Motion planning: Beyond Navmeshes

CMSC425.01 Spring 2019

Administrivia

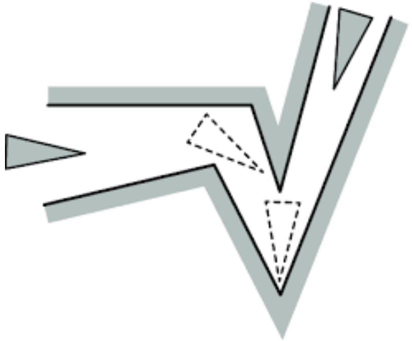
- Hw2 questions?
- Project 2a questions?
- Exam review on Thursday
- Grading push this week

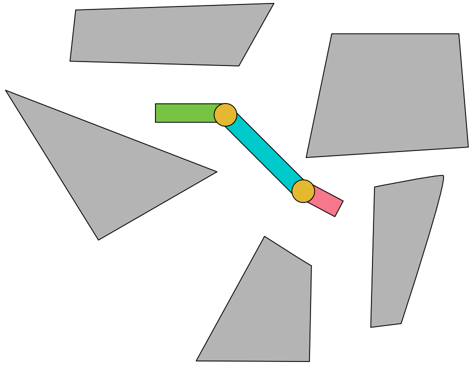
Networking and motion: Foreshadowing

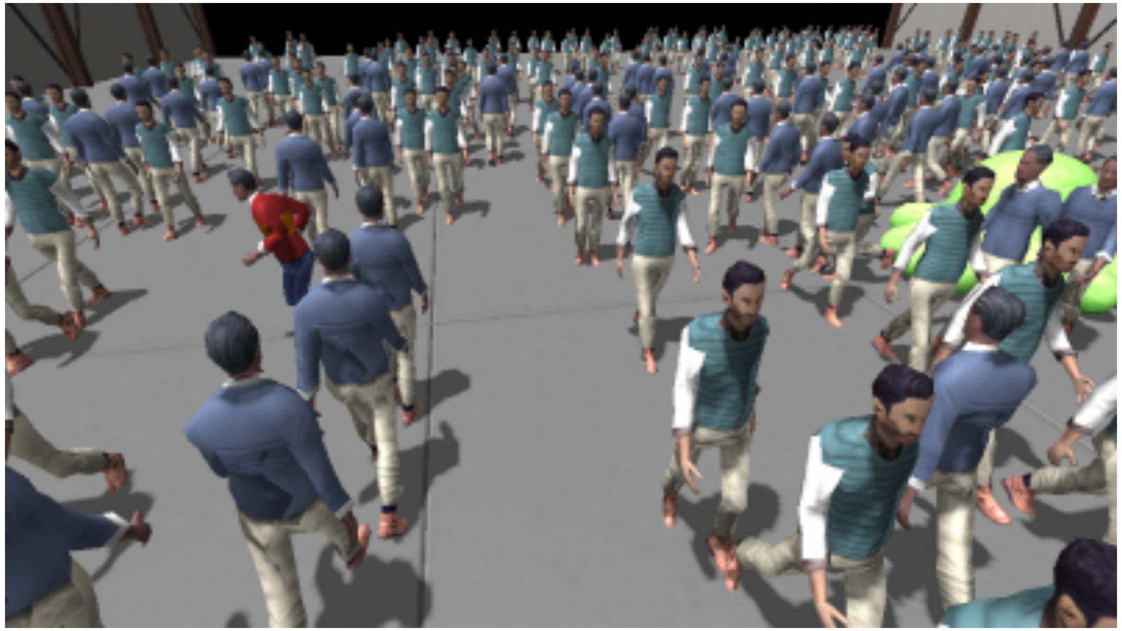
- CMSC 425: Lecture 22 Multiplayer Games and Networking
- https://www.cs.umd.edu/class/spring2018/cmsc425/Lects/lect22-multiplayer.pdf
- Ideas:
  - Topology: Centralized server vs. peer to peer?
  - Transport level: TCP (validated) vs. UDP (unvalidated)?
  - Game objects: How distribution game object data?
What data needs to be where?
Central database?
  - Latency: Concern for network delays?
Realtime game: predict motion of other players?

Navigation problems

- Navigating from place to place
- Dense crowd navigation
- Coordinated team movement
- Pursuit
- Moving complex/articulated shape
  - Piano movers problem(rigid)
  - Skeleton (articulated)

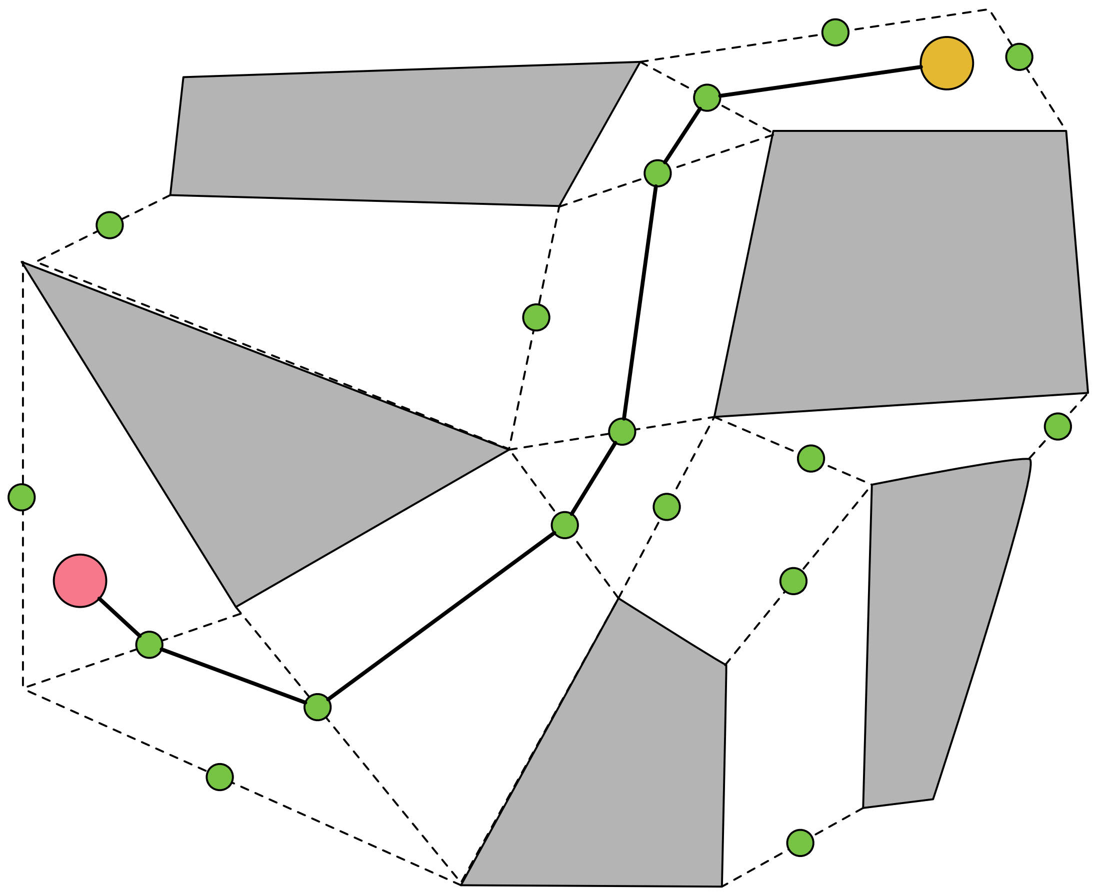






Navmesh

1. Mark navigable space
   - Use agent height/width/slope
2. Triangulate navigable area
   - Tile with triangles
3. Connect with graph
   - Connect in and out points
4. Search with algorithm
   - Dijkstra's or A*



Review: smoothing bounding

- Step 2: Simplify boundaries
  - Simplify polygon "map"

- Recursive refinement of straight line

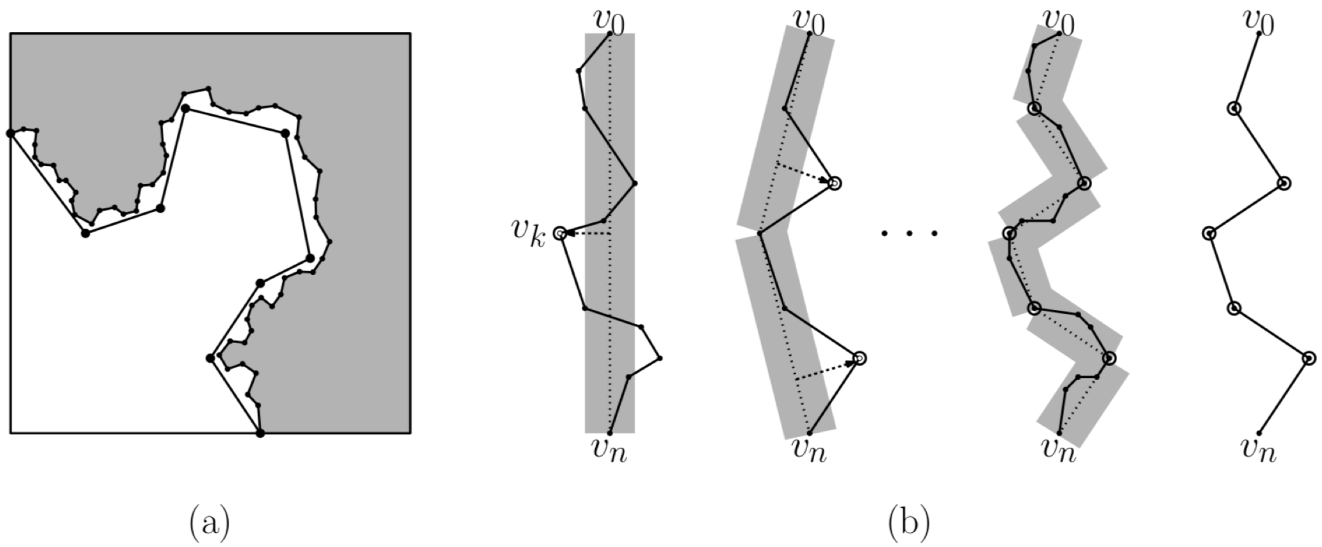
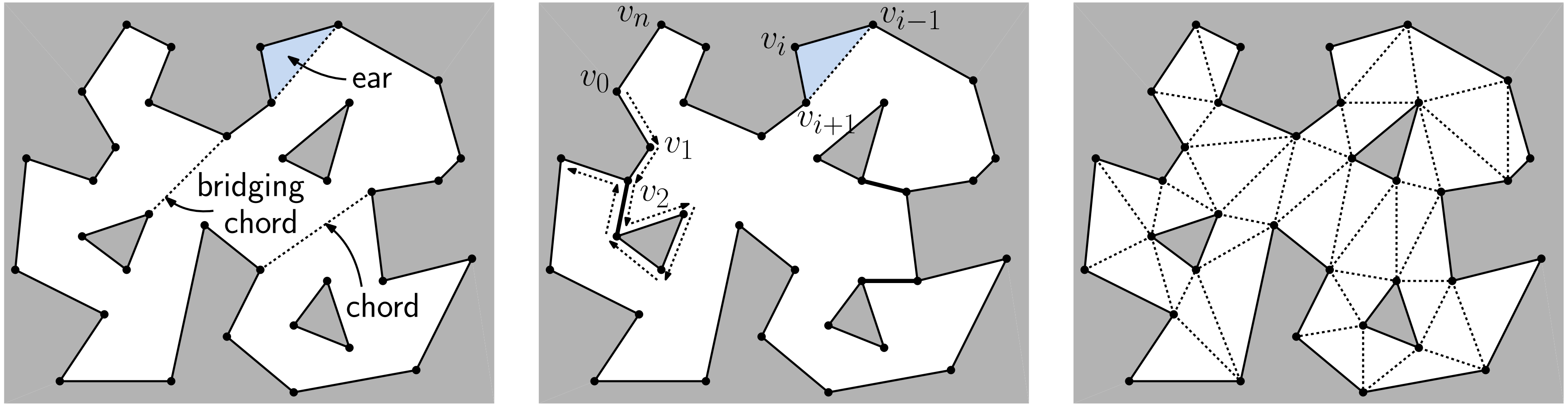


Fig. 3: The Ramer-Douglas-Peucker Algorithm.

Review of triangulation: how do efficiently?

- Step 3: Triangulate "map"
  - Cover with set of triangles

- Bridge holes
- Cut ears (!)



(a) (b) (c)

Beyond Navmesh

Navmesh: moving circle

1. Mark navigable space
   - Use agent height/width/slope
2. Triangulate navigable area
   - Tile with triangles
3. Connect with graph
   - Connect in and out points
4. Search with algorithm
   - Dijkstra's or A*

Generalizing: jointed polygon

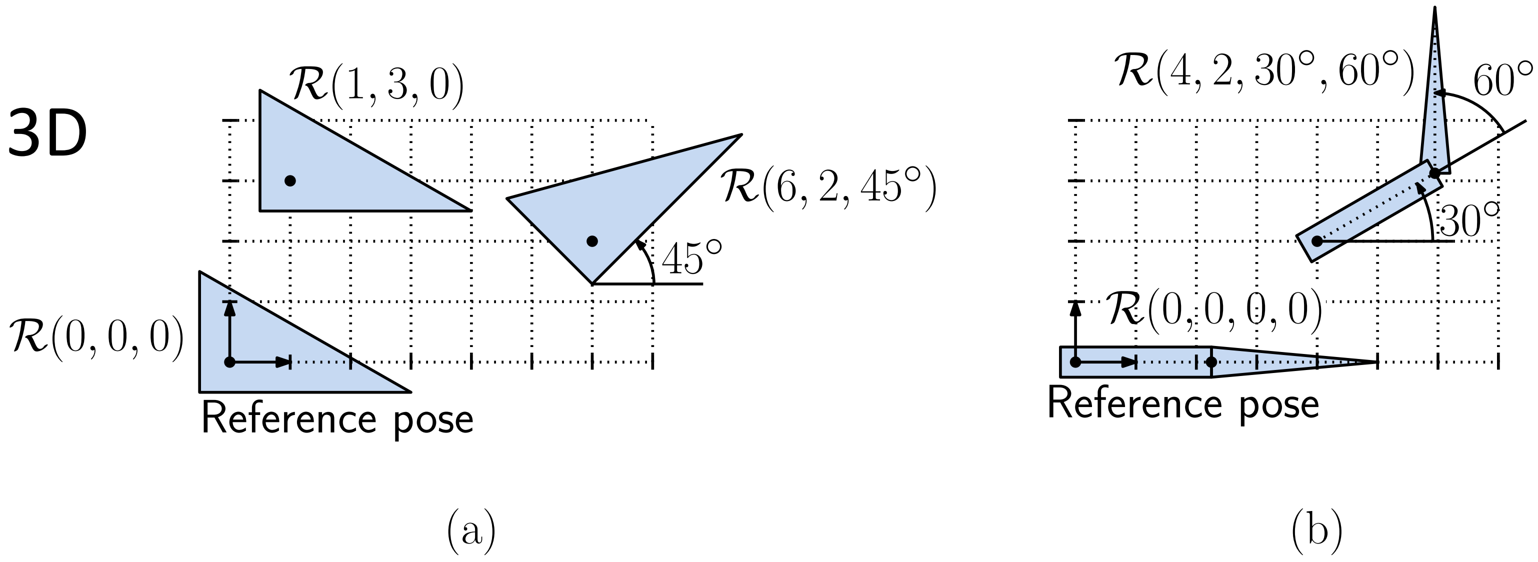
1. Define a navigable space
   - Jointed characters
   - Configuration space!
2. Find optimal paths in the space
3. Create a road network
4. Search the network

Defining robot configuration *R*

- Multiple degrees of freedom

- 3DOF – translate/rotate $\mathcal{R}(x, y, \theta)$ (region covered by robot)
- 4DOF – translate/rotate/bend $\mathcal{R}(x, y, \theta, \phi)$

- 6DOF – rigid object in 3D
- Human – 244

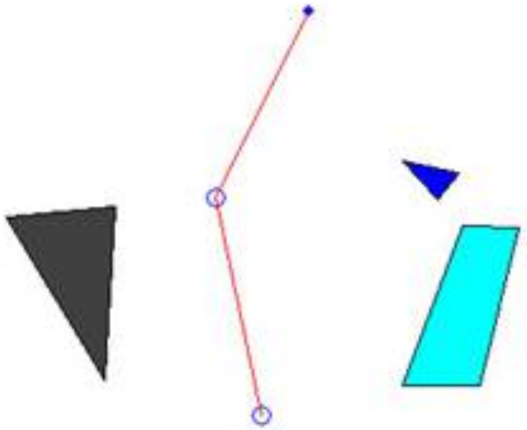


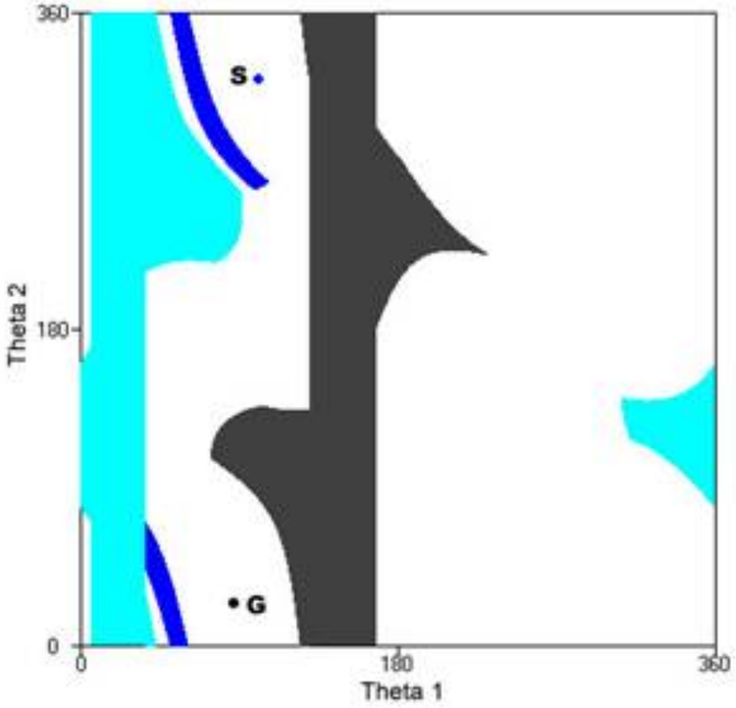
Defining workspace *S*

- Boundary of space + obstacles
- In same DOF space as robot

- Defines free and forbidden ranges of values of *R*

- $C_{free}(R, S)$
- $C_{forbidden}(R, S)$

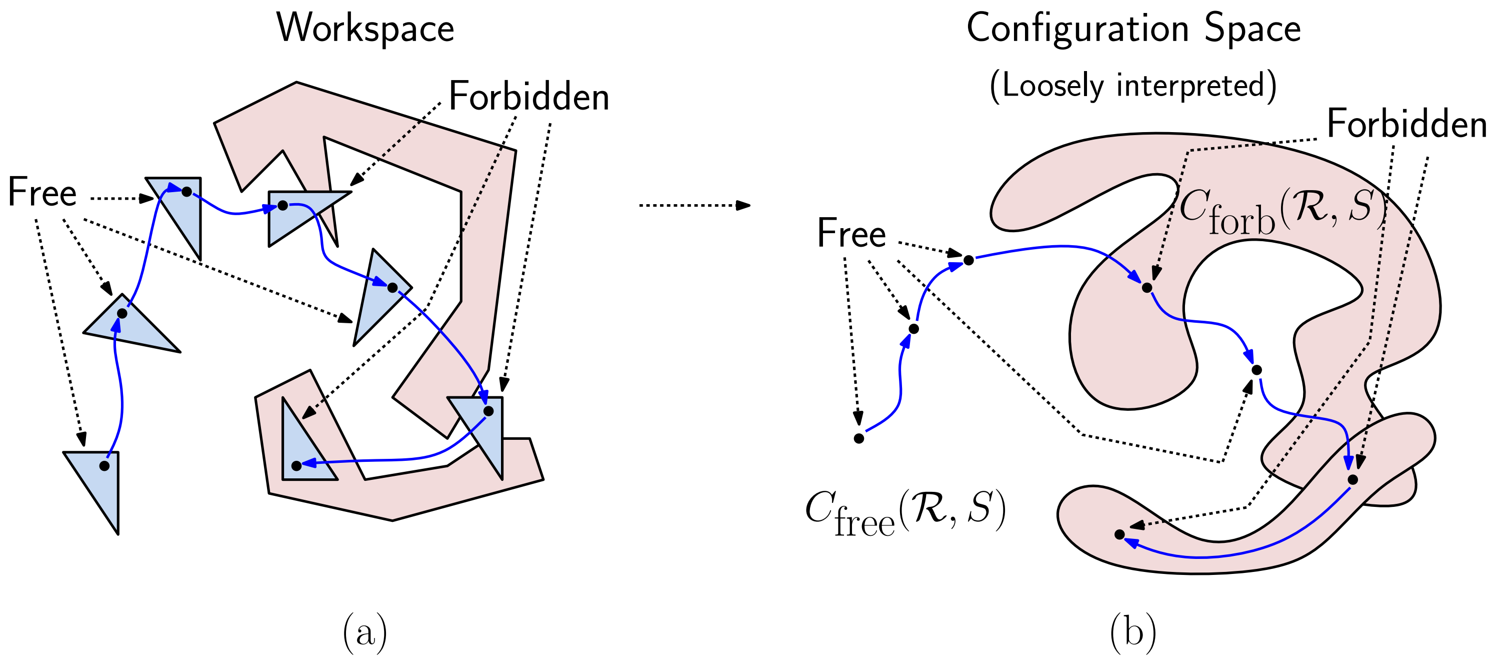




Motion planning in configuration space

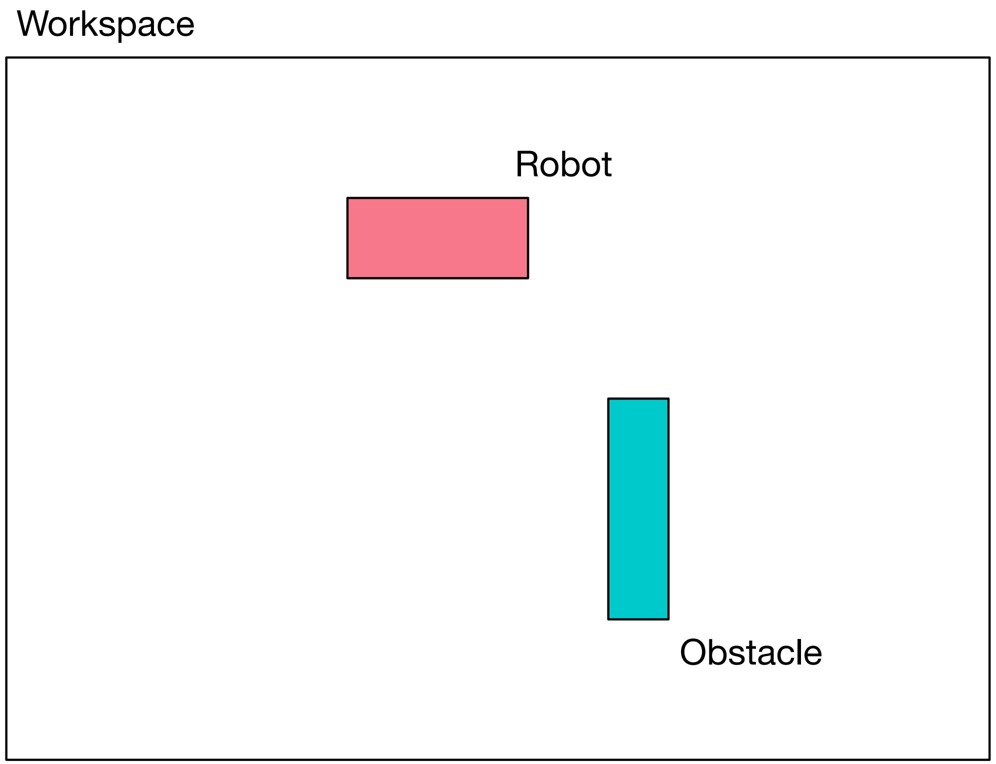
- Path from $s, t \in C_{free}(R, S)$
- if we have $\mathcal{R}(s) \rightarrow \mathcal{R}(t)$
- with all configurations in free space

- One path can be better than another based on length, maximum bend, etc

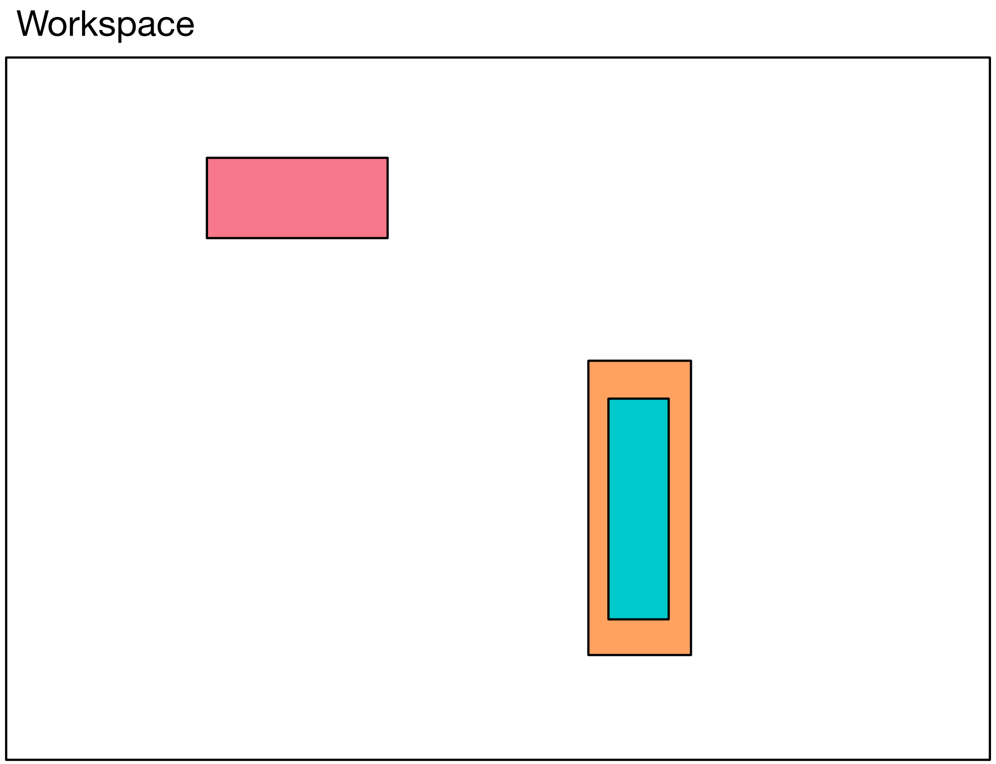


Building configuration space

Robot and obstacle

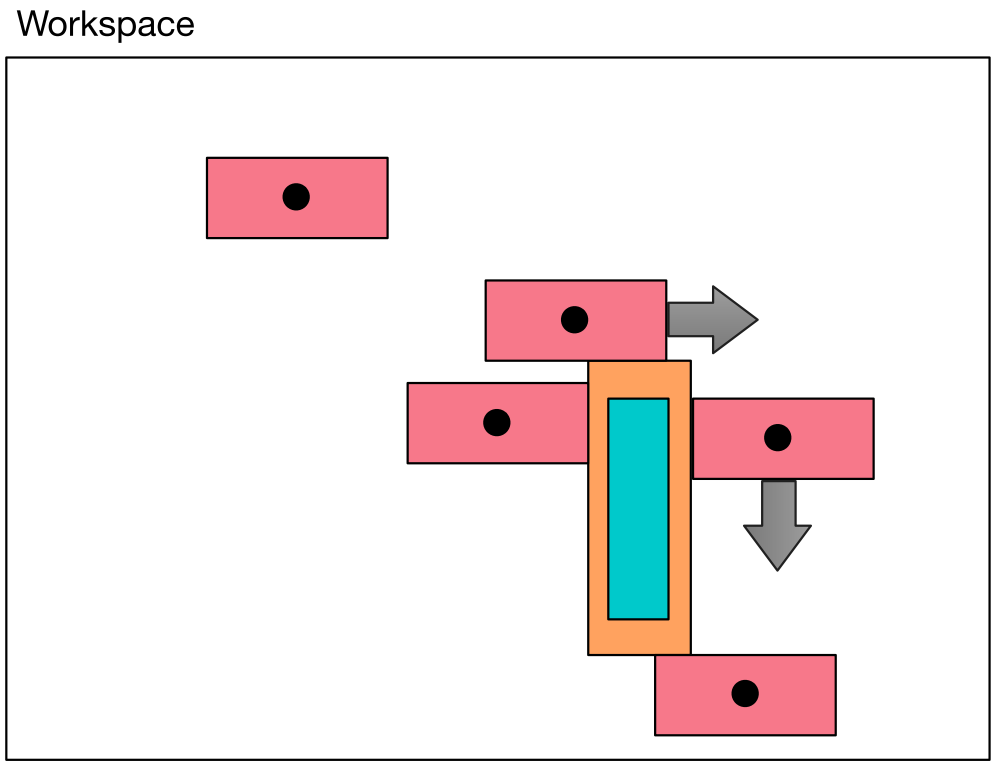


Step 1: Establish buffer distance

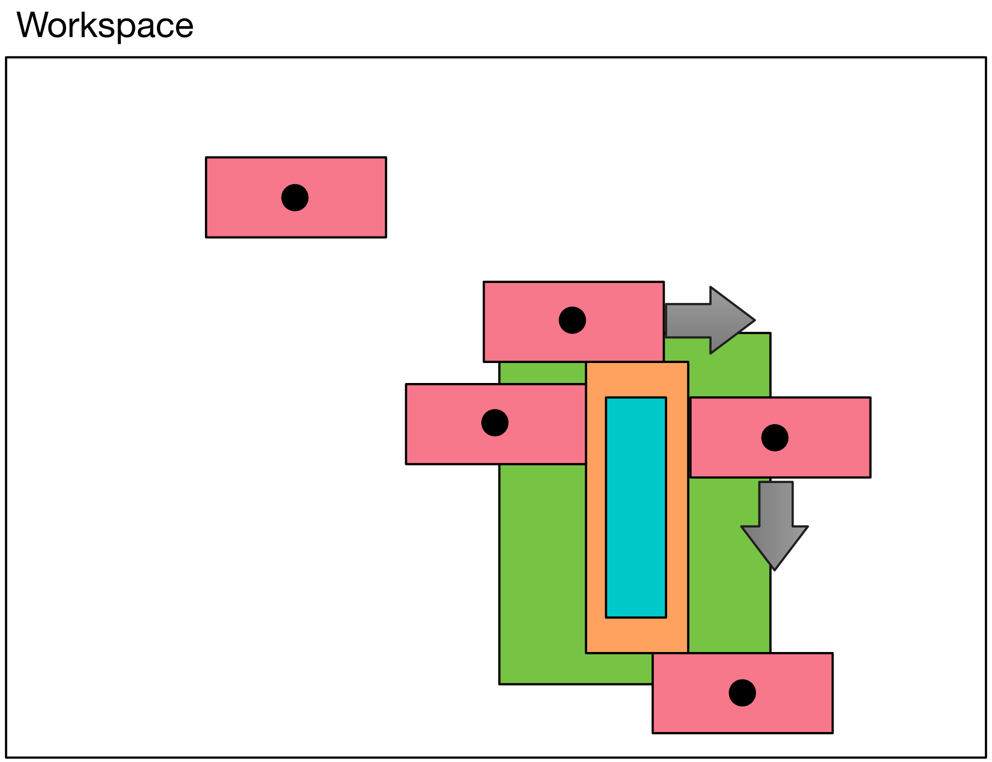


Building configuration space

Step 2: Move shape around obstacle

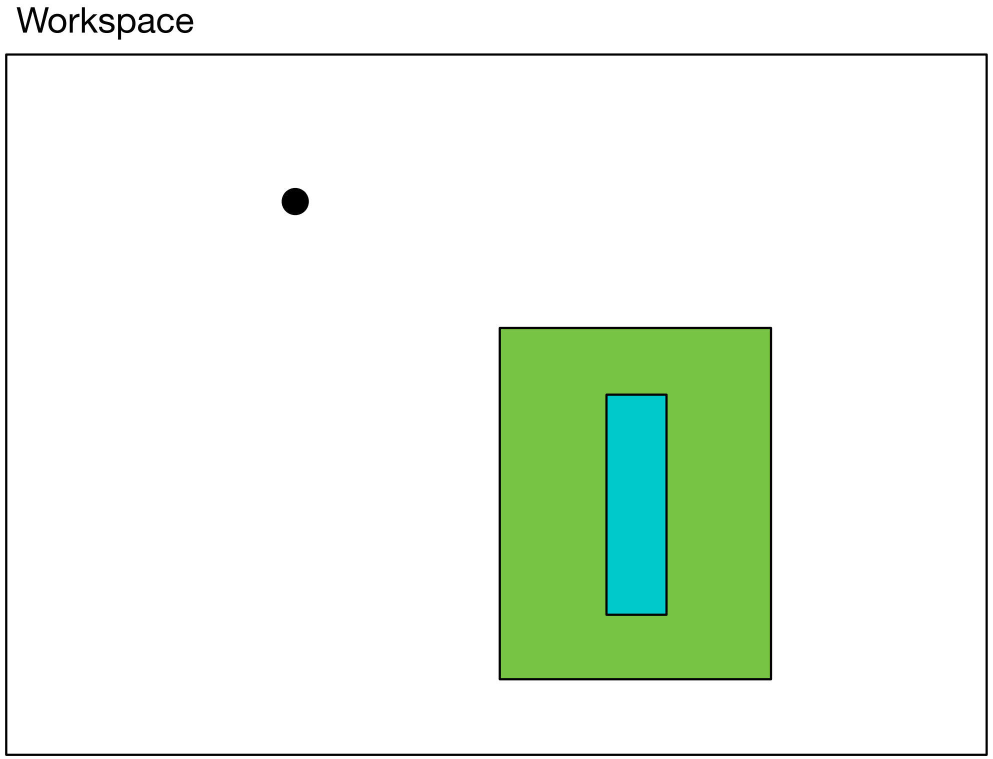


Step 3: Create extended obstacle (green) by midpoint



Building configuration space

Step 4: Reduce robot to point

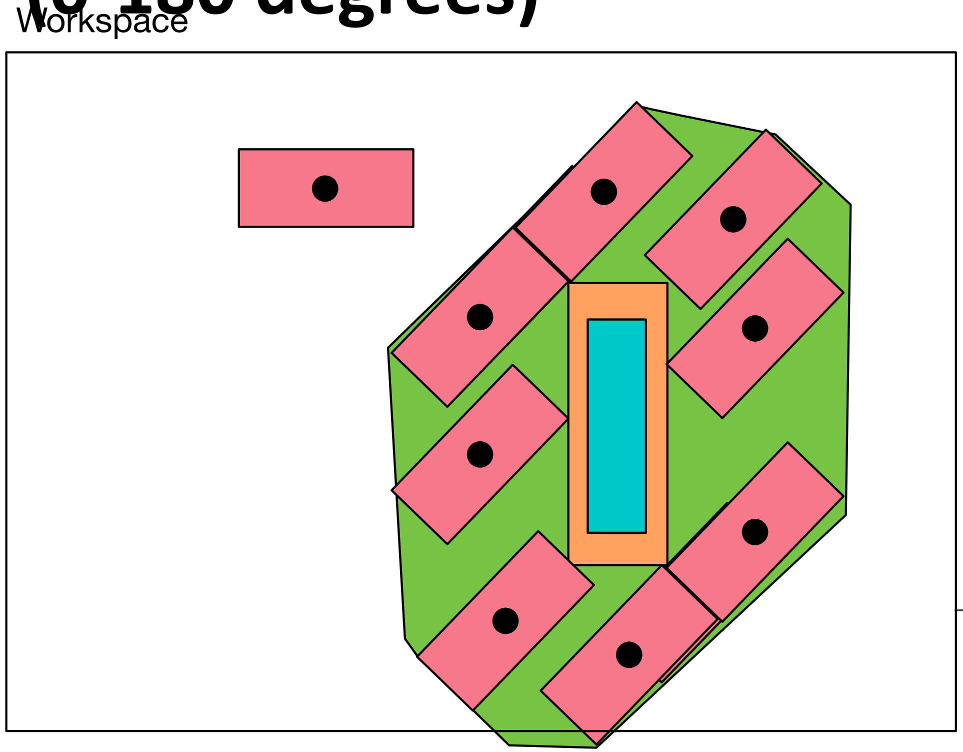


- Robot becomes point
- Obstacle become C-obstacle

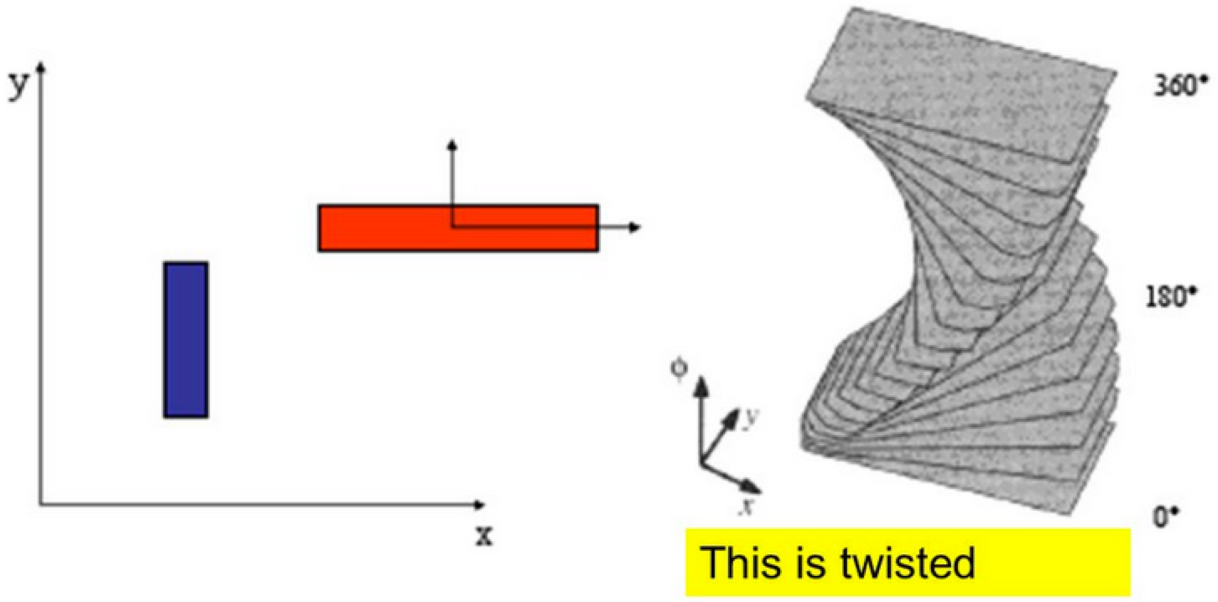
- Path finding reduces to finding a path for a single point around extended obstacles

In higher dimensions

Step 6: Rotate and repeat (0-180 degrees)



Creates solid in 3D space



- $\mathcal{R} = (x, y, \theta)$
- 3D space
- (Howie Choset CMU)

Creating in higher dimensional space

- Expensive!
  - 5DOF or 7DOF arm?
  - Robot base with arm? 10DOF

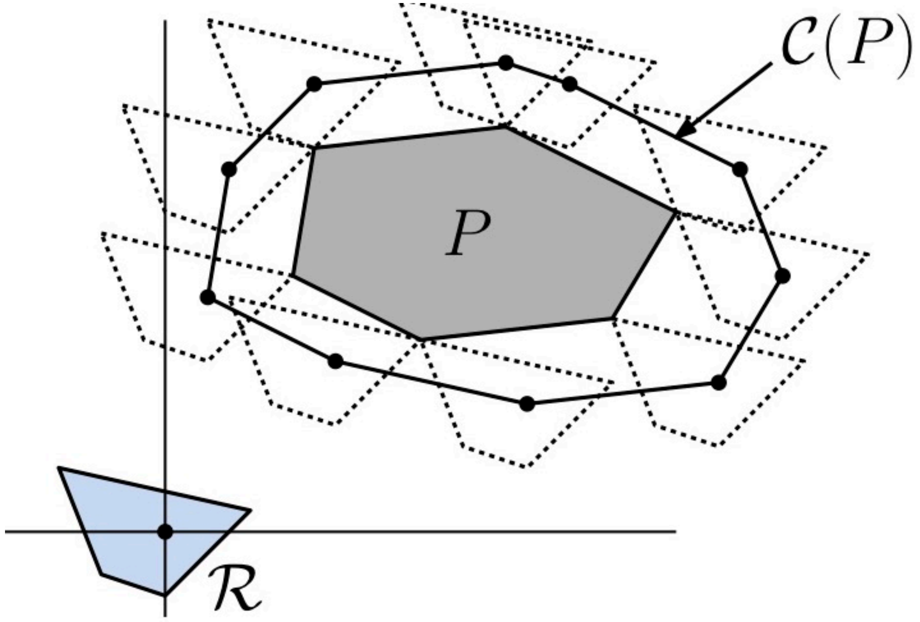
- Sample space, fit surface, approximate

(a) (b) (c) (d) (e)

Formalizing: Minkowski sums

- Motivation
- $\mathcal{R}(p)$ is region of $\mathcal{R}$ translated to p
- P is an obstacle region

- $C(p) = \{p : \mathcal{R}(p) \cap P \neq \emptyset\}$



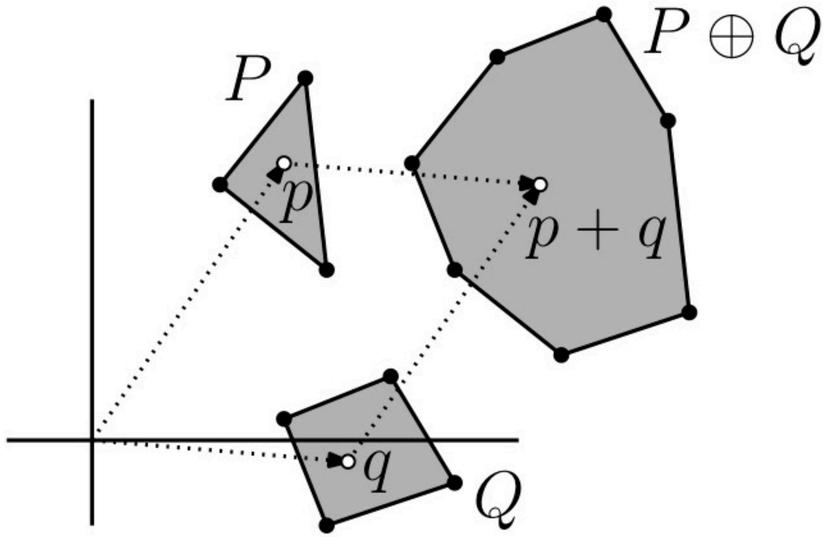
(a)

Definitions

- Minkowski sum
- $P \oplus Q = \{p + q : p \in P,\ q \in Q\}$

- Negated region
- $-P = \{-p : p \in P\}$

- Sum with point
- $P \oplus p = P \oplus \{p\}$



(b)

Claim: $C(P) = P \oplus (-\mathcal{R})$

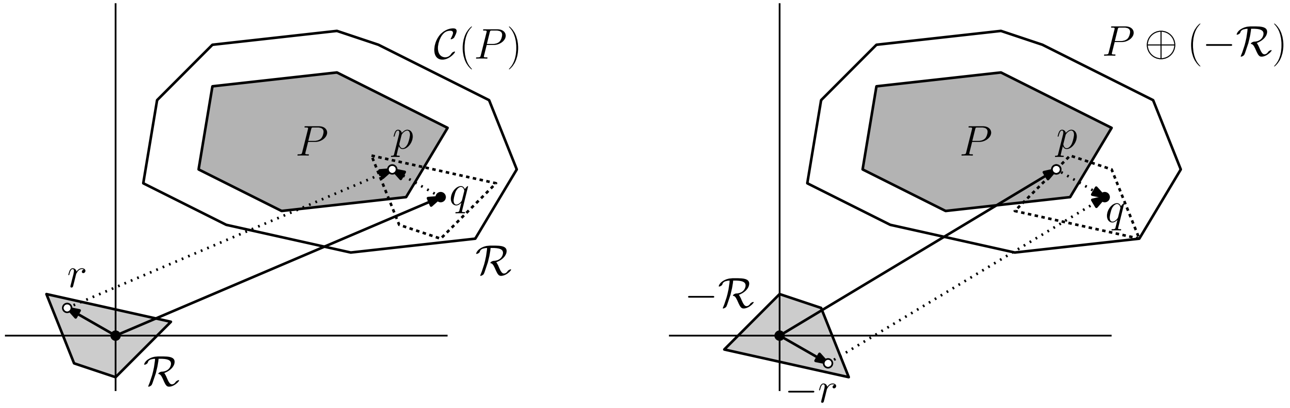
- “Proof”:

If robot R intersects obstacle P when at location q (R(q) in P)

Then we have for r in R that p = q+r

Then we can deduce q = p – r

The points q are those that compose C(P)



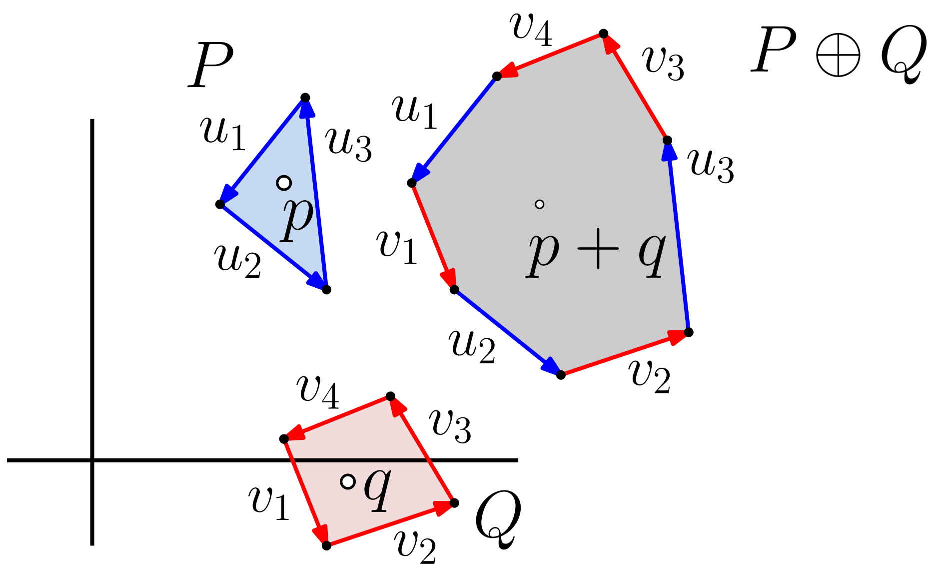
(a)

(b)

Algorithm: Computing Minowski sum

- Input: two polygons
- Output: polygon of M-sum

- Algorithm:
  - Take each edge in CCW direction
  - Sort by angle
  - Combine



Finding paths in polygonal configuration space

- Version 1: Navmesh
- Others?

Finding paths in polygonal configuration space

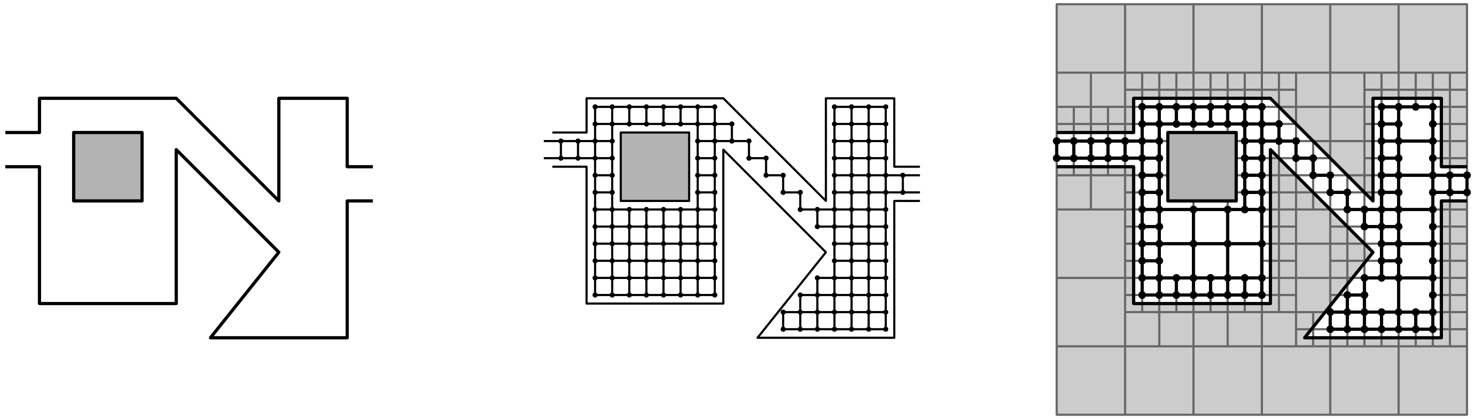
- Version 1: Navmesh
- Others?

- Version 2: Game designer draws …

Finding paths in polygonal configuration space

- Version 1: Navmesh
- Others?

- Version 3: Grid

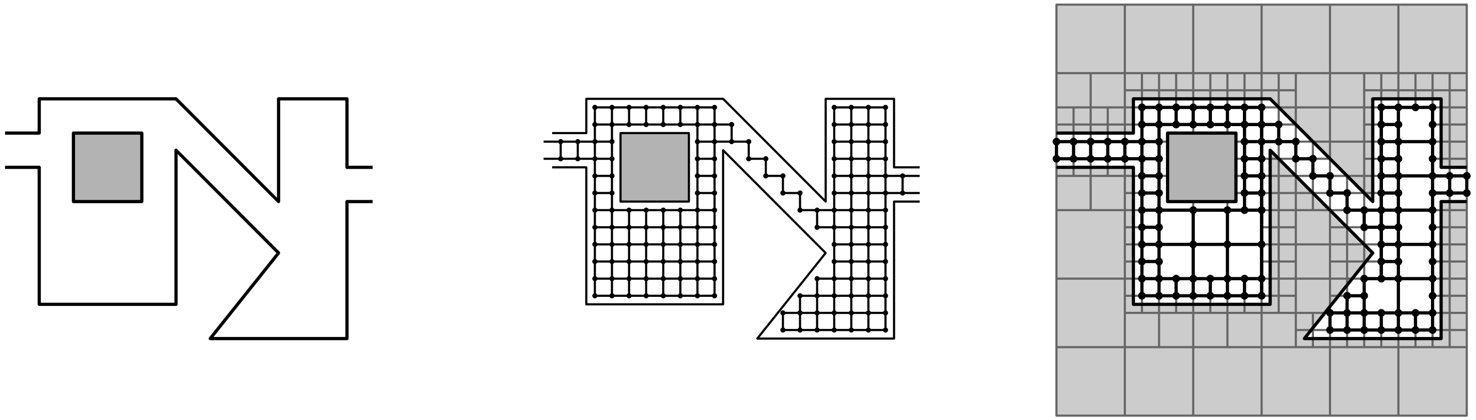


(a) (b) (c)

Finding paths in polygonal configuration space

- Version 1: Navmesh
- Others?

- Version 4: Multiresolution grid

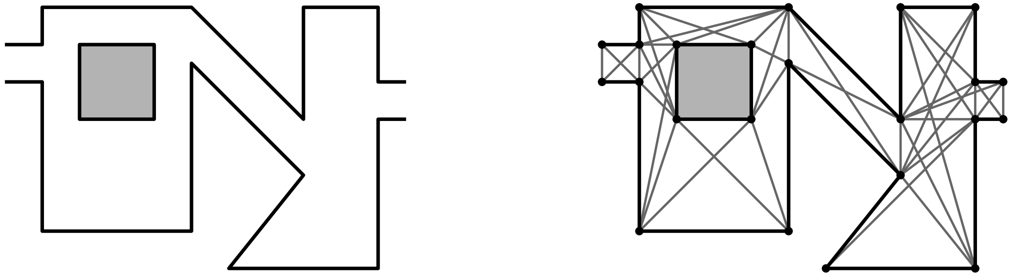


(a) (b) (c)

Finding paths in polygonal configuration space

- Version 1: Navmesh
- Others?

- Version 5: Visibility graph

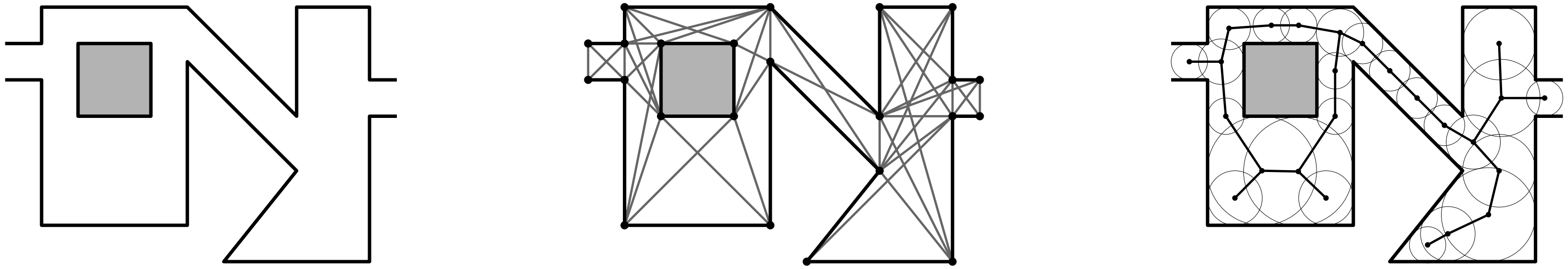


(a) (b)

Finding paths in polygonal configuration space

- Version 1: Navmesh
- Others?

- Version 6: Medial axis (c)

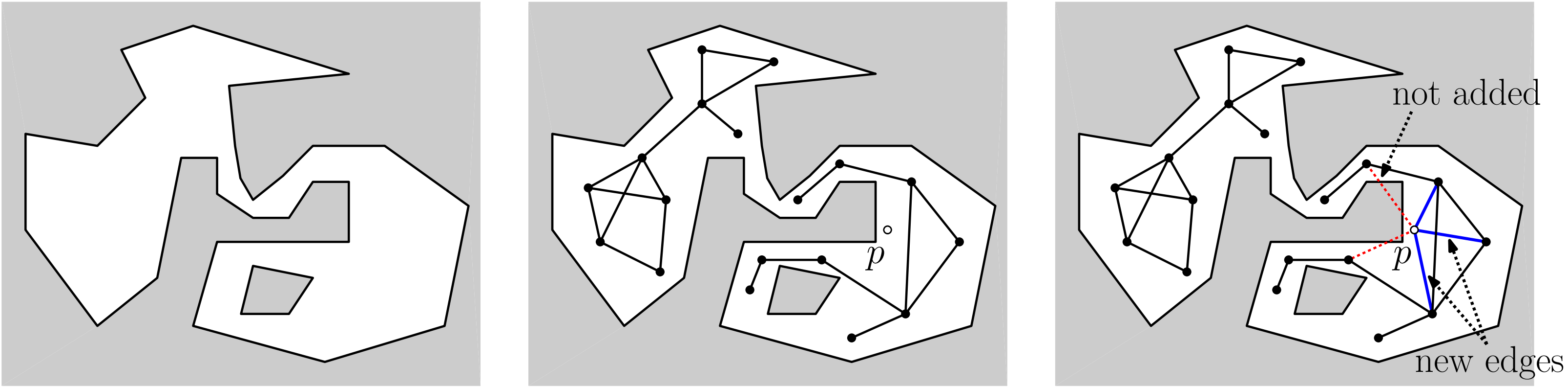


(a) (b) (c)

Finding paths in polygonal configuration space

- Version 1: Navmesh
- Others?

- Version 7: Randomized placement (sampling)

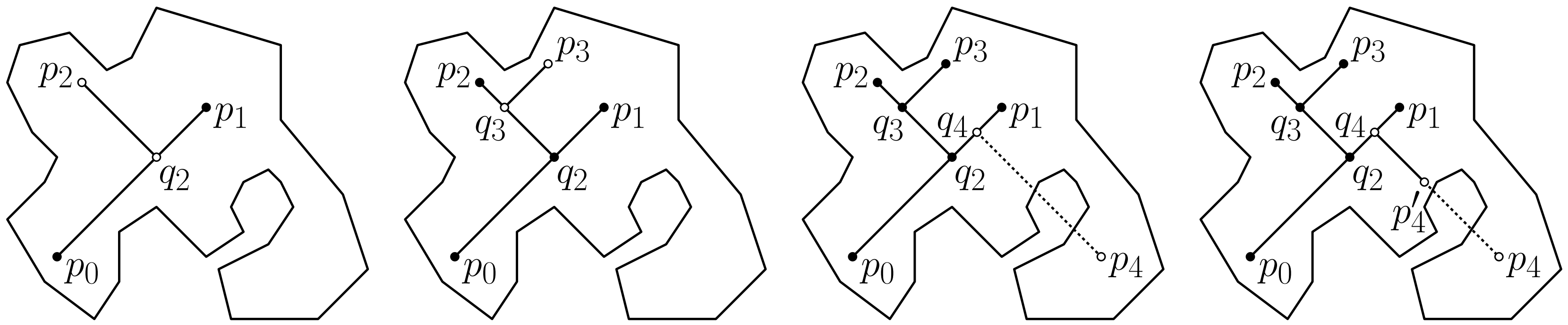


(a) (b) (c)

Finding paths in polygonal configuration space

- Version 1: Navmesh
- Others?

- Version 8: Rapidly-expanded Random Trees (RRTs)



(a) (b) (c) (d)