6.141:Robotics systems and science Lecture 8: Motion Planning I

control architectures and c-space

Lecture Notes Prepared by Daniela Rus
EECS/MIT

Spring 2012

Thanks to Vijay Kumar

Reading: Chapter 3, and Craig: Robotics

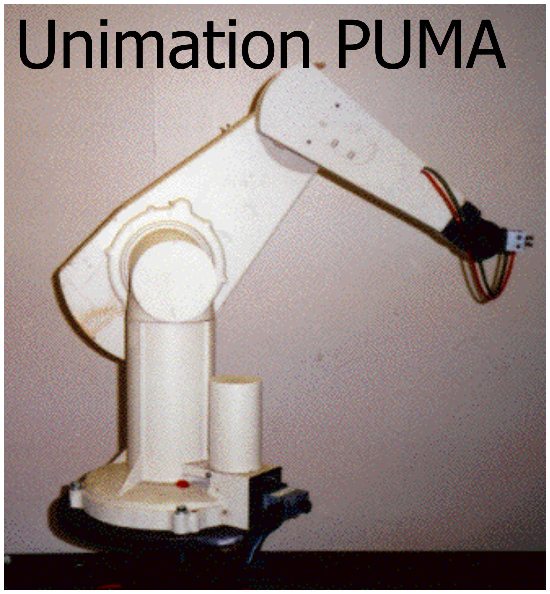
http://courses.csail.mit.edu/6.141/

Challenge: Build a Shelter on Mars

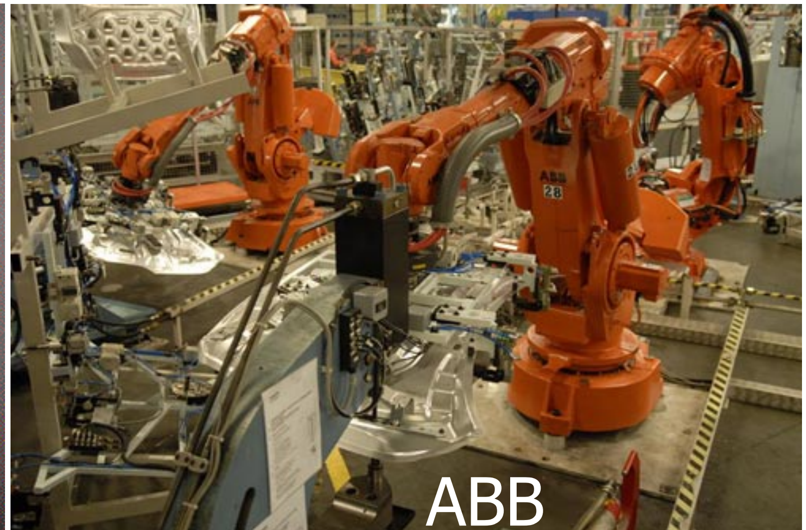
Where are we?

- Last Module:
  - Vision
  - Software engineering
- Today:
  - Robot Control Architectures
  - Deliberative control: motion planning and c-spaces
- Reading: chapter 6

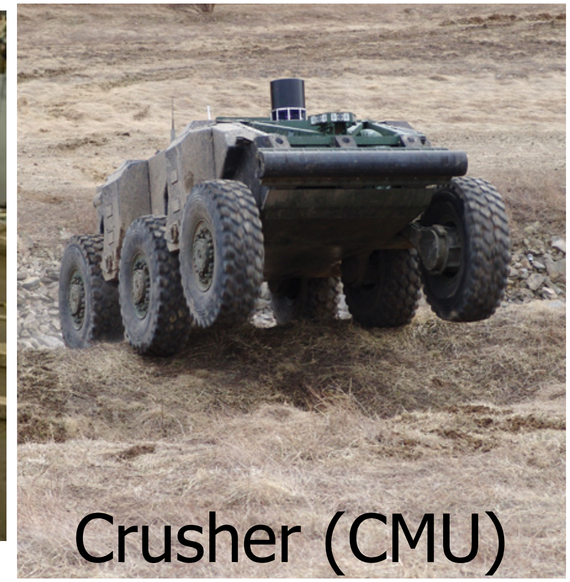
Motion Planning



Unimation PUMA



ABB



Crusher (CMU)

Kawada HRP-2

Willow Garage PR2

KUKA youBot

HERB (CMU)

What is motion planning?

Objective:

- Find a series of control actions that moves the robot from a start state to a goal condition, while respecting constraints and avoiding collision.

Task planning:

- Start and goal are expressed in terms of environment state rather than robot state
- May require symbolic sequential reasoning

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

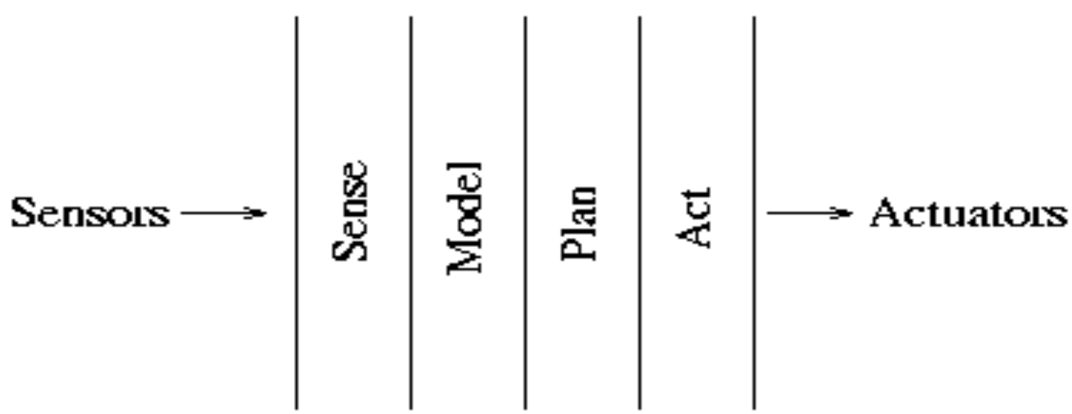
- A control architecture provides a set of principles for organizing a robot (software) control system.
- Like in computer architecture, it specifies building blocks
- It provides:
  - structure
  - constraints

Control Architecture Types

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

Deliberative Architecture

- Maps, lots of state
- Look-ahead
- "Think hard, then act"

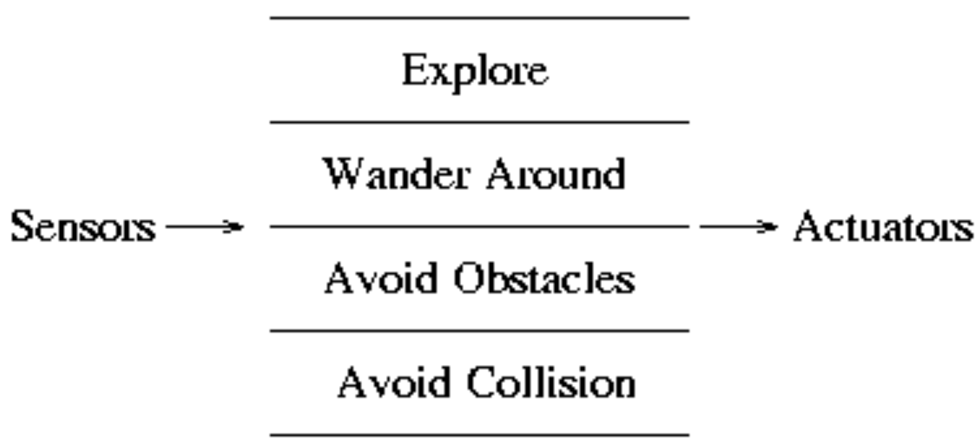


Reactive Architecture

- No maps, no state
- No look ahead
- "Don't think, react!"

Behavior-based Architecture

- Multiple concurrent sense-act processes
- Each behavior uses local sensing to compute its best action
- Robot a combination of behaviors
- "Think the way you act"



Criteria For Selection

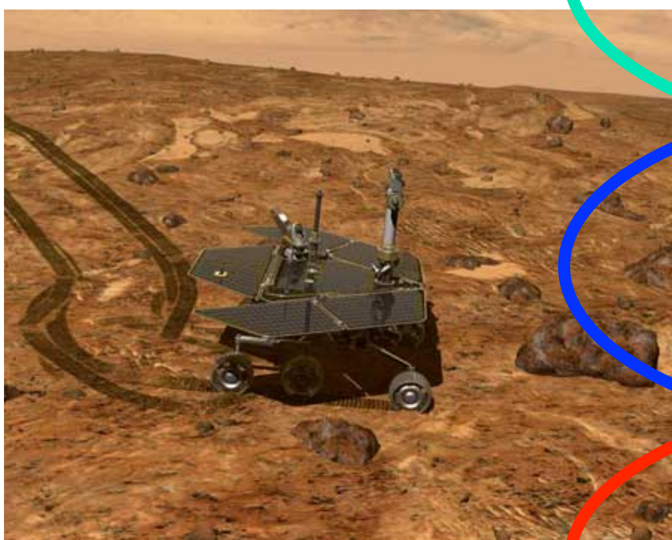
	deliberative	reactive	behavior
Task and environment			
Run-time constraints			
Correctness/ Completeness			
Hardware			

Motion Planning

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

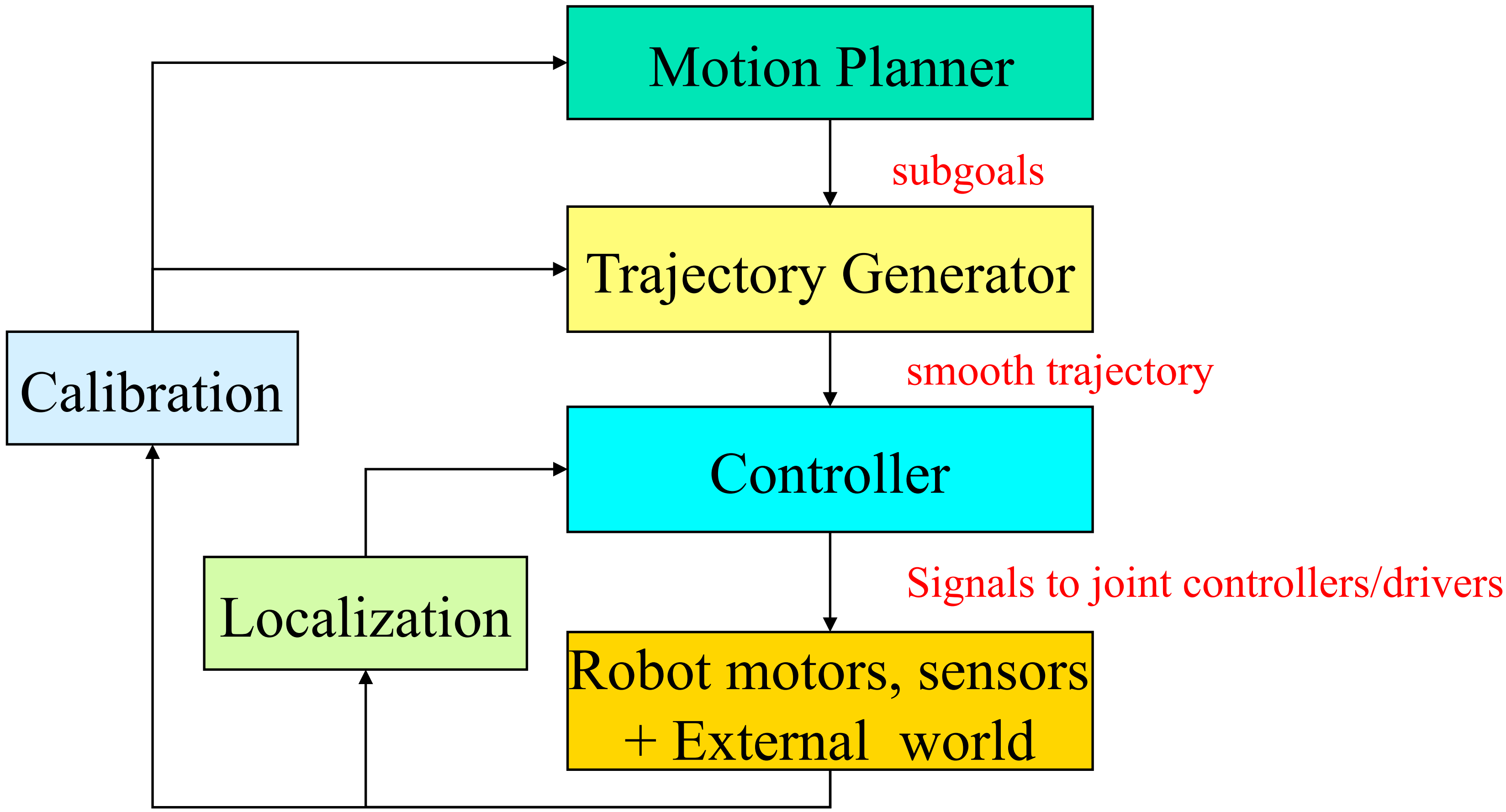
Complications: error in maps, sensing, control, unexpected obstacles, etc.

Deliberative Architecture

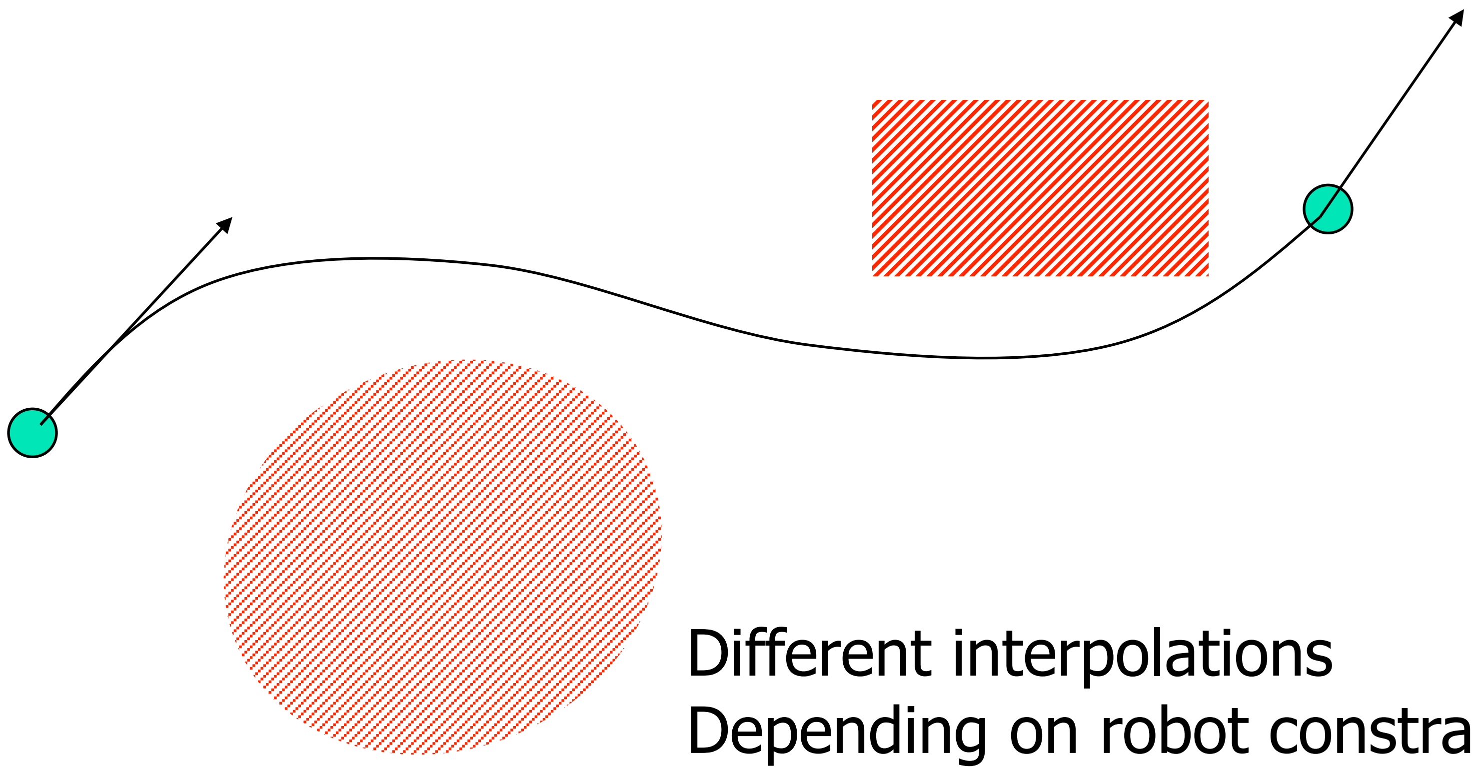


- Sense — Last Week
  - Local data about the world
- Map — In 2 Weeks
  - Global picture pf the world
- Plan — Next week
  - Signals to joint controllers/drivers
    - joint velocities, joint torques
- Act

Motion Planning



Trajectory generation from waypoints



Different interpolations
Depending on robot constraints

Motion Planning

- Known Environments (Model)

- Unknown Environments (No Model)

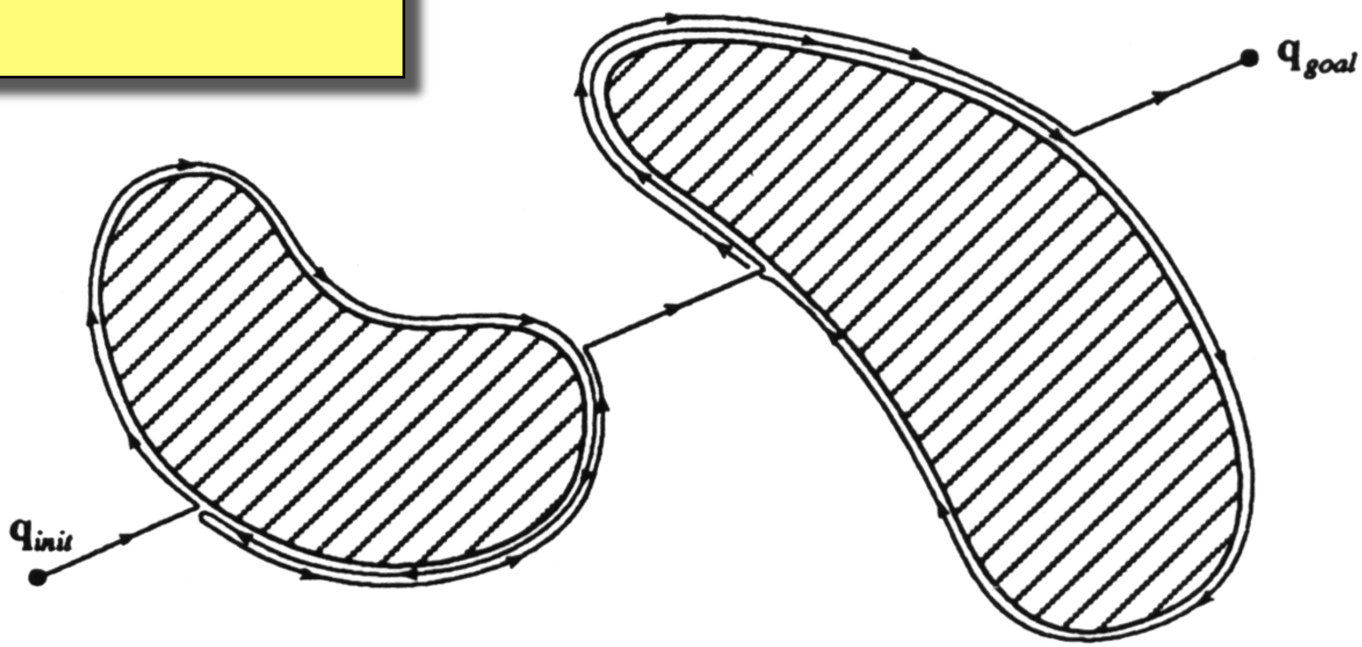
OFFLINE ALGORITHMS

ONLINE ALGORITHMS

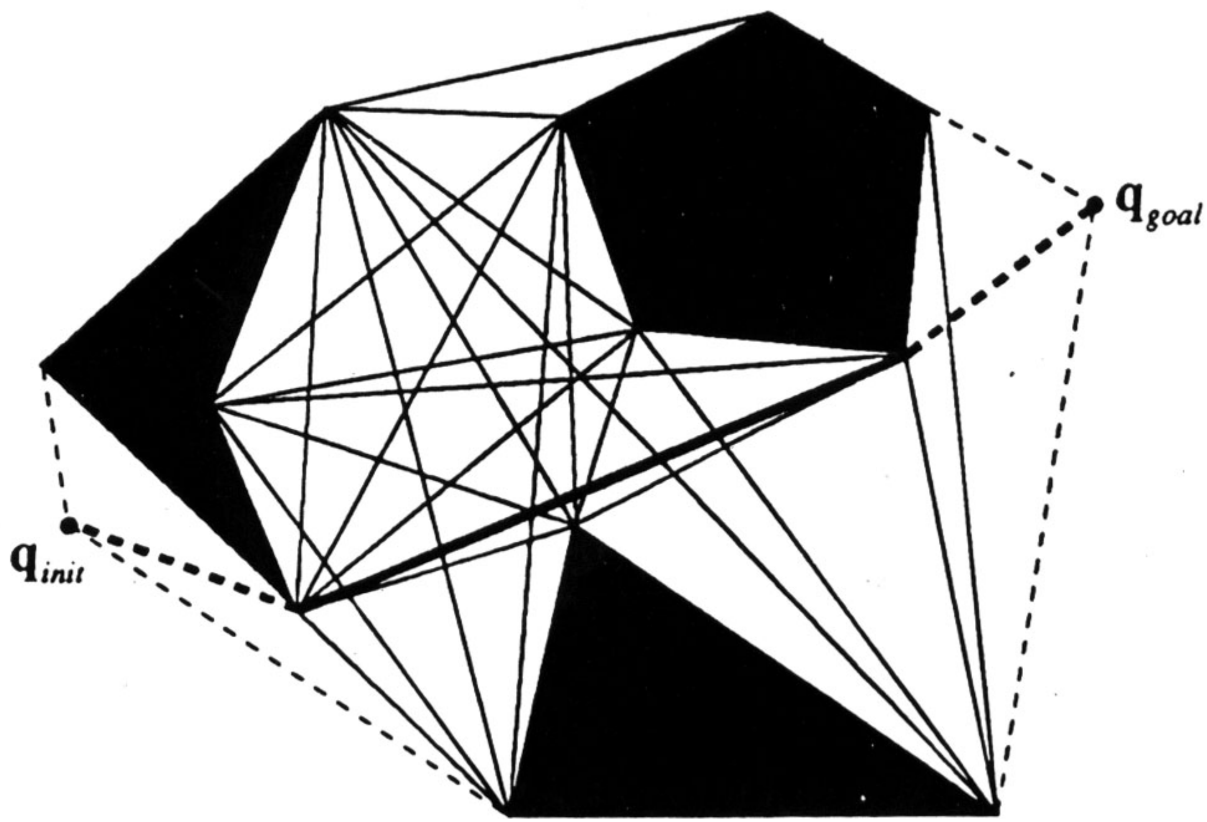
Example: how do we find a bridge in the fog?

Online Motion Planning

Always finds a path
(if it exists)



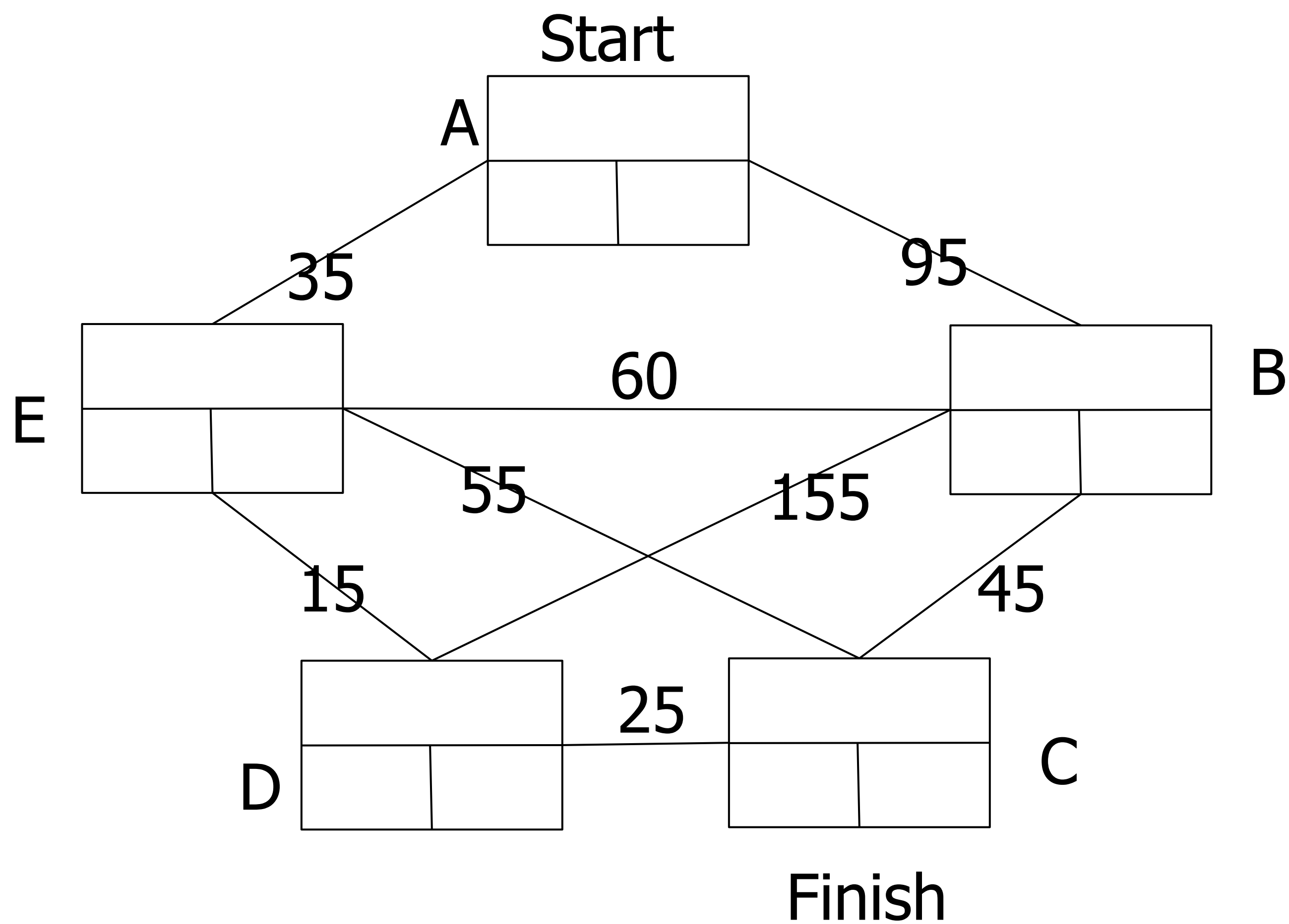
Visibility Graphs



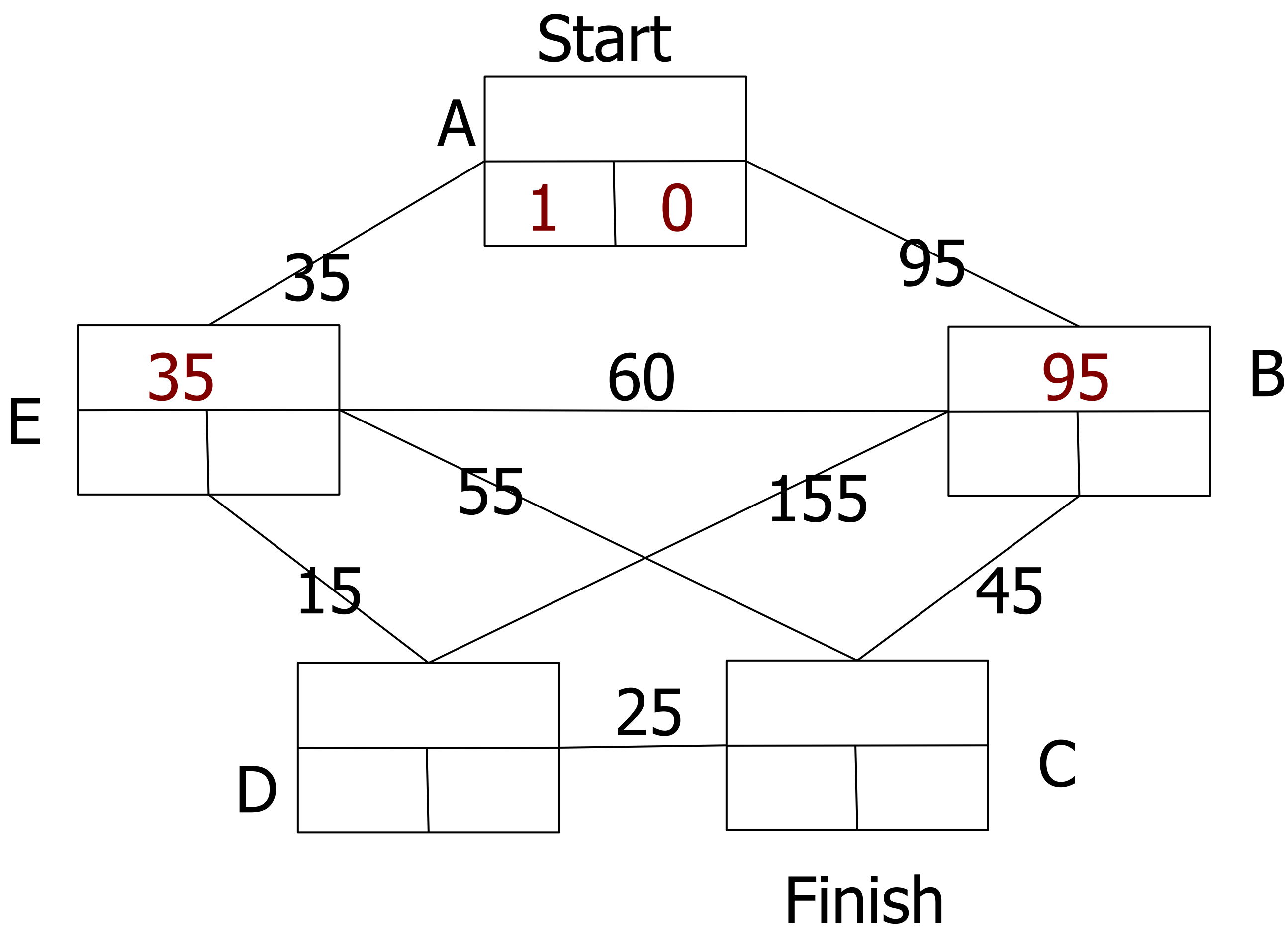
Vertices: Start, Goal,
obstacle vertices

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

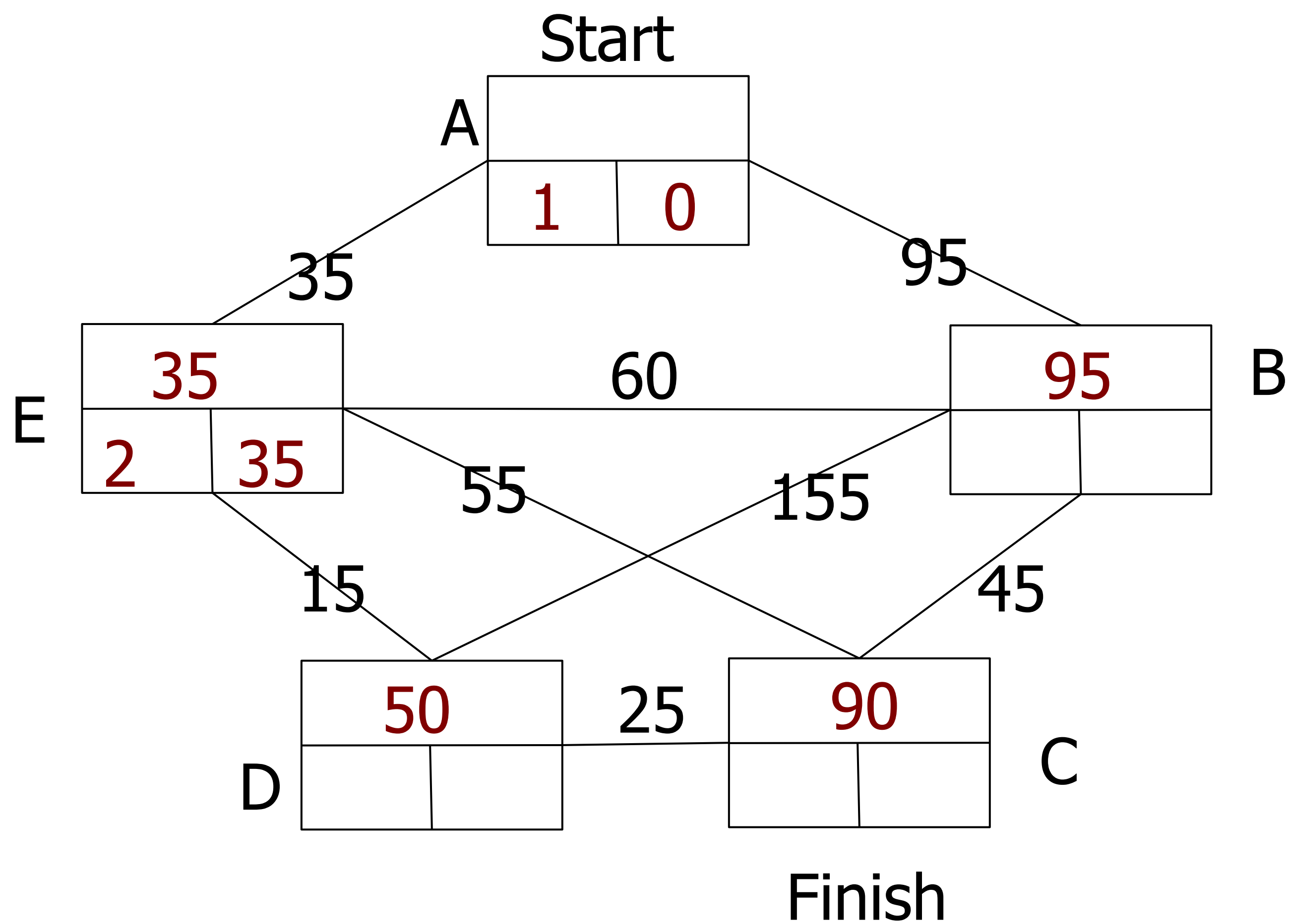
Shortest path



Shortest path

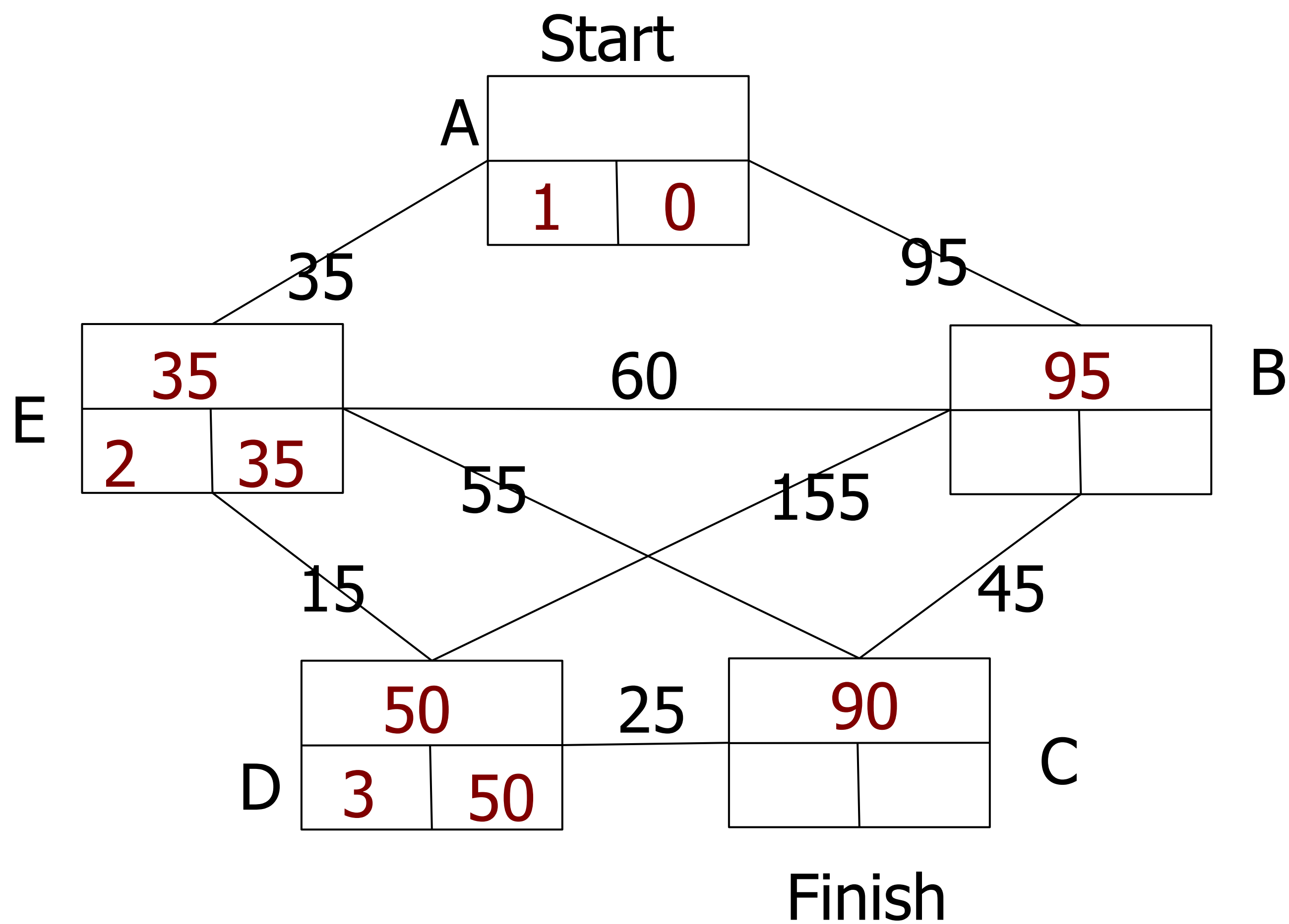


Shortest path



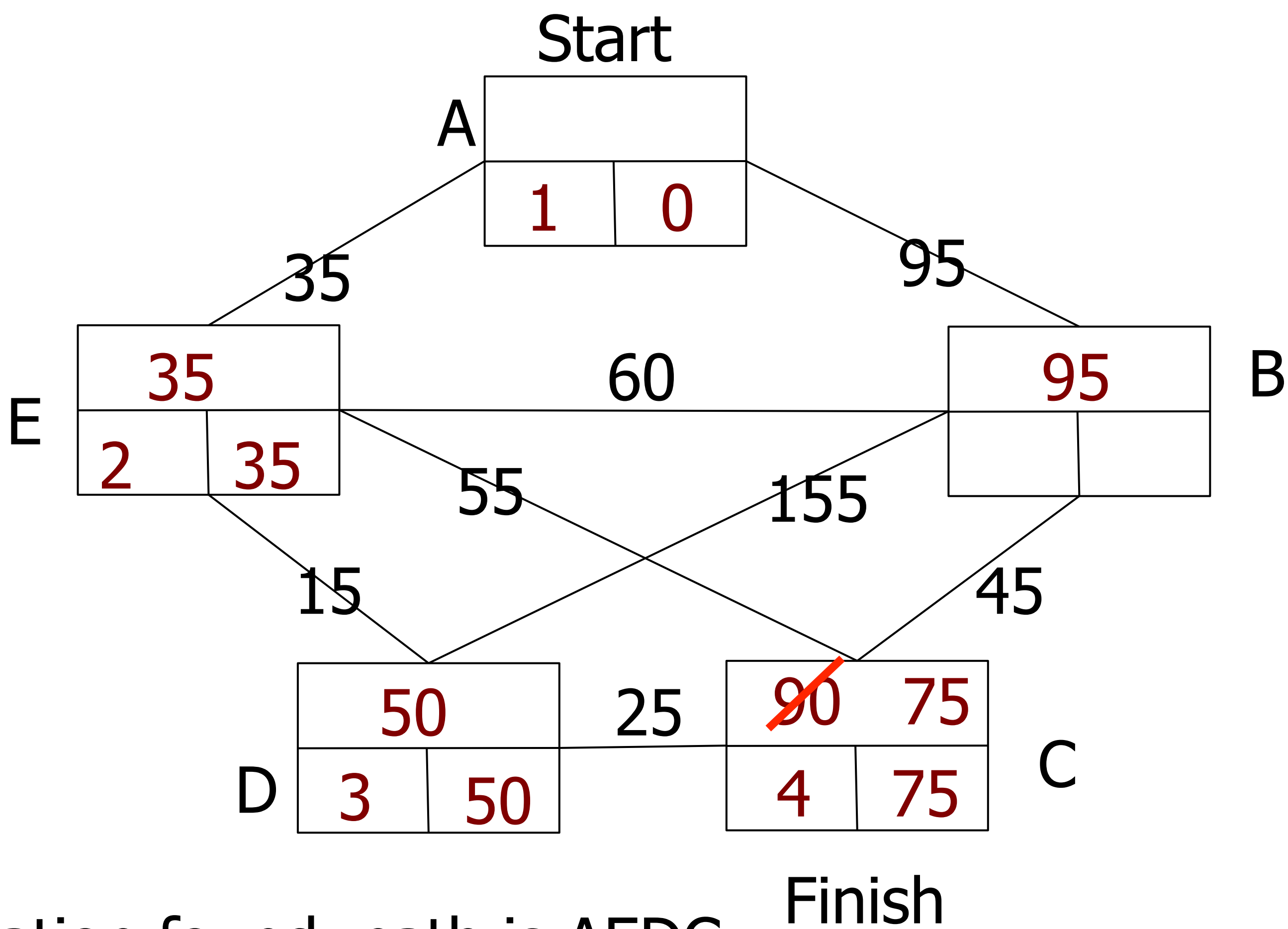
Find node with smallest temporary value; label neighbors

Shortest path



Find node with smallest temporary value; label neighbors

Shortest path

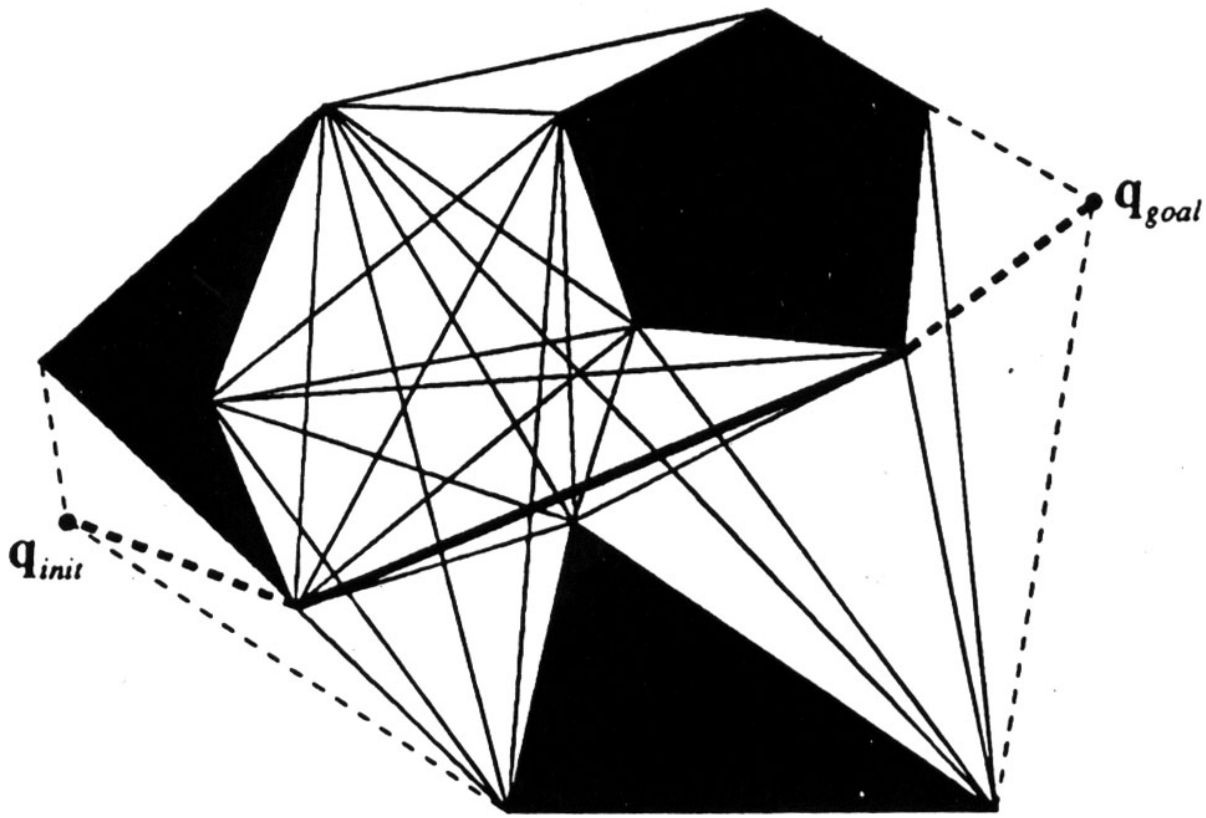


Destination found, path is AEDC

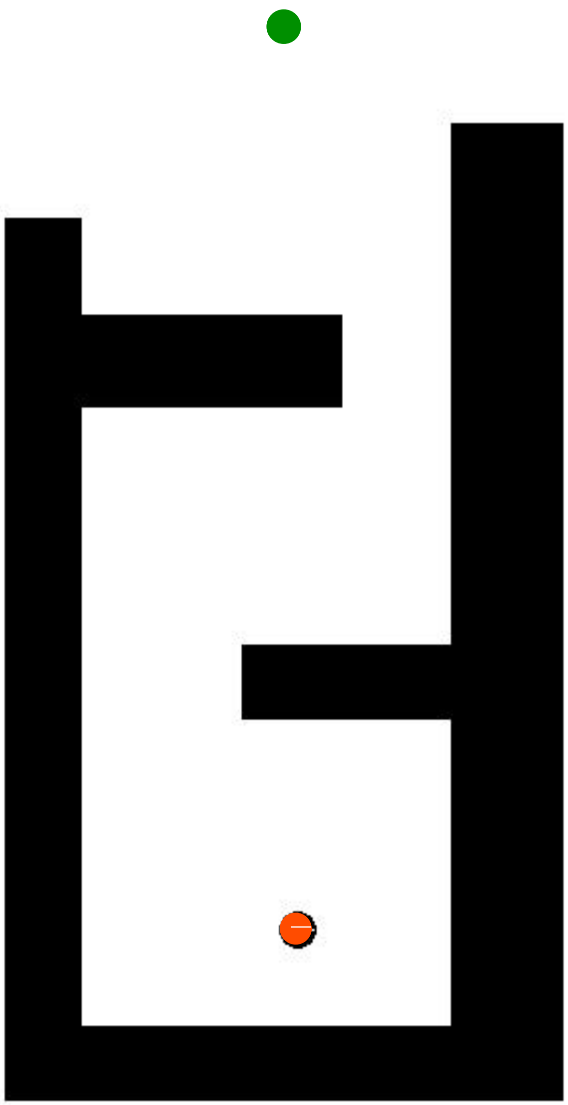
Search Path: Dijkstra's Algorithm

```
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
```

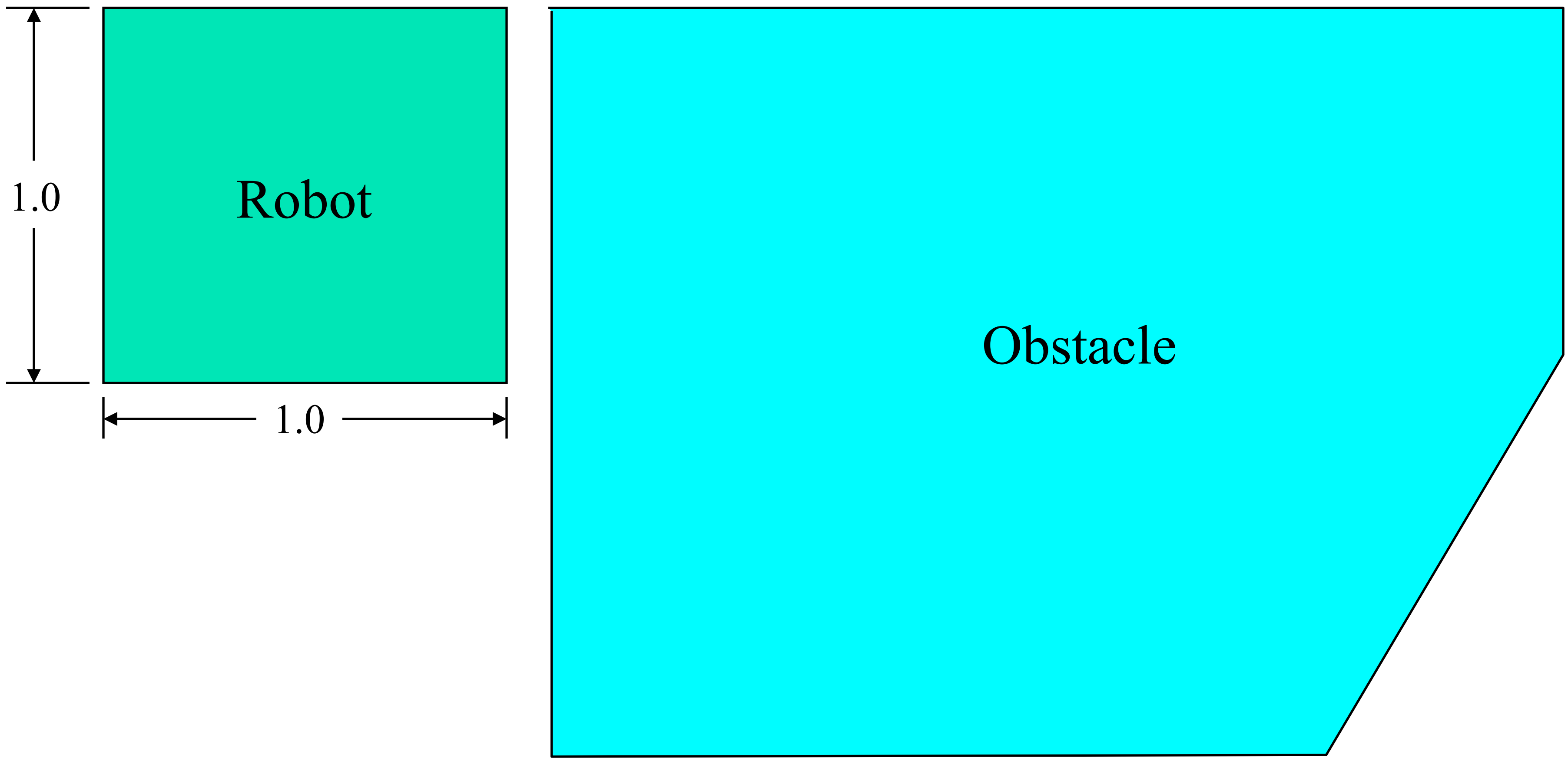
Visibility Graphs Summary



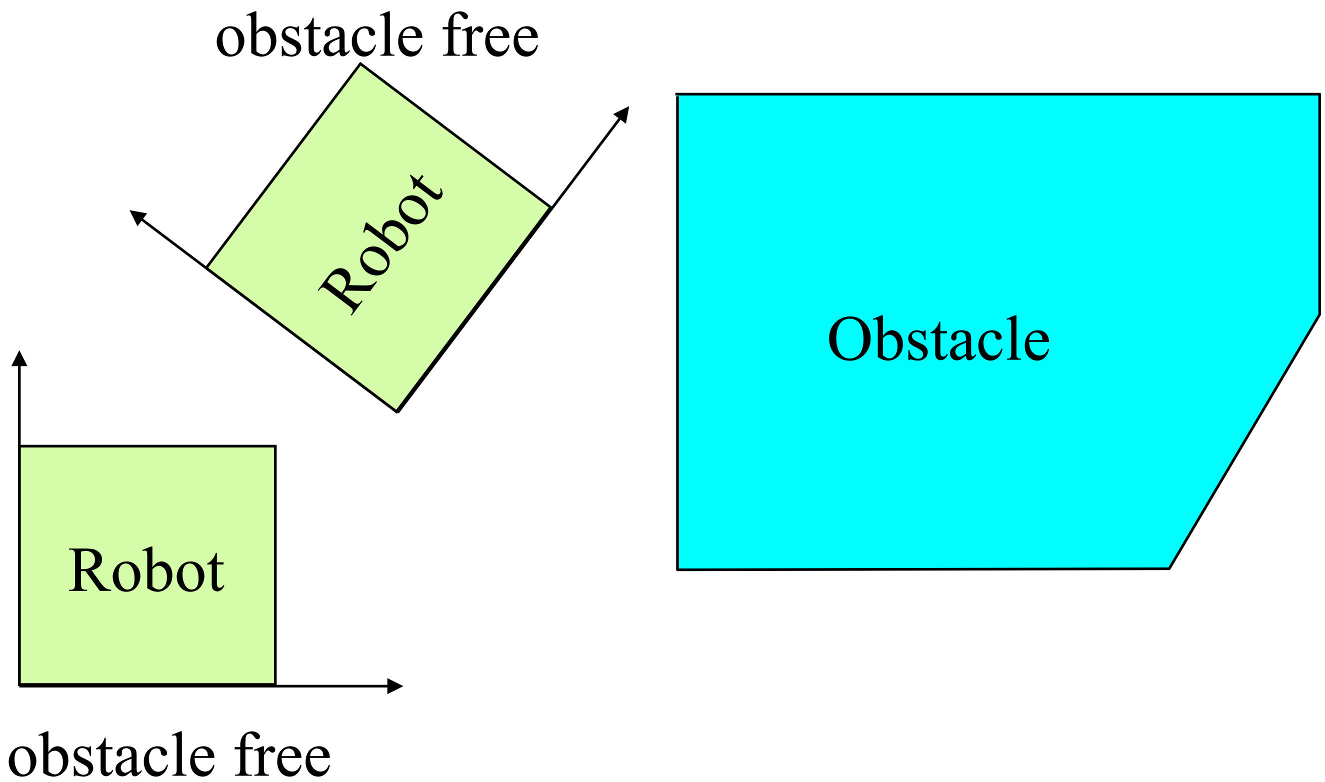
What if the robot is not a point?



What if the robot is not a point?

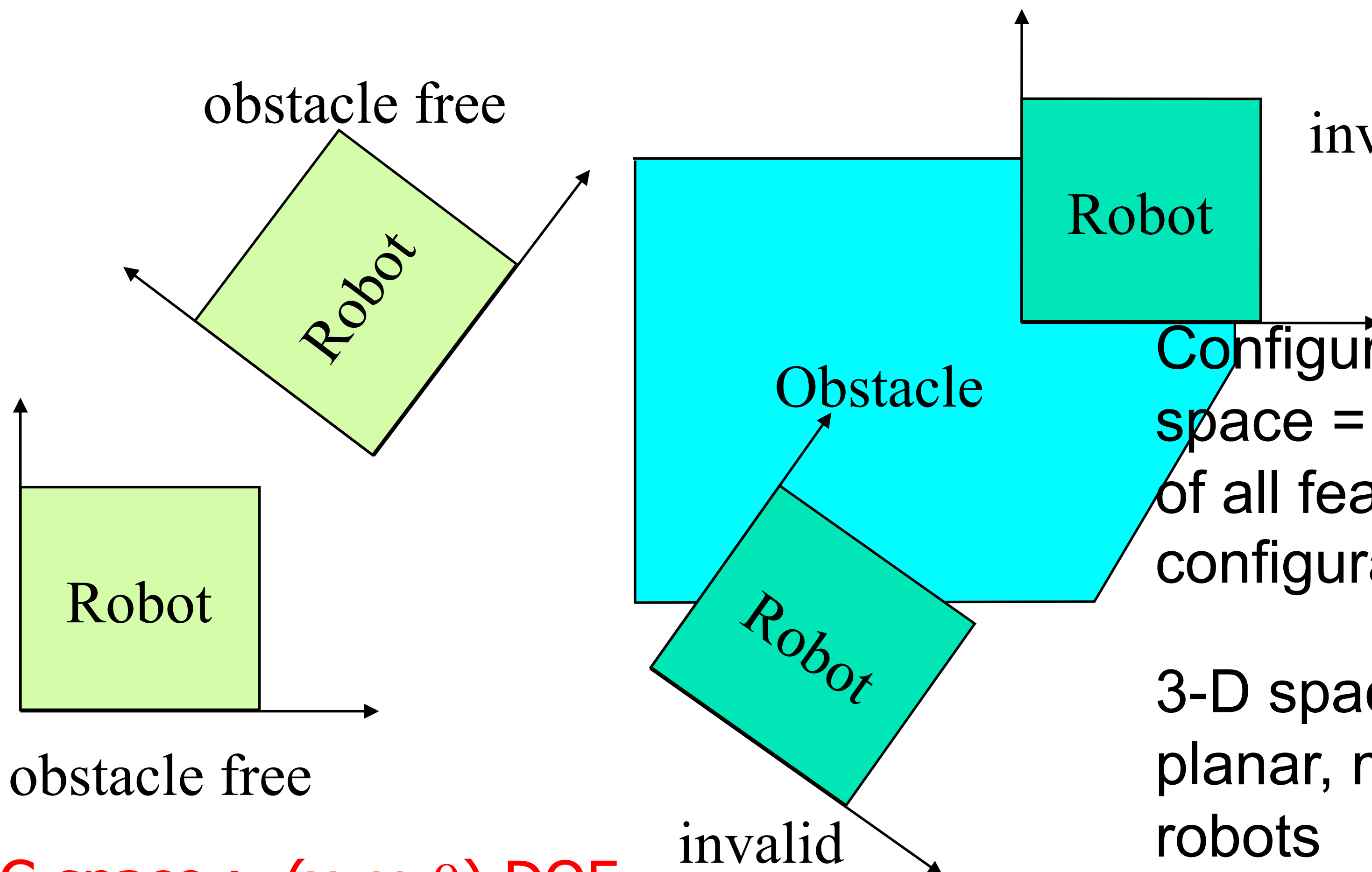


Configuration space



C-space : (x, y, θ) DOF

Configuration space



Configuration space = the set of all feasible configurations

3-D space for planar, mobile robots

C-space : (x, y, θ) DOF

Free Space

From
Robot Motion Planning
J.C. Latombe

C-OBSTACLE REGION

$B_1, B_2, \ldots, B_m$ —— obstacles

CB_i —— C-obstacle

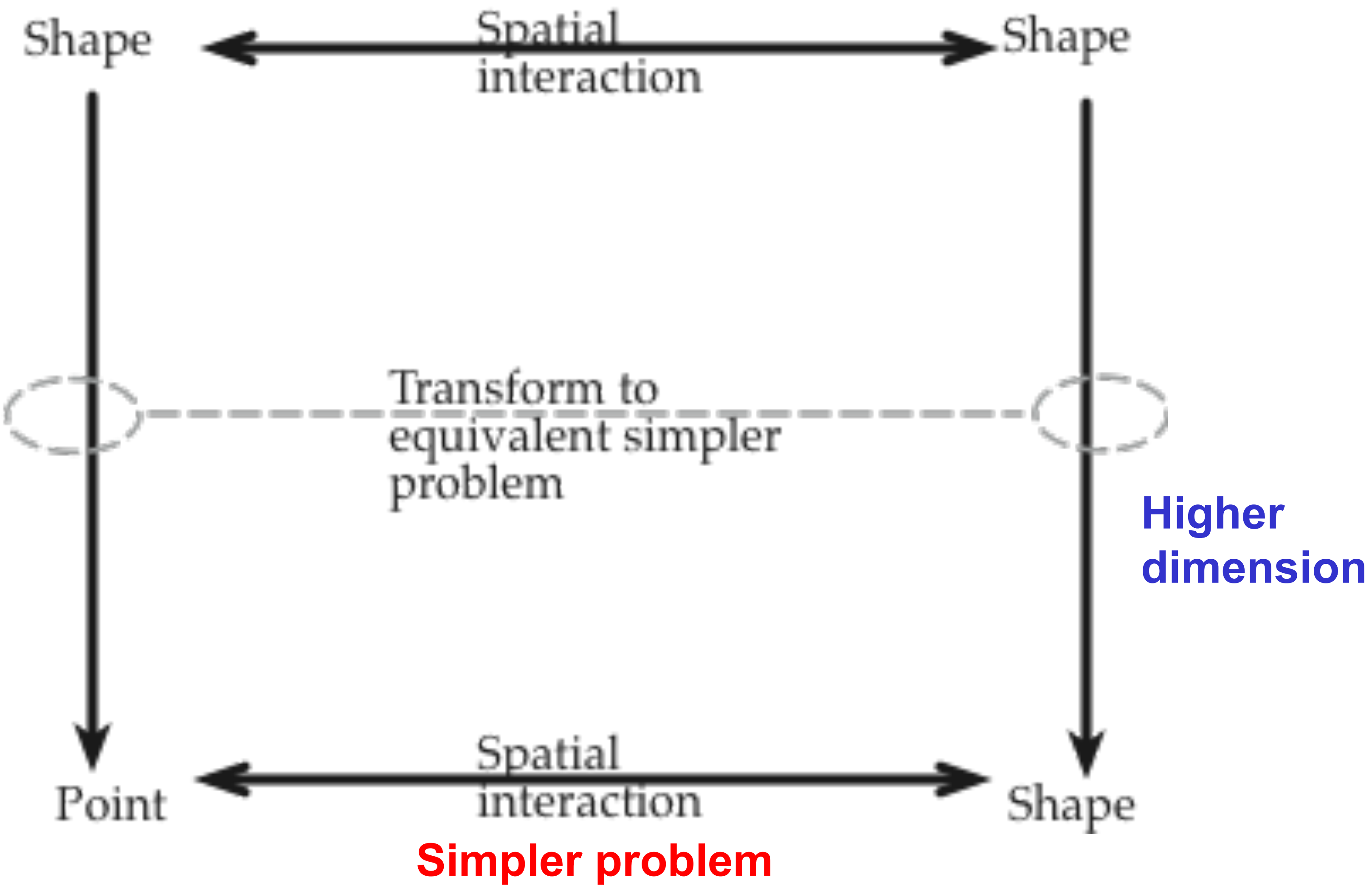
$\bigcup_{i=1}^{m} CB_i$ —— C-obstacle region

FREE SPACE

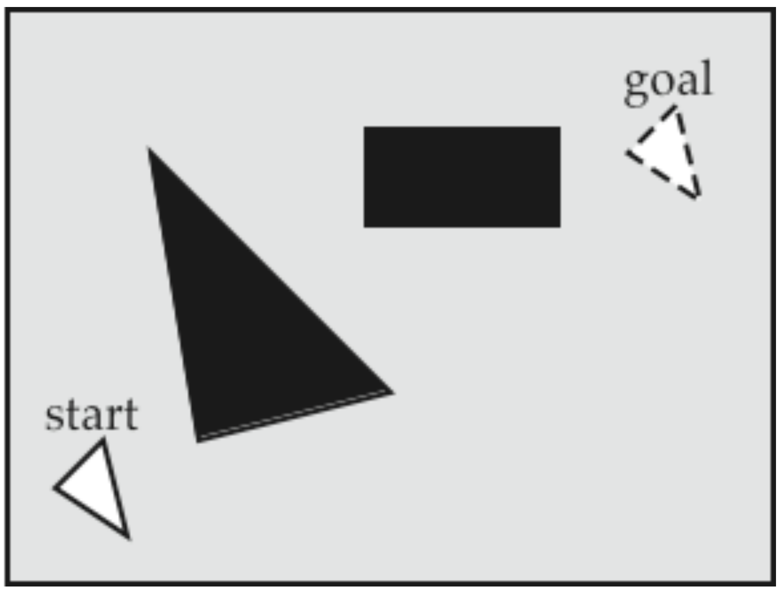
$$C_{free} = C \setminus \bigcup_{i=1}^{m} CB_i = \{q \in C : A(q) \cap \bigcup_{i=1}^{m} CB_i = \phi\}$$

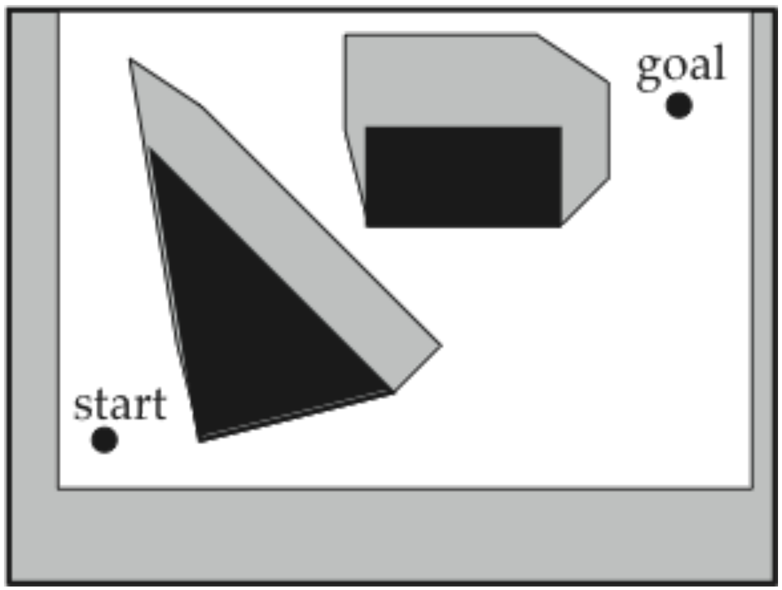
Free configuration q iff $q \in C_{free}$

Transforming to C-Space

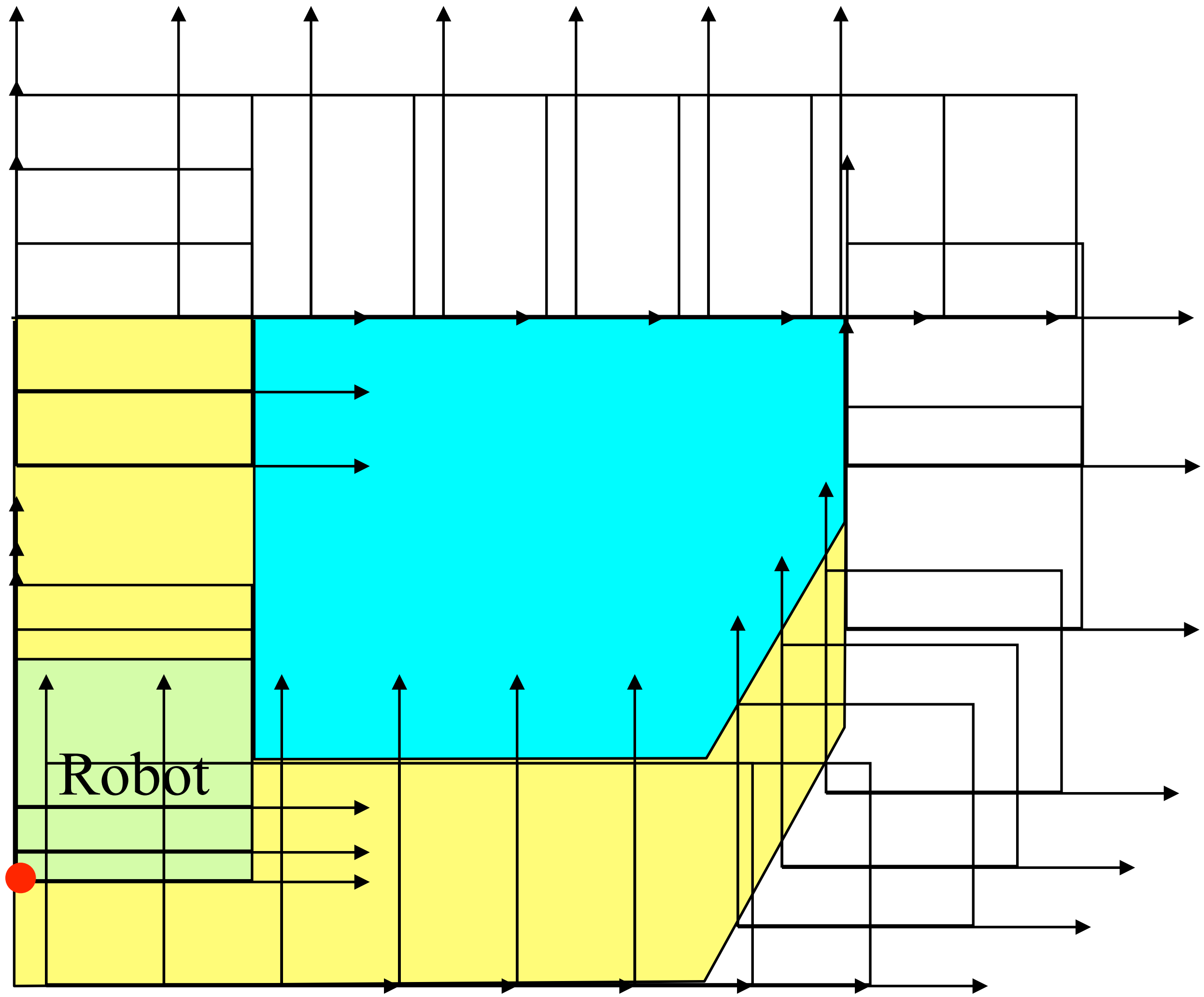


Transforming to C-Space

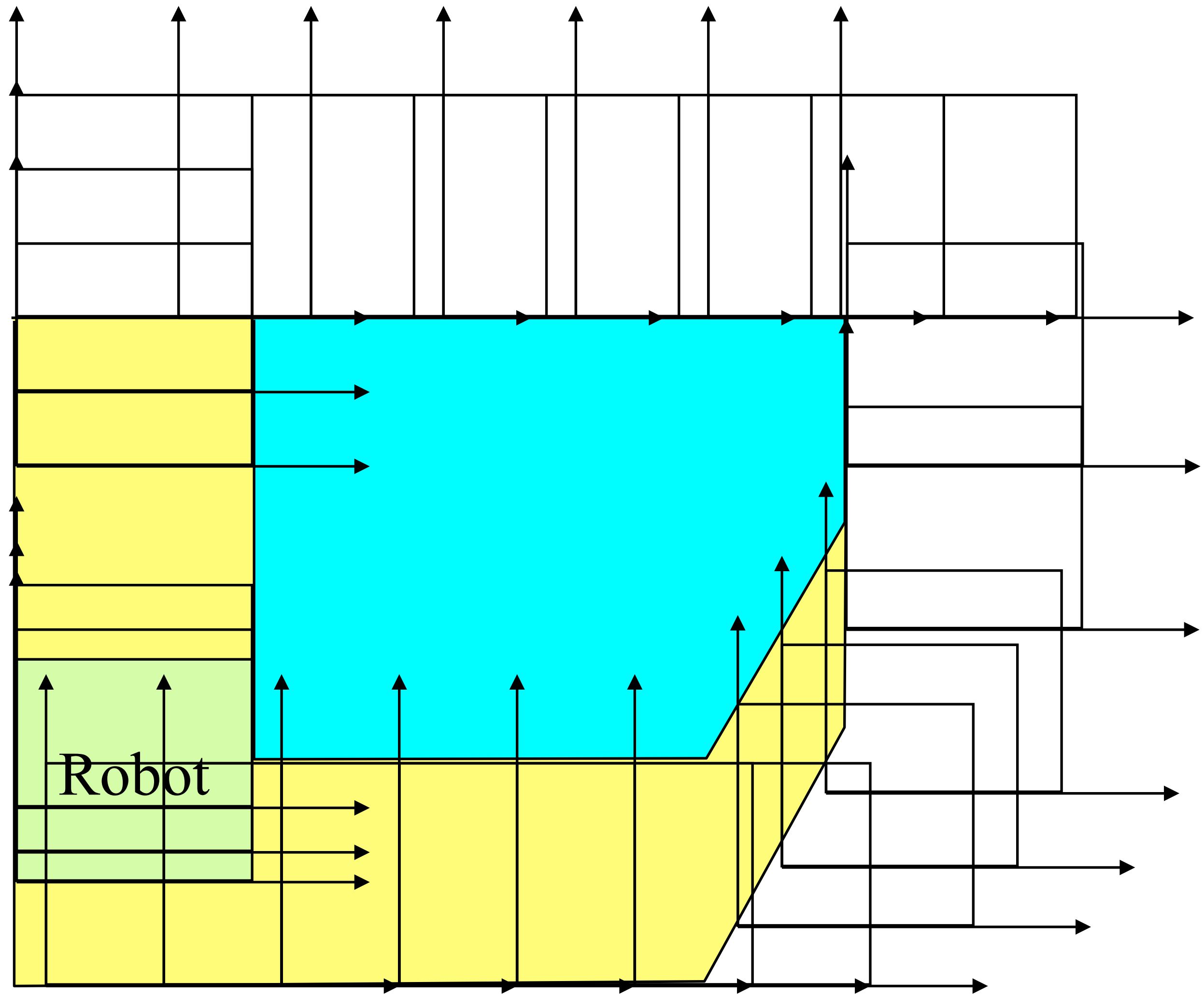




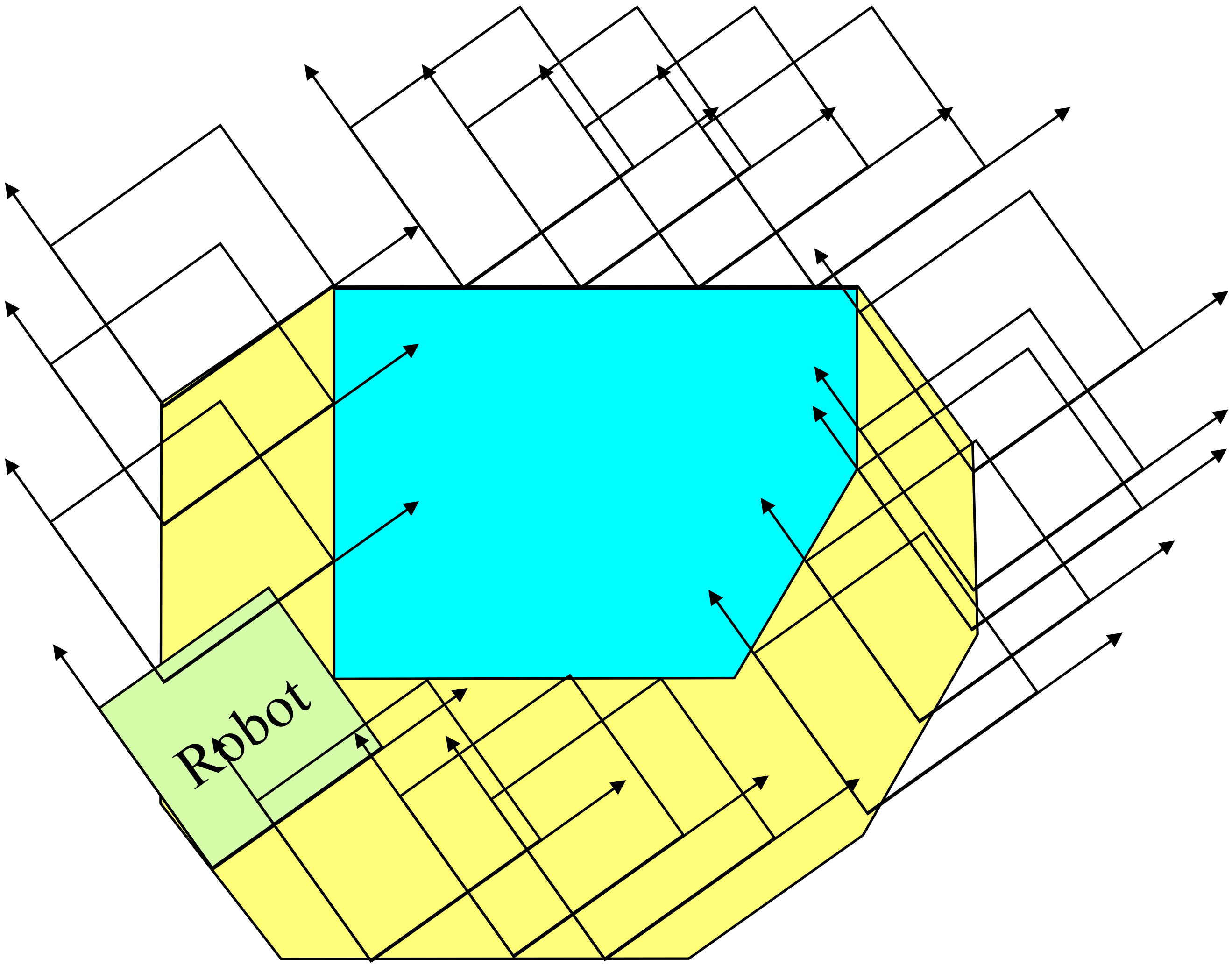
Allowable Robot positions (no rotations)



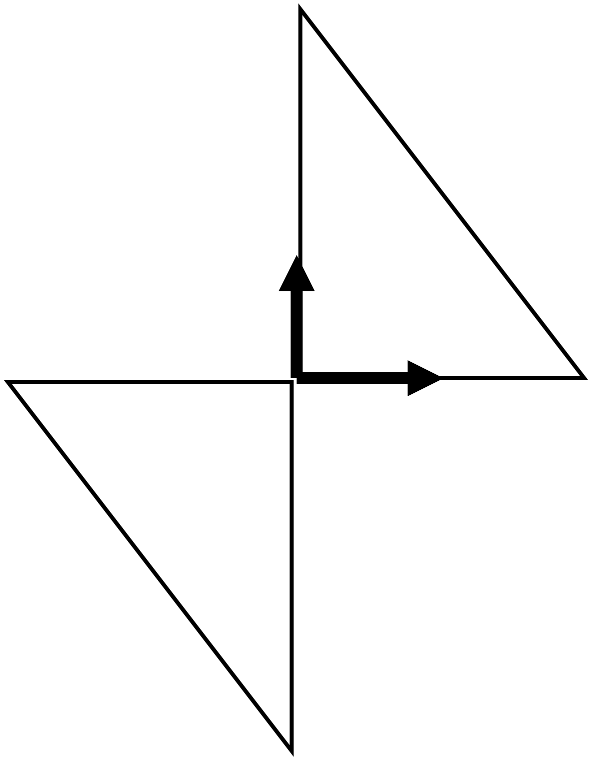
Allowable Robot positions (no rotations)



Allowable Robot positions (for some robot rotation)

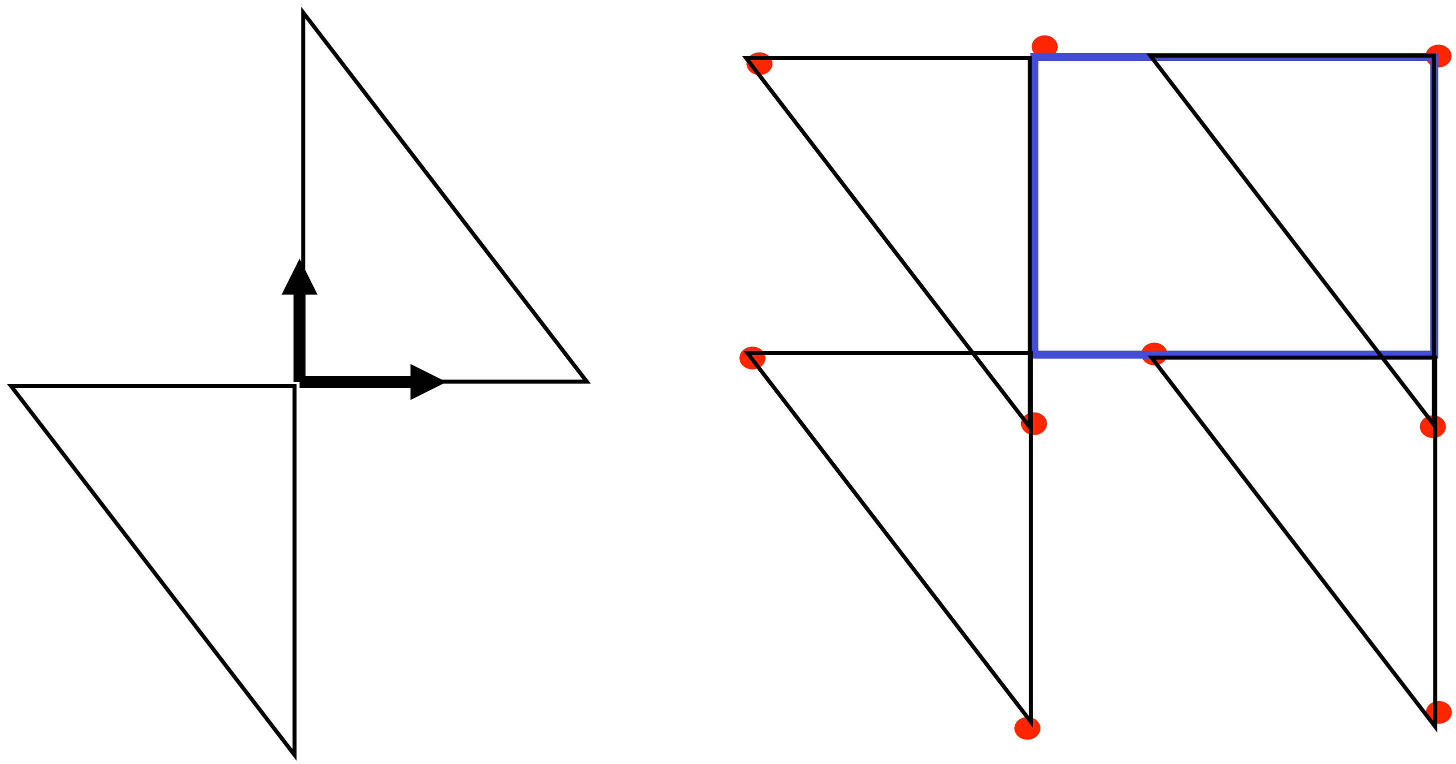


C-space Algorithm



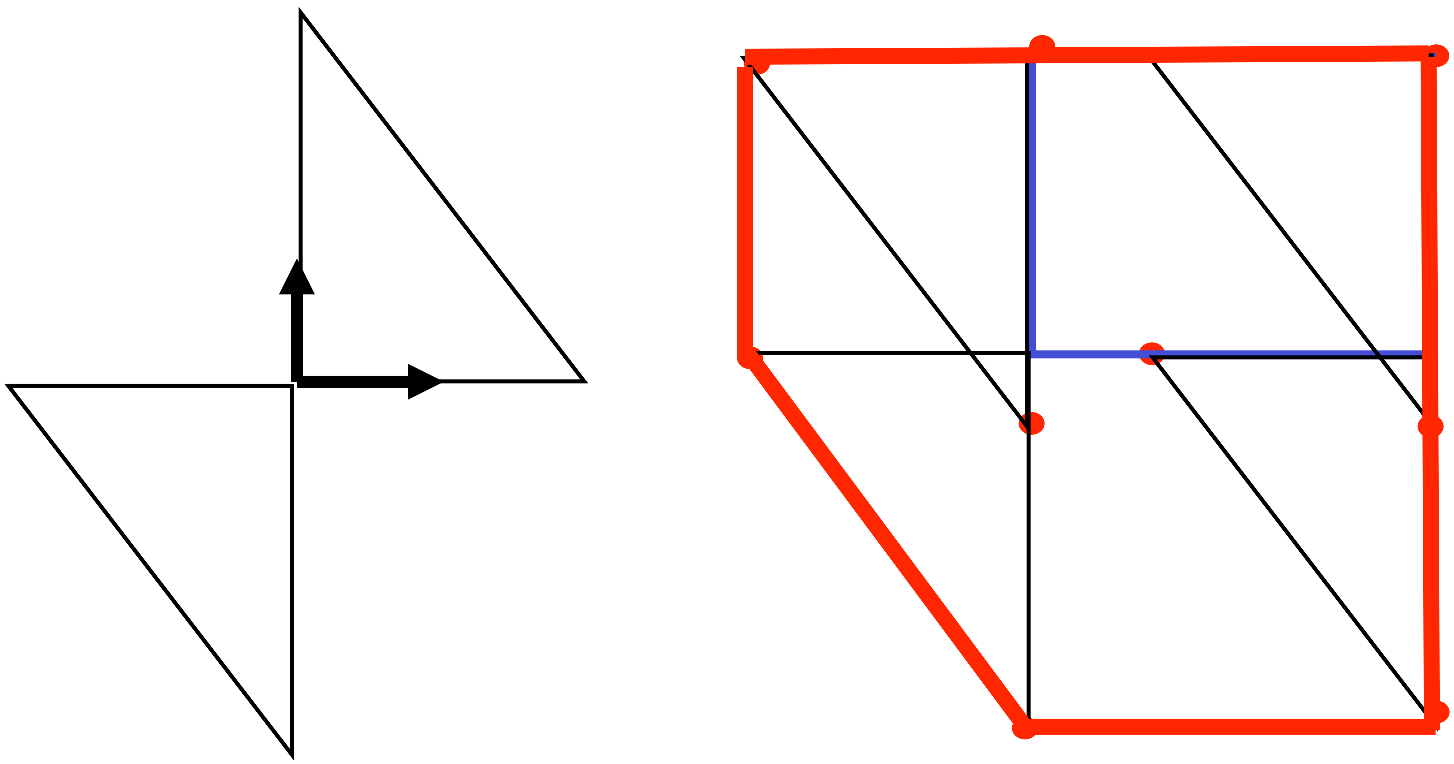
Step 1: Reflect Robot

C-space Algorithm



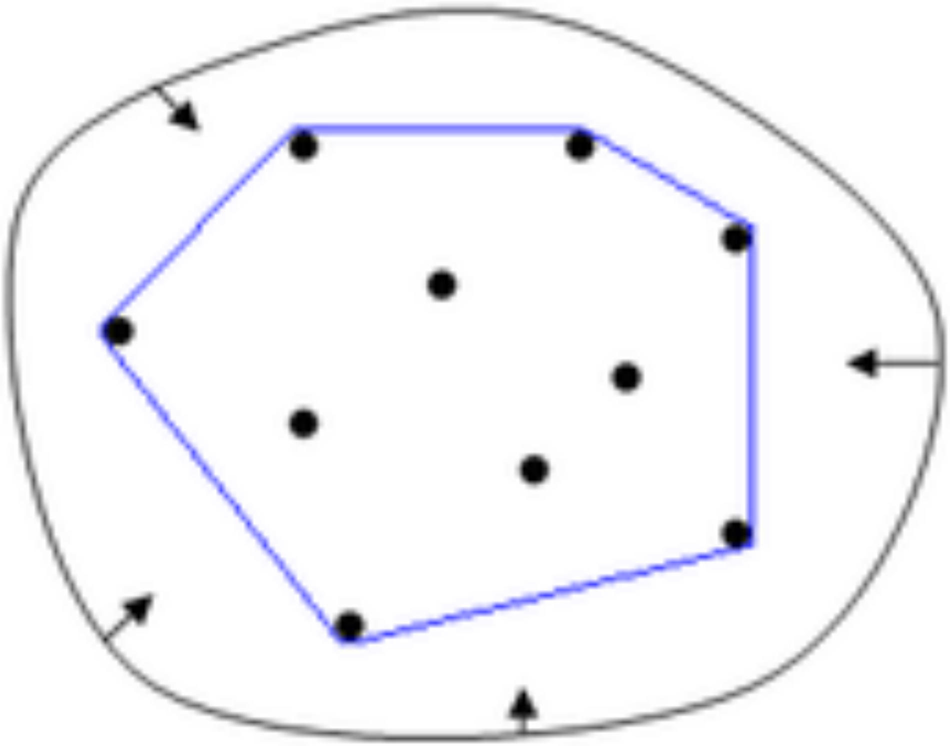
Step 2: Vert (⊖Robot) ⊕ Vert (Obstacle)

C-space Algorithm

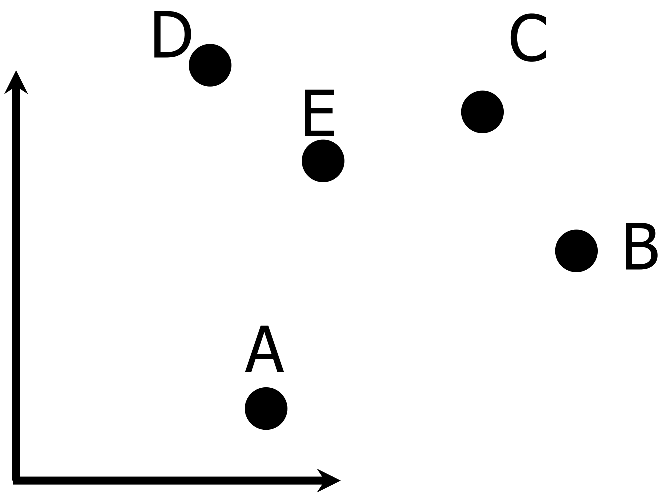


Step 3: ConvexHull (Vert (- Robot) + Vert (Obstacle))

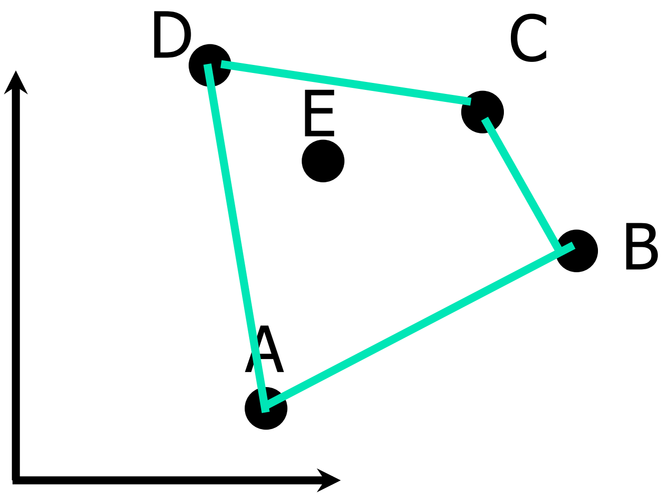
Convex Hull Algorithm



Convex Hull Algorithm

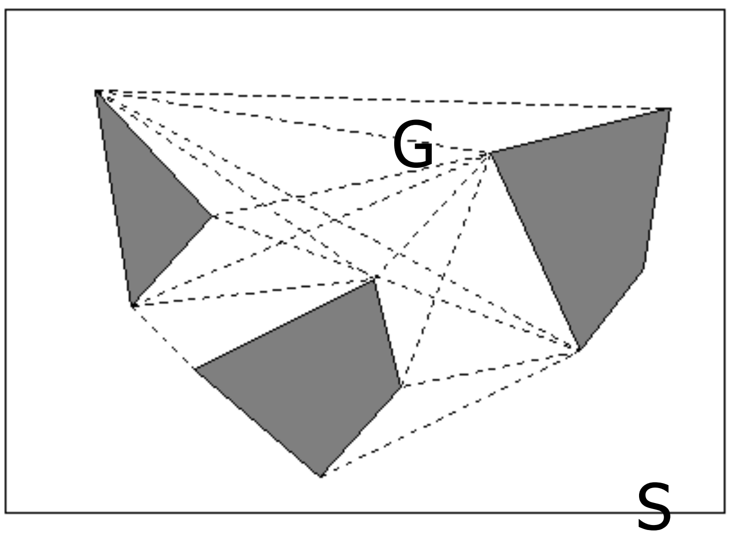


Convex Hull Algorithm



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?