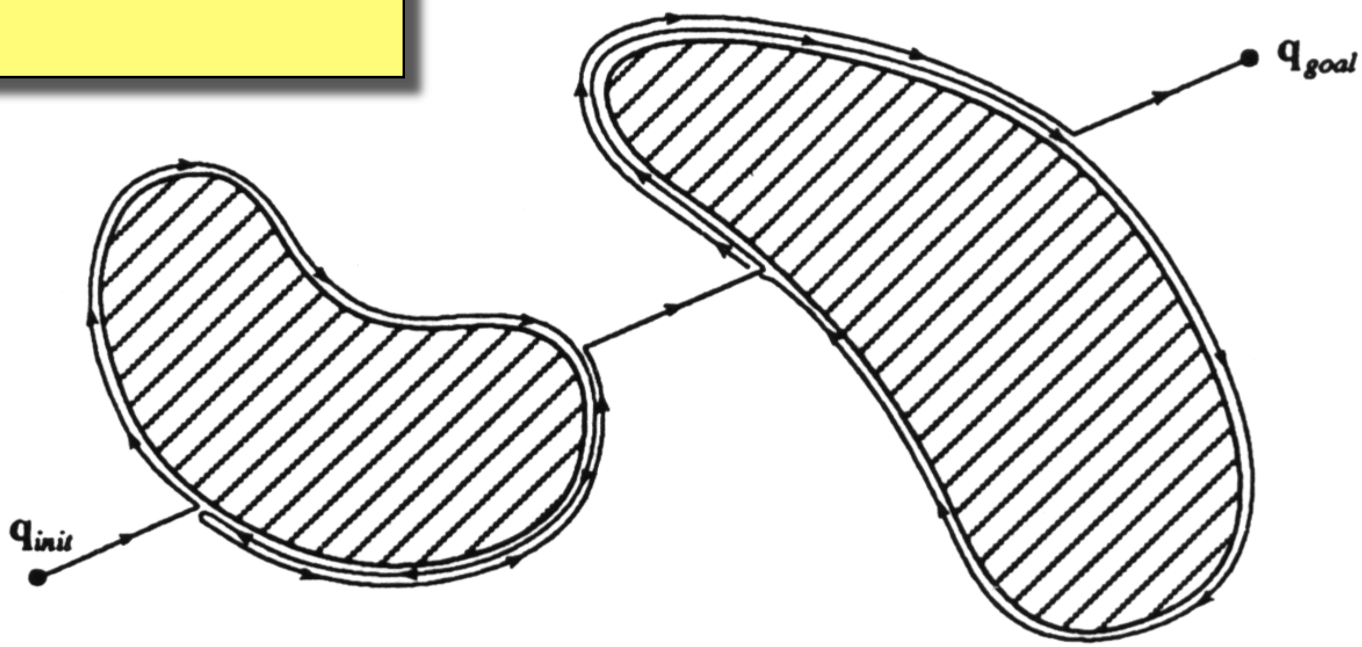
OFFLINE ALGORITHMS

ONLINE ALGORITHMS

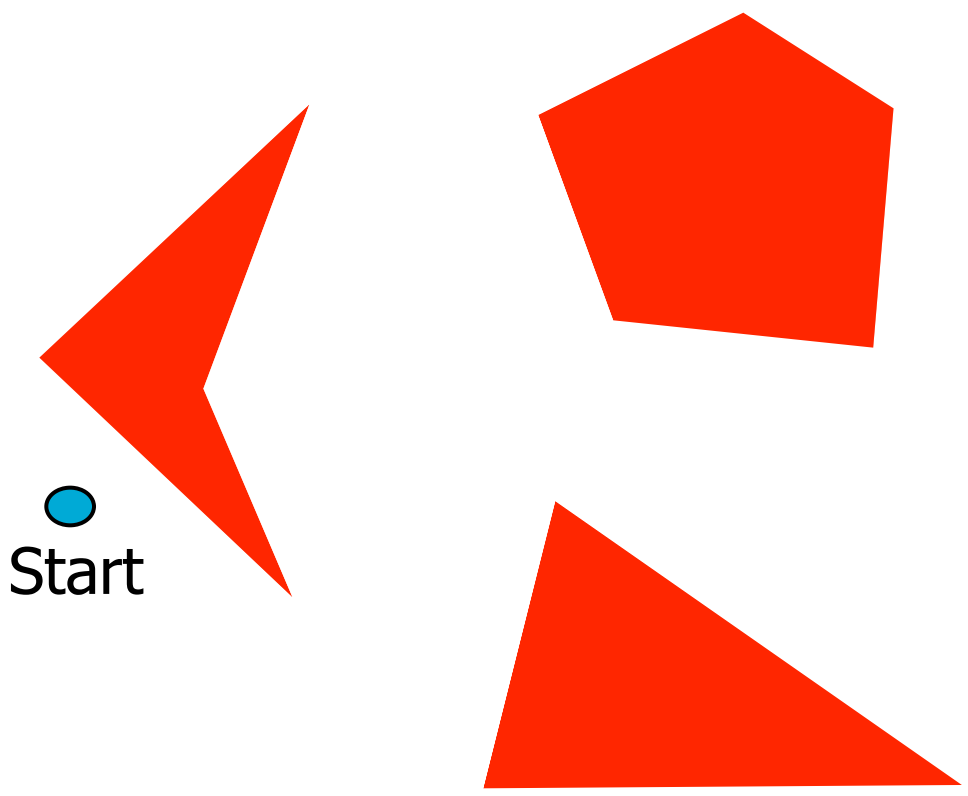
Example: how do we find a bridge in the fog?

Online Motion Planning

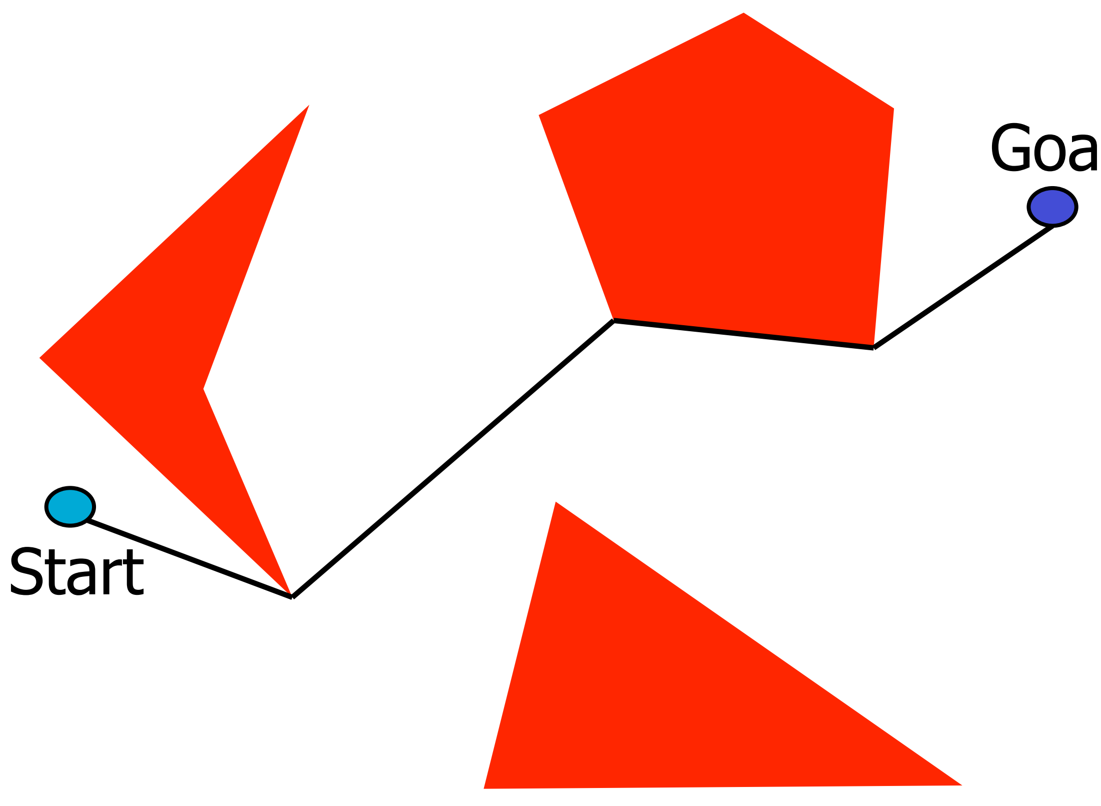
Always finds a path
(if it exists)



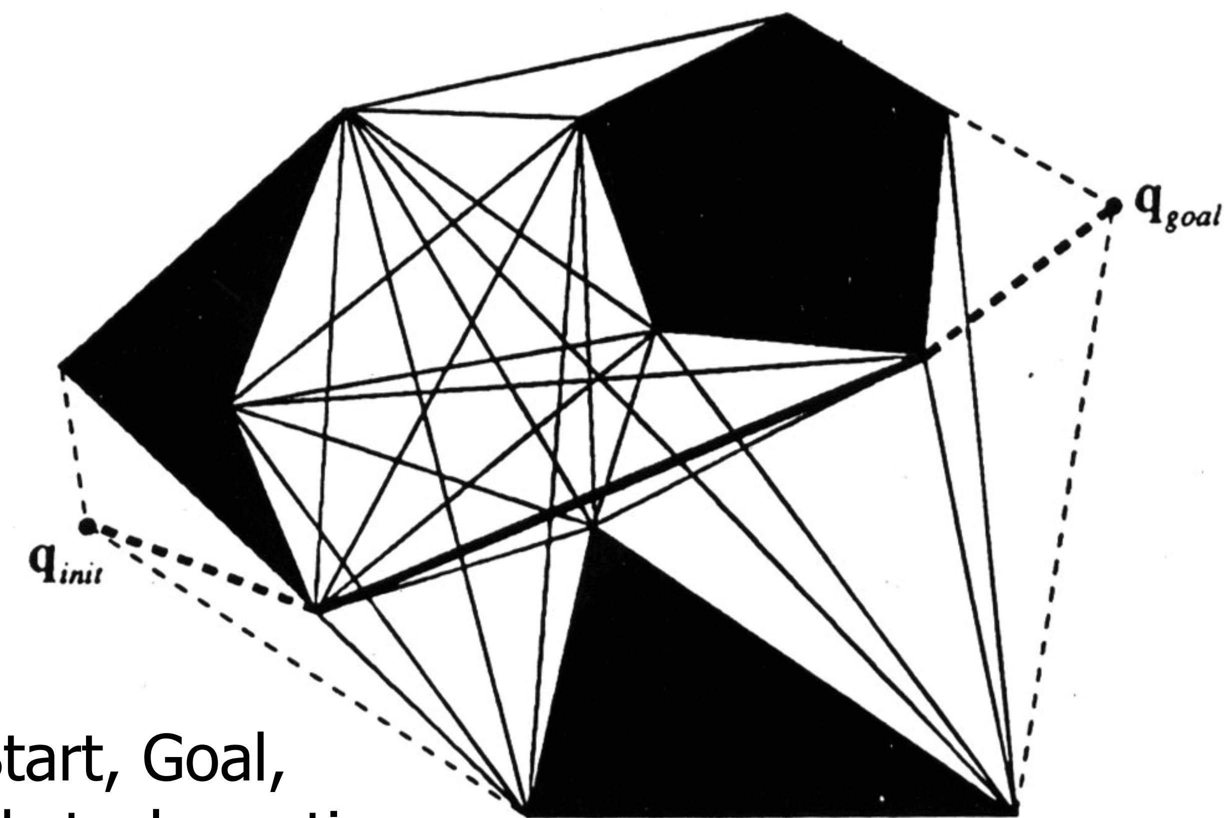
Off-line Motion Planning



Off-line Motion Planning



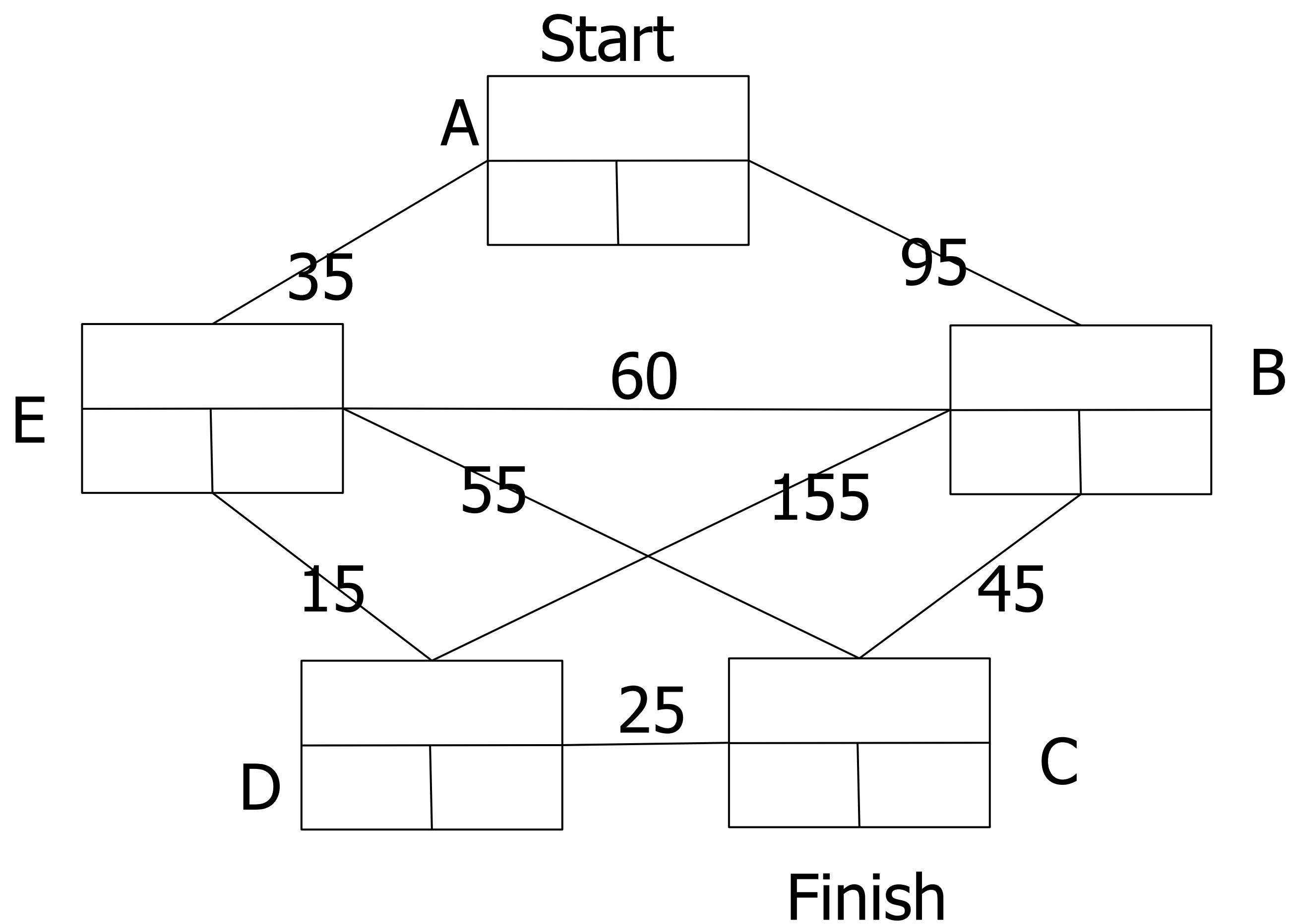
Visibility Graphs



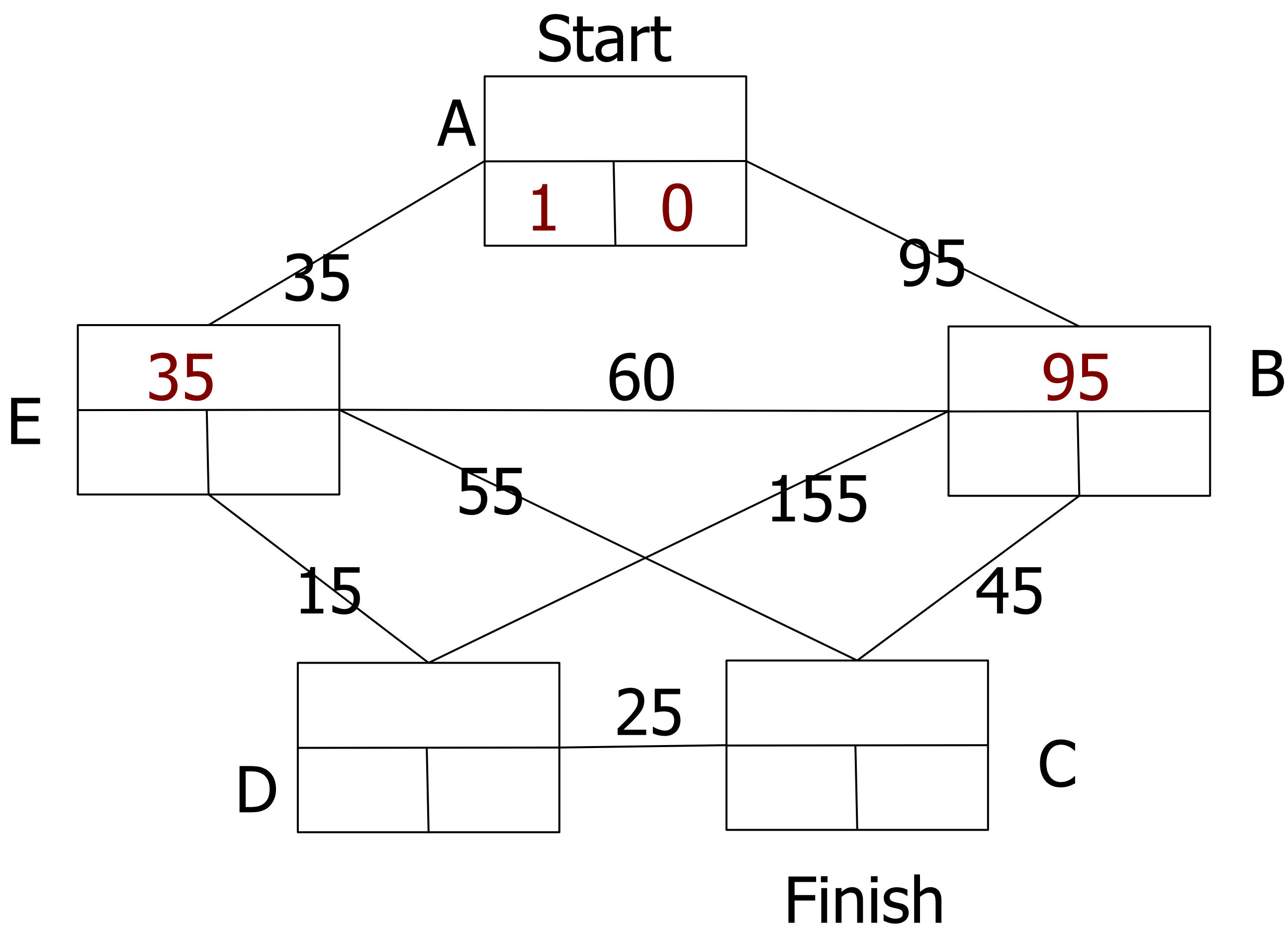
Vertices: Start, Goal,
obstacle vertices

Edges: all combinations (v_i, v_j) that do not intersect any obstacle

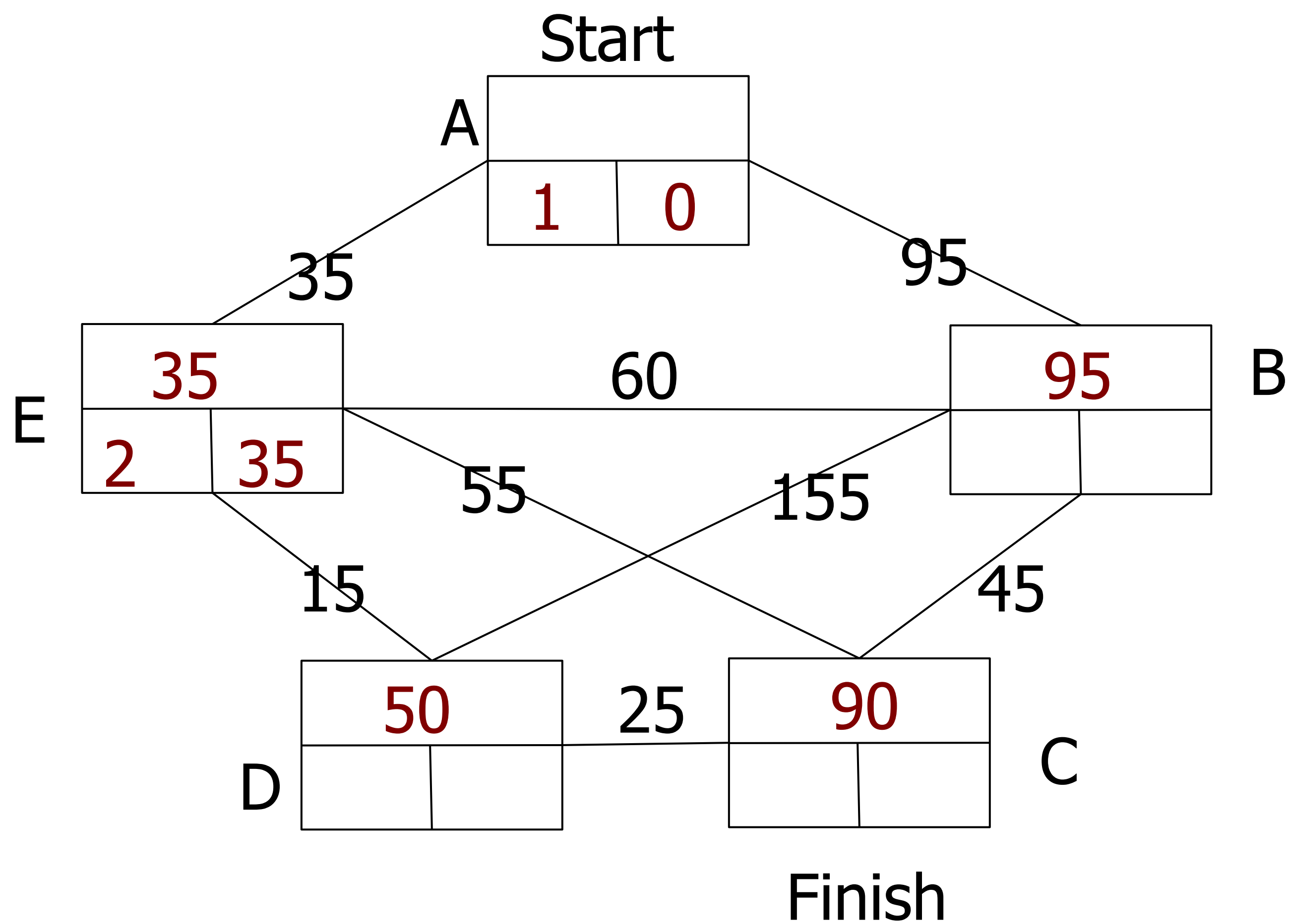
Shortest path



Shortest path

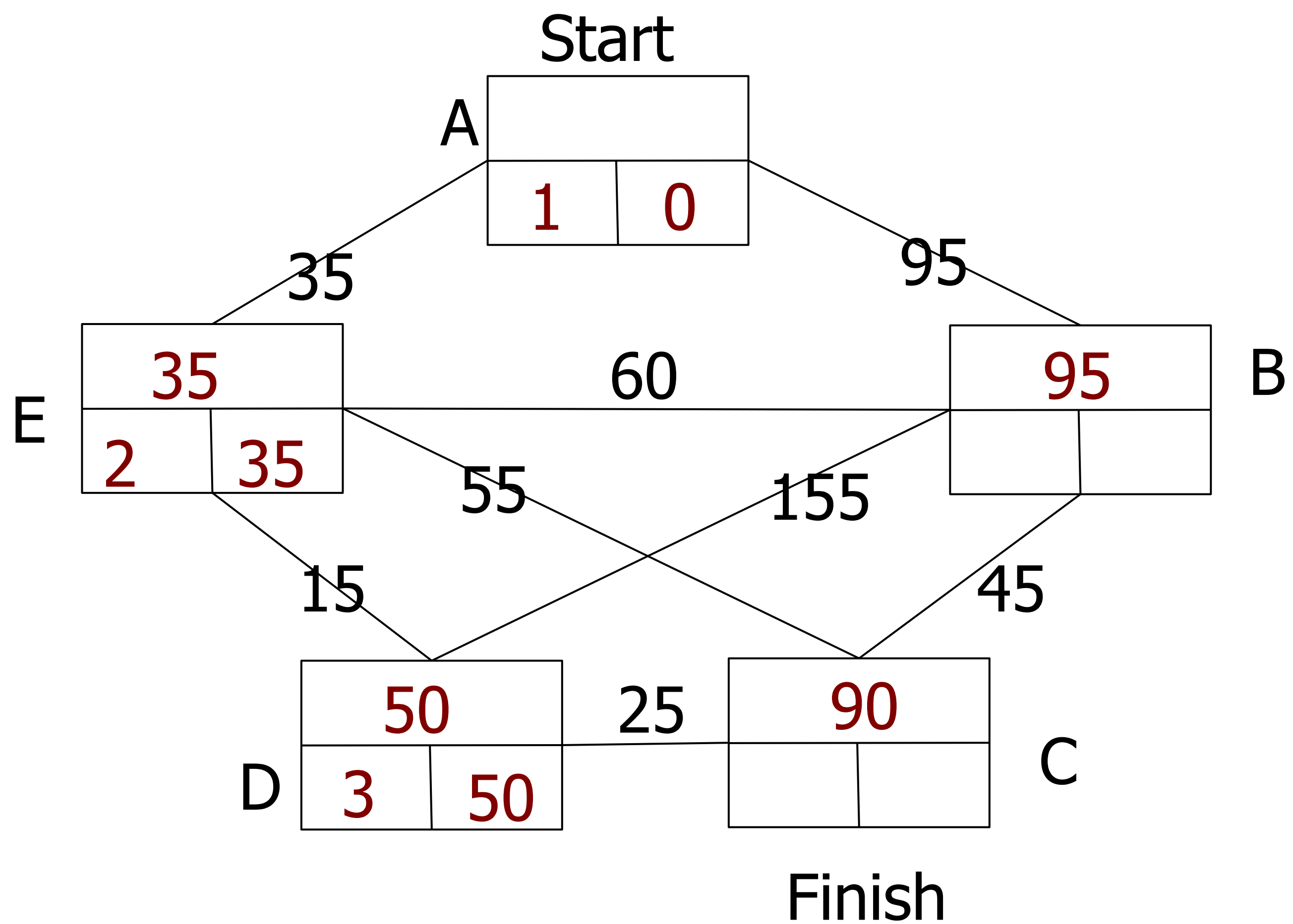


Shortest path



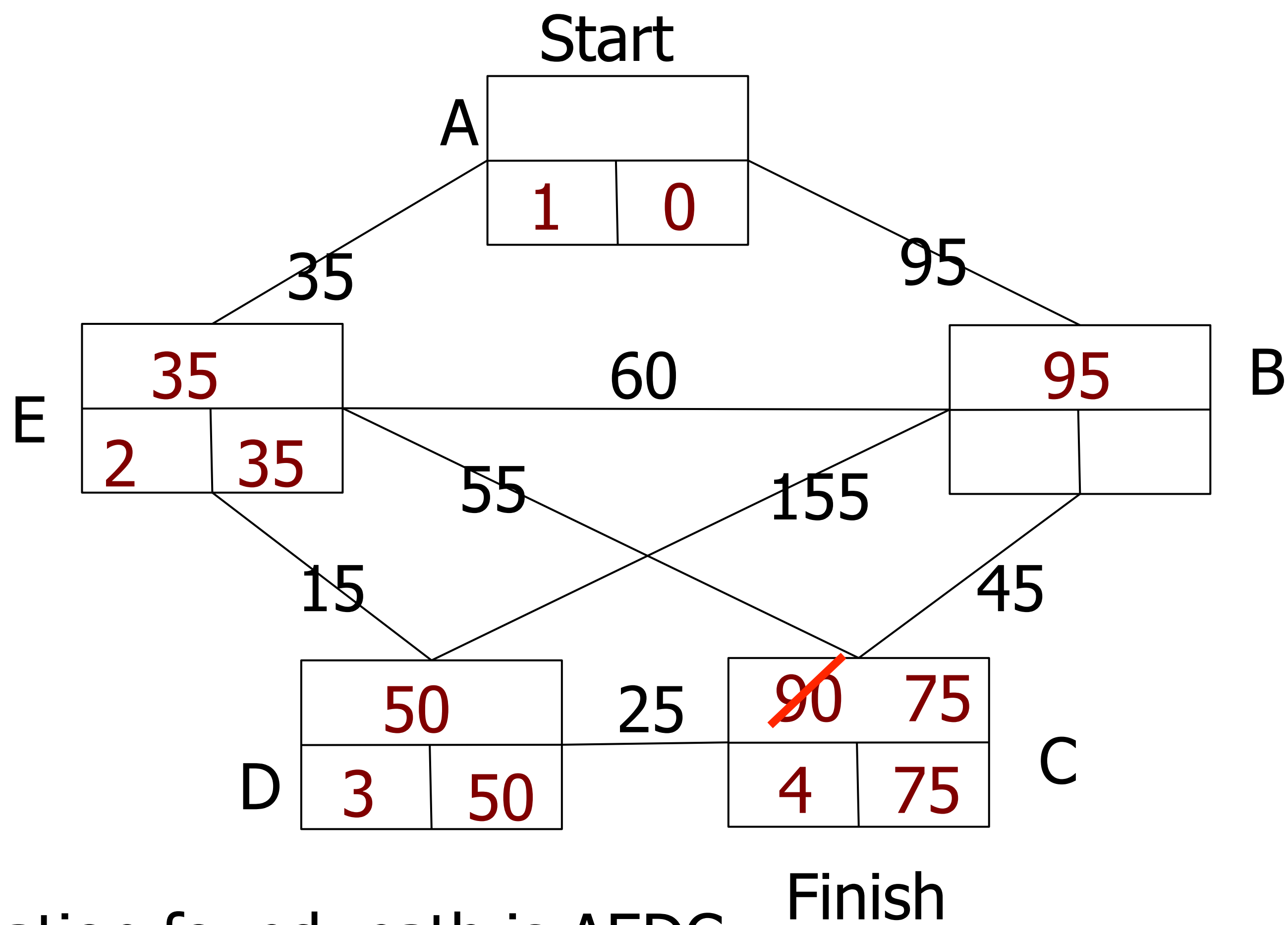
Find node with smallest temporary value; label neighbors

Shortest path



Find node with smallest temporary value; label neighbors

Shortest path

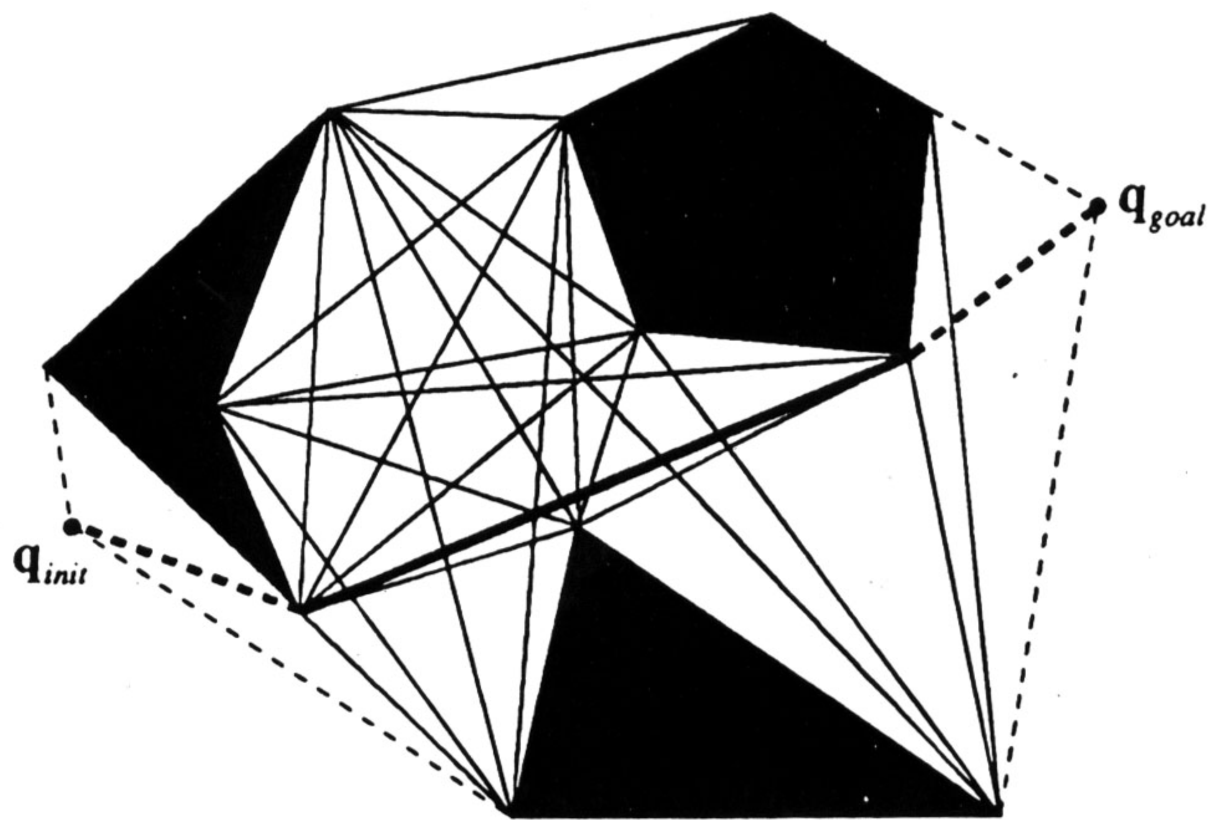


Destination found, path is AEDC

Search Path: Dijkstra's Algorithm

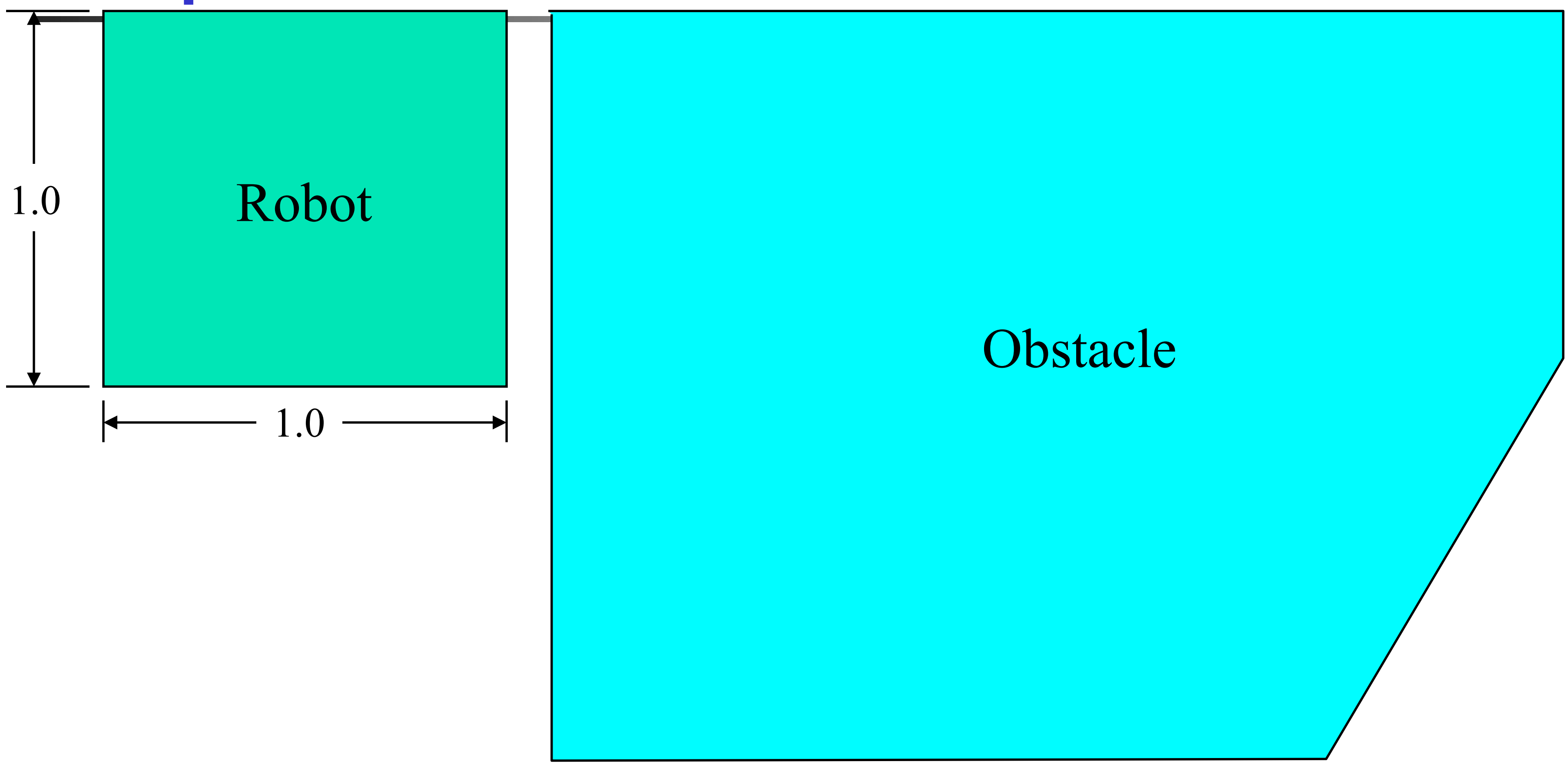
```
function Dijkstra(G, w, s)
    for each vertex v in V[G]                        // Initializations
        d[v] := infinity
        previous[v] := undefined
    d[s] := 0
    S := empty set
    Q := set of all vertices
    while Q is not an empty set                      // The algorithm itself
        u := Extract_Min(Q)                          // O(n) for linked lists; Fib. Heaps?
        S := S union {u}
        for each edge (u,v) outgoing from u
            if d[v] > d[u] + w(u,v)                  // Relax (u,v)
                d[v] := d[u] + w(u,v)
                previous[v] := u
```

Visibility Graphs Summary

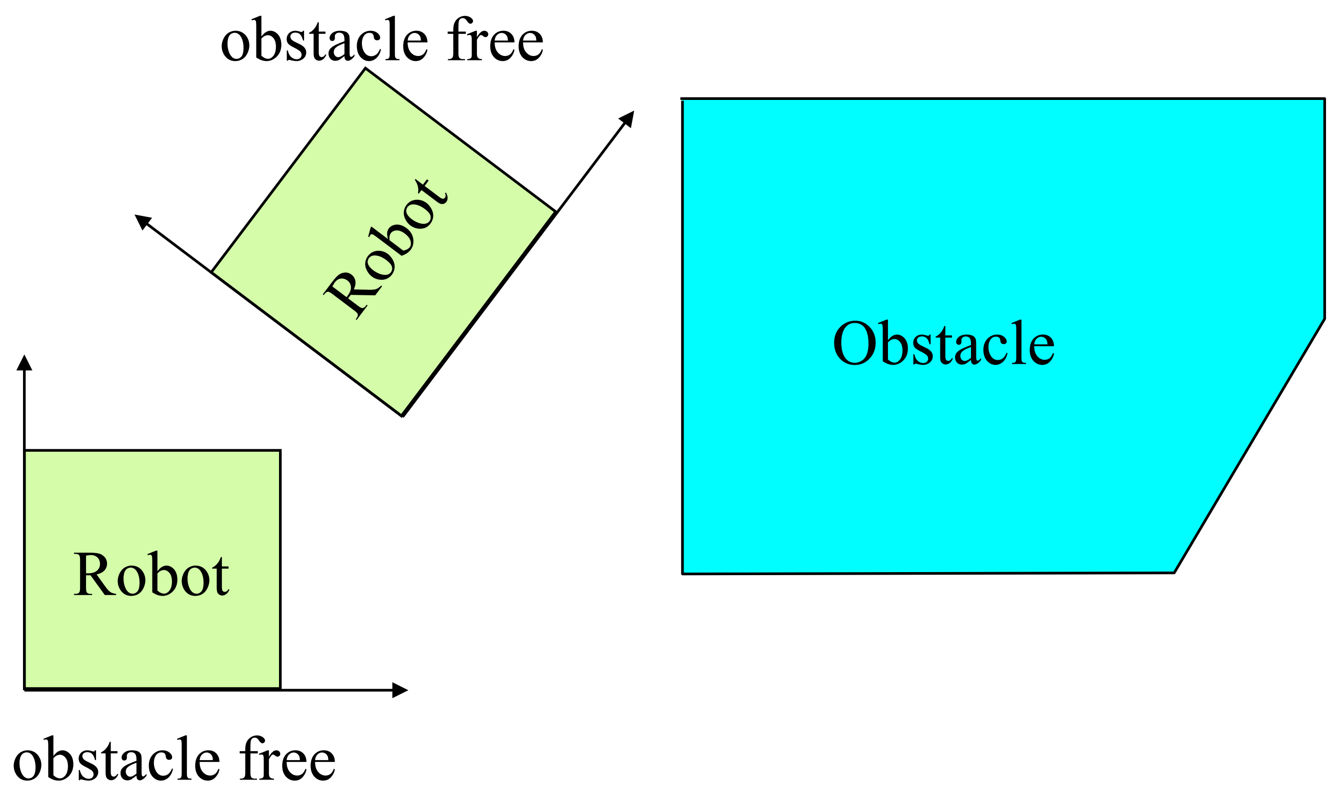


For what robot shapes does this work?

What if the robot is not a point?

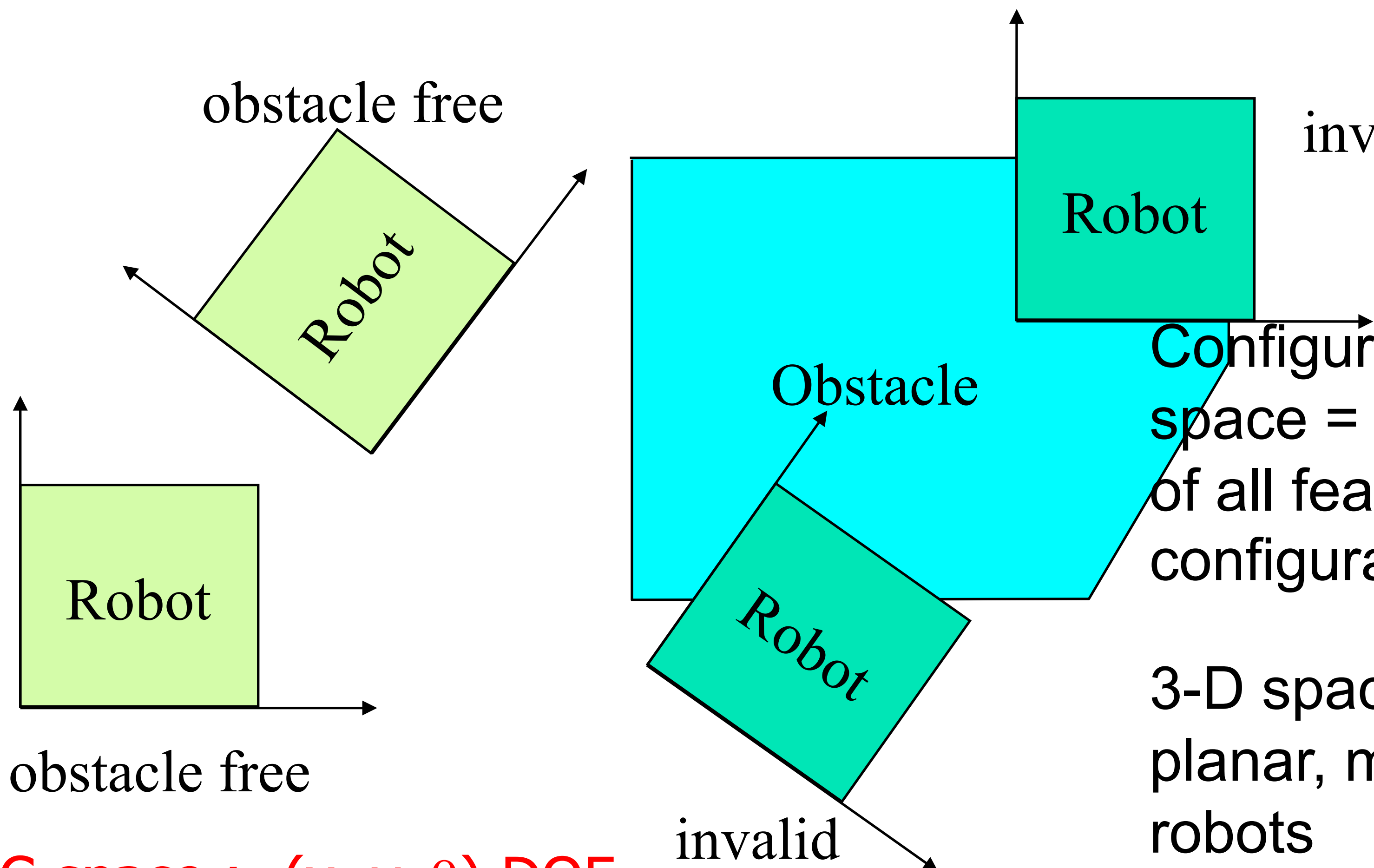


Configuration space



C-space : (x, y, θ) DOF

Configuration space

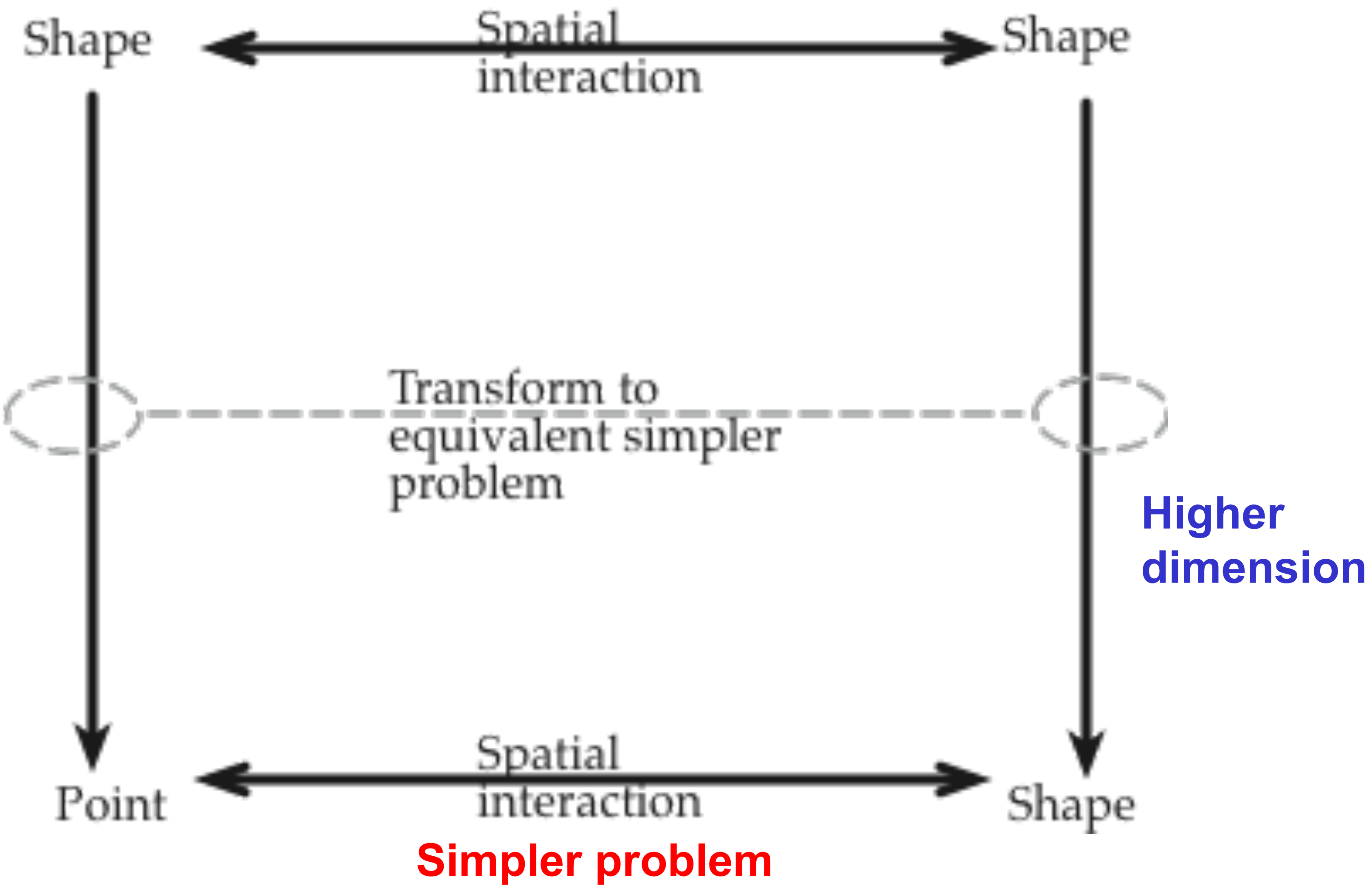


C-space : (x, y, θ) DOF

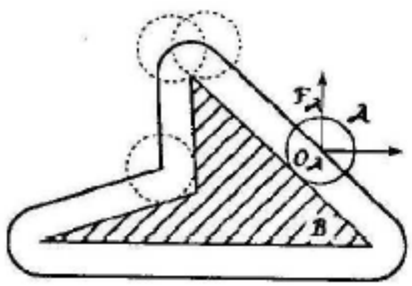
Configuration space = the set of all feasible configurations

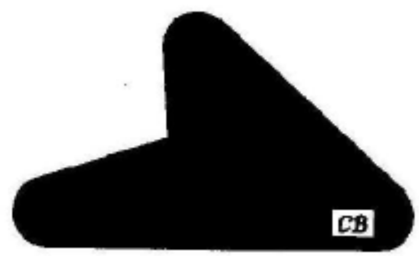
3-D space for planar, mobile robots

Transforming to C-Space

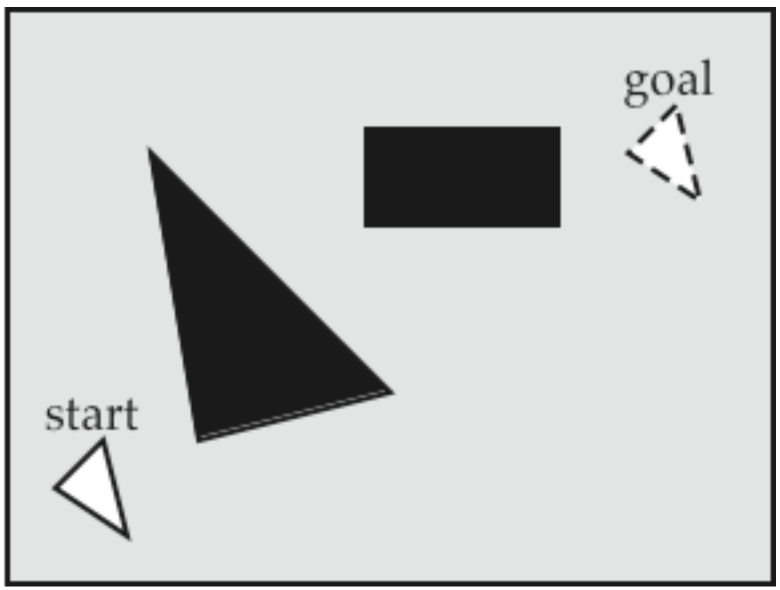


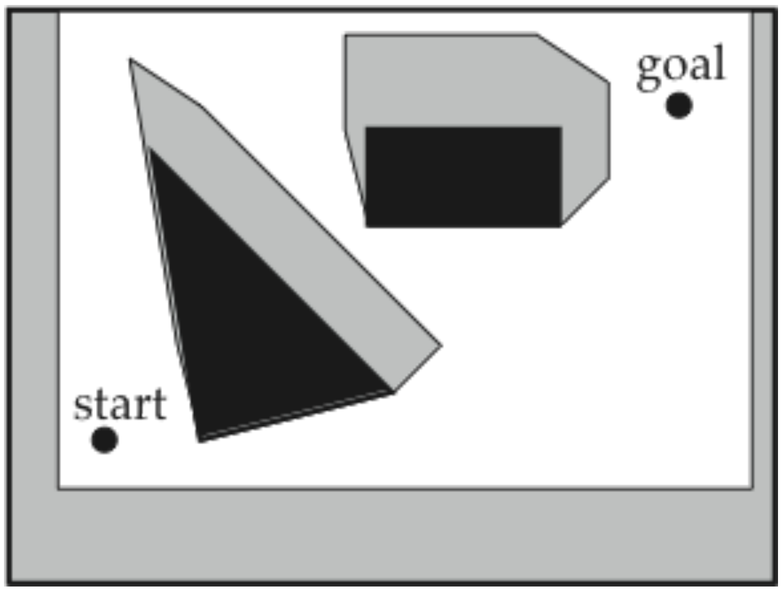
Robot Configuration Space



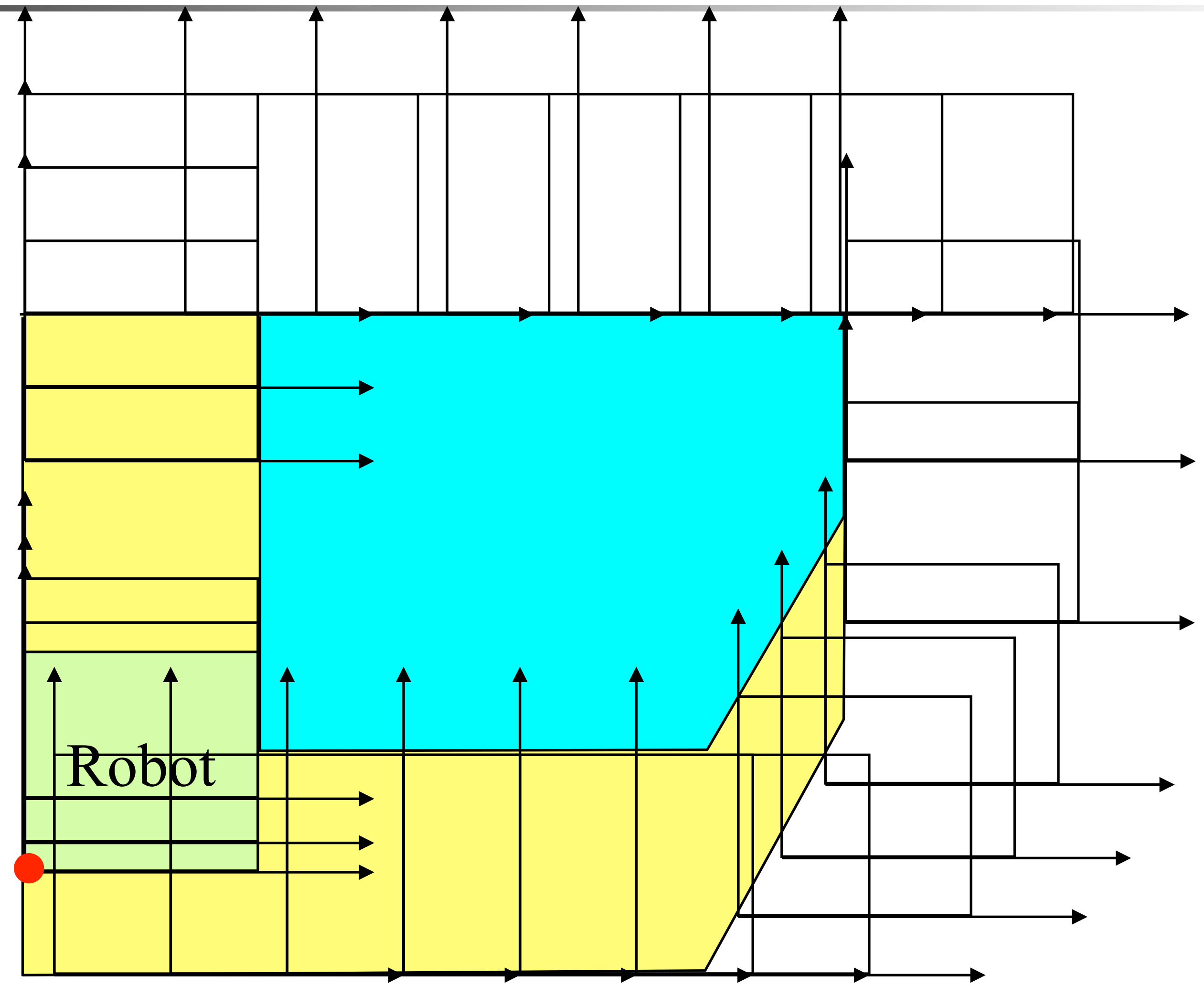


Transforming to C-Space

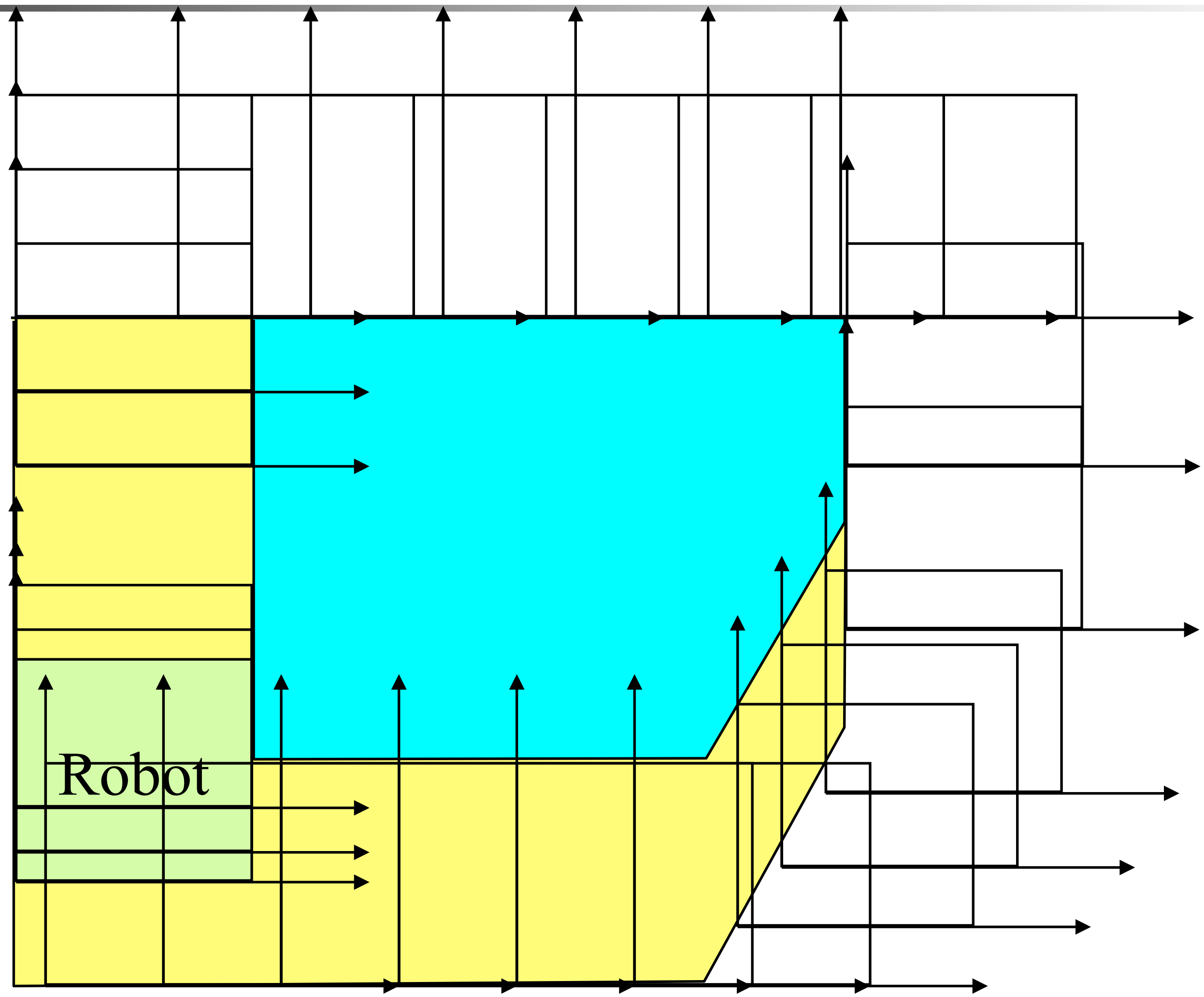




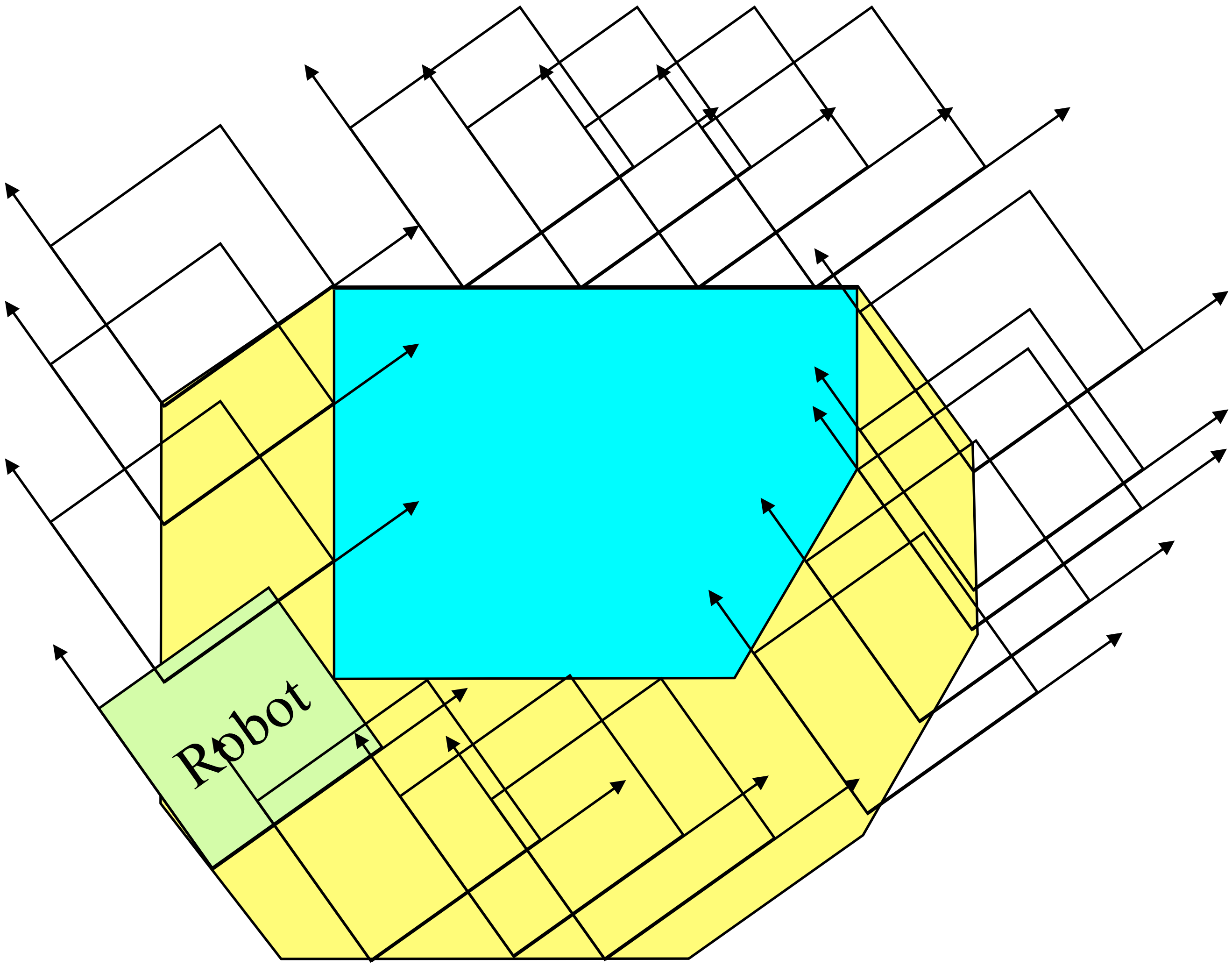
Allowable Robot positions (no rotations)



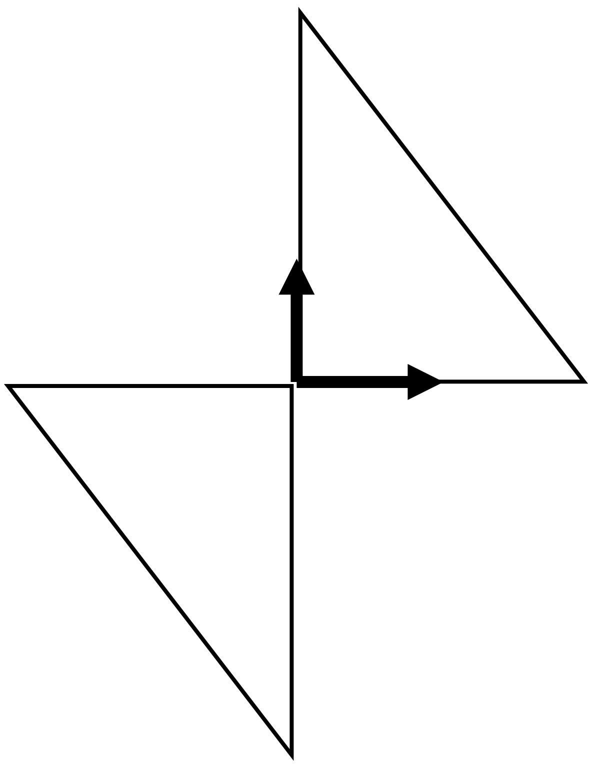
Allowable Robot positions (no rotations)



Allowable Robot positions (for some robot rotation)

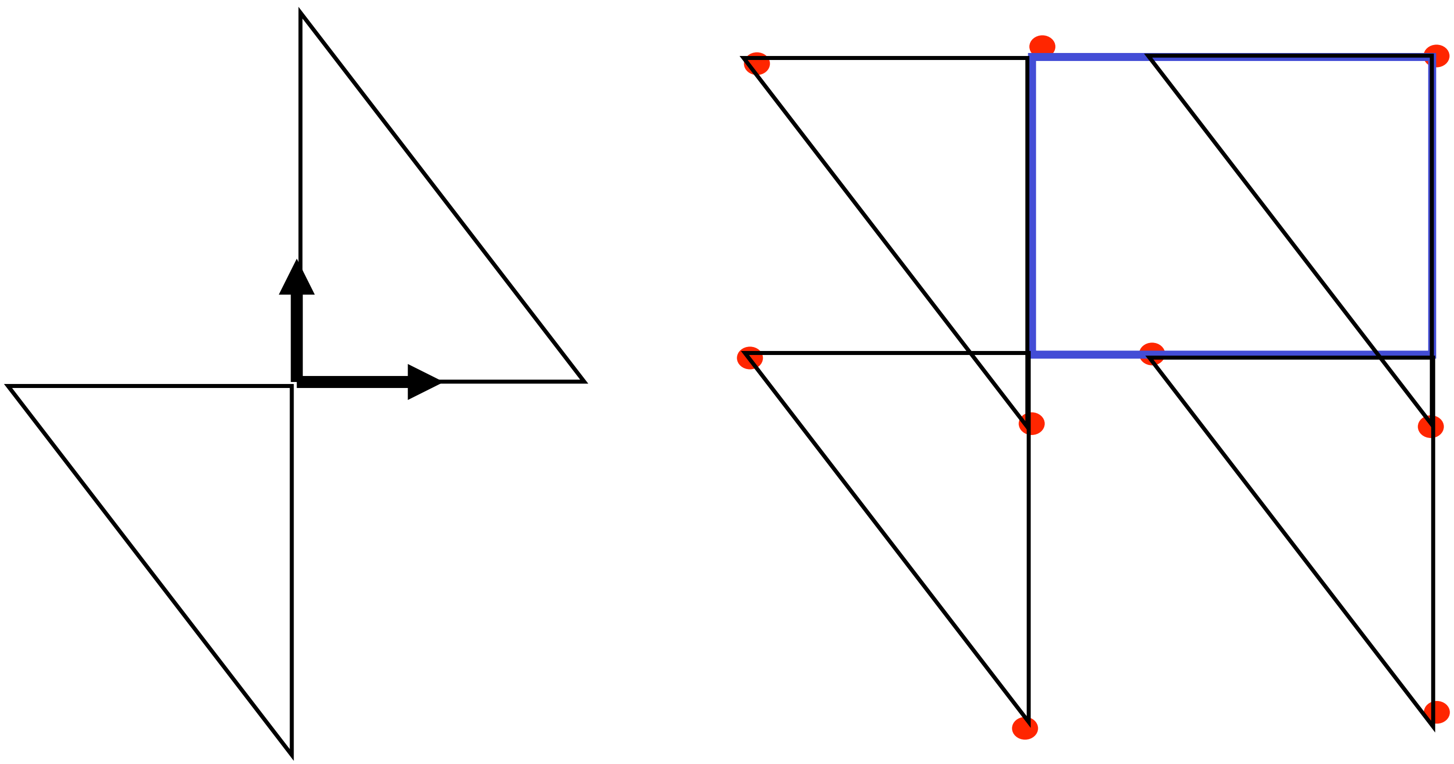


C-space Algorithm



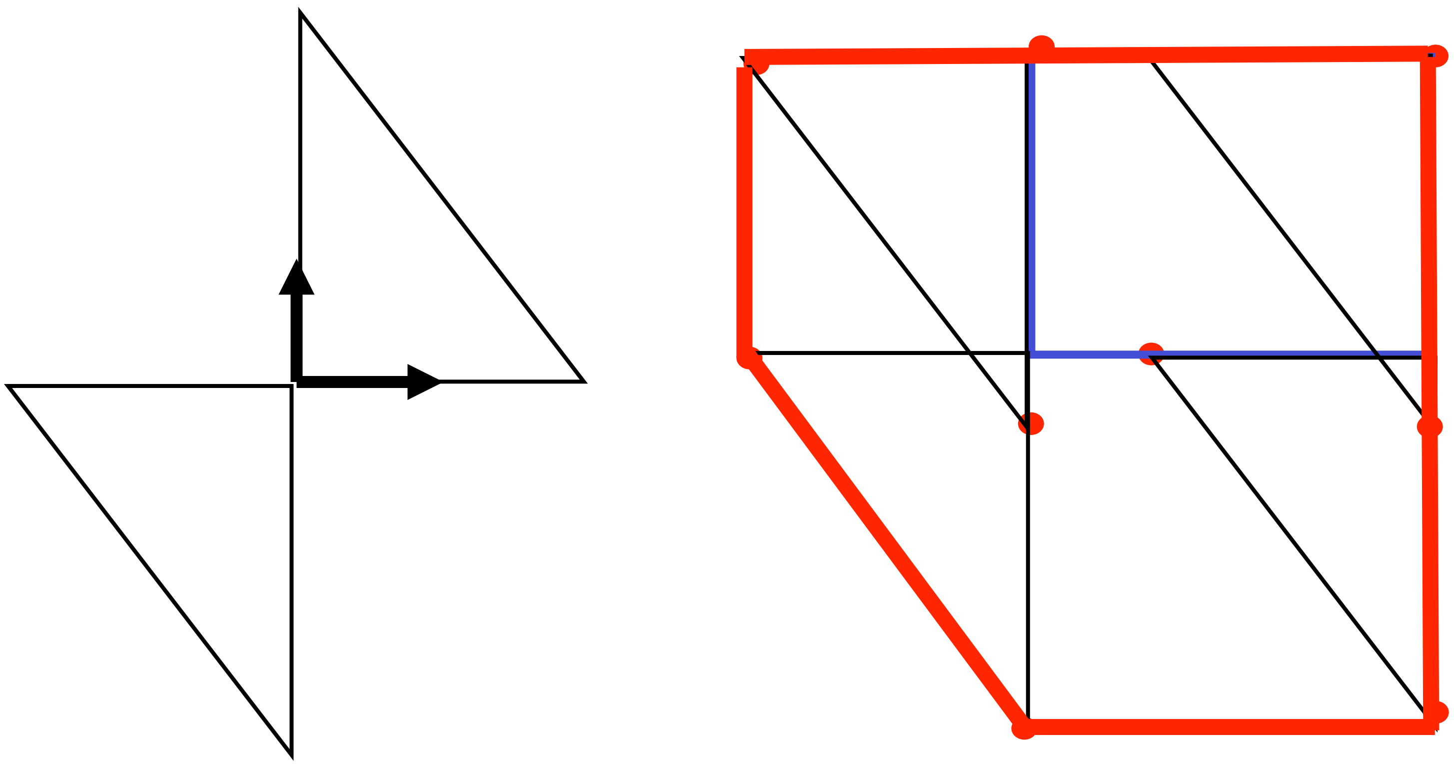
Step 1: Reflect Robot

C-space Algorithm



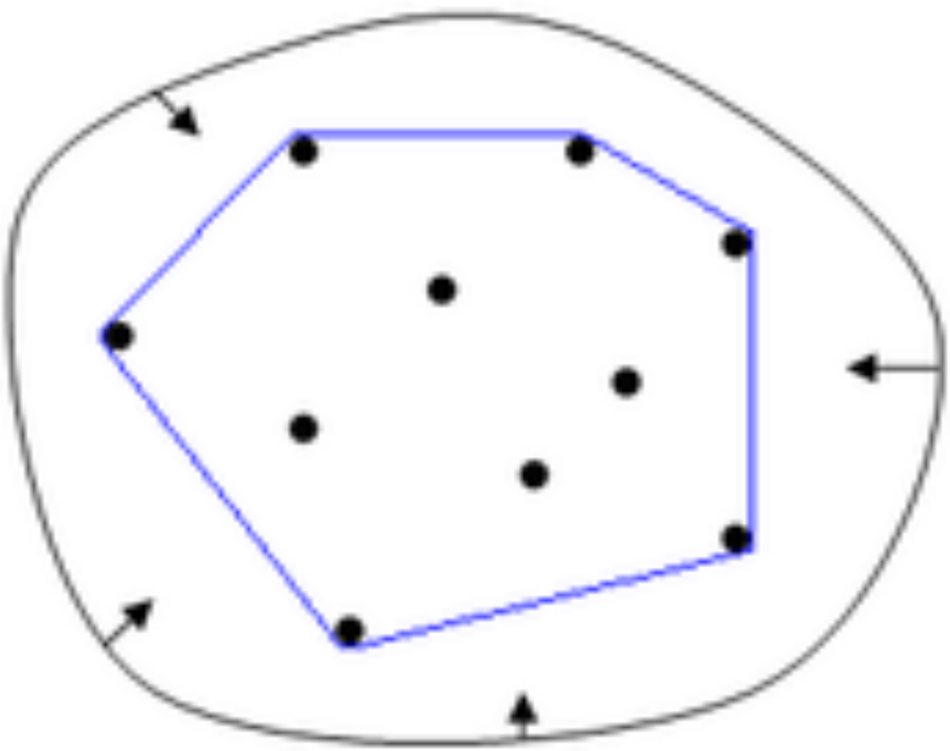
Step 2: Vert (⊖Robot) ⊕ Vert (Obstacle)

C-space Algorithm

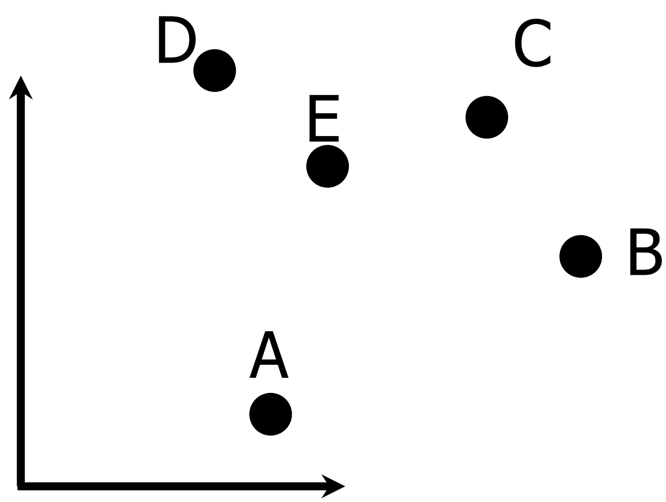


Step 3: ConvexHull (Vert (- Robot) + Vert (Obstacle))

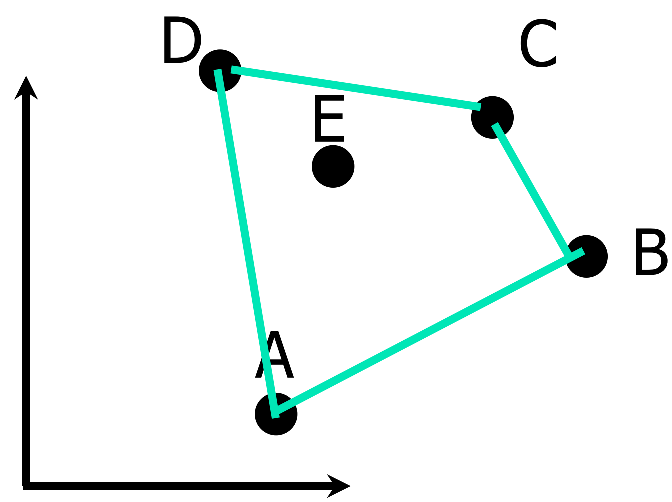
Convex Hull Algorithm



Convex Hull Algorithm

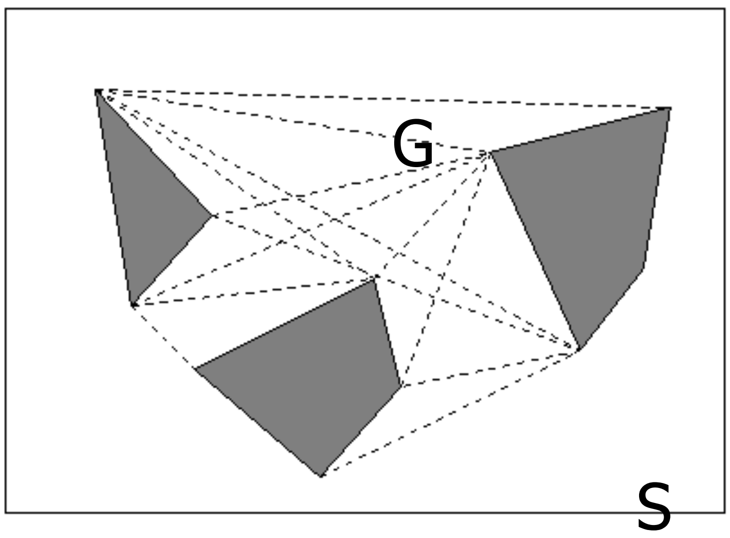


Convex Hull Algorithm



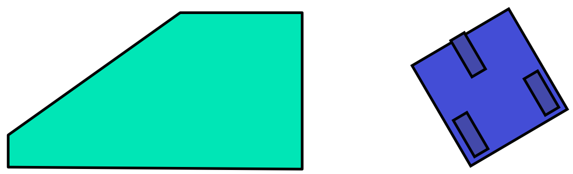
Algorithm Summary

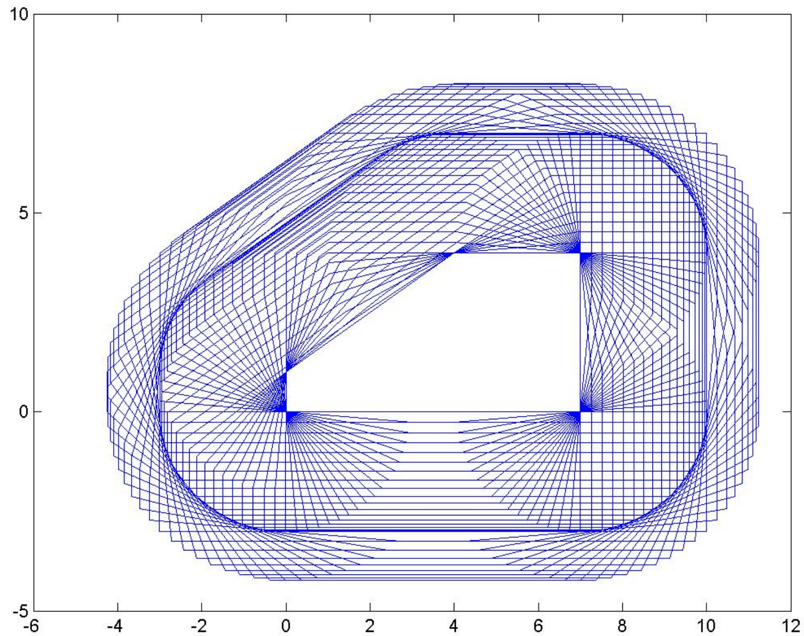
- Compute c-space for each obstacle
- Compute v-graph
- Find path from start to goal

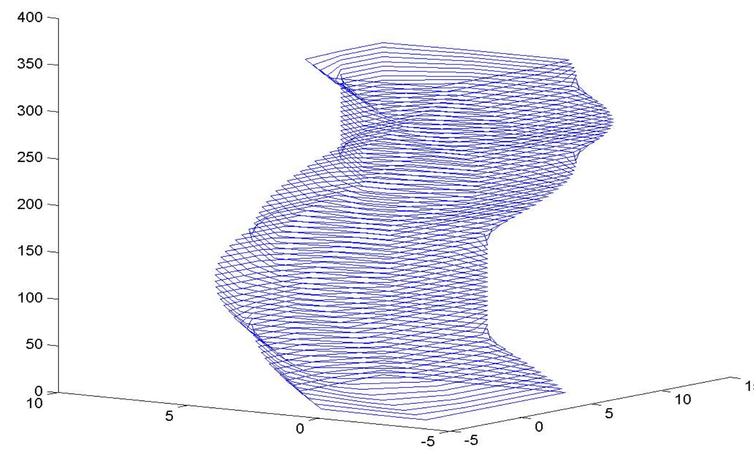


V-graph complete; gives optimal shortest path in 2d
What about 3d? What else can we optimize?

Configuration Space with Rotations







Piano Movers' Problem