Manipulation and Path Planning

15-494 Cognitive Robotics
David S. Touretzky &
Ethan Tira-Thompson

Carnegie Mellon
Spring 2010

Introduction

- How do we get from basic kinematics to actually *doing* something?
- Two kinds of manipulation/path planning problems, really the same thing:

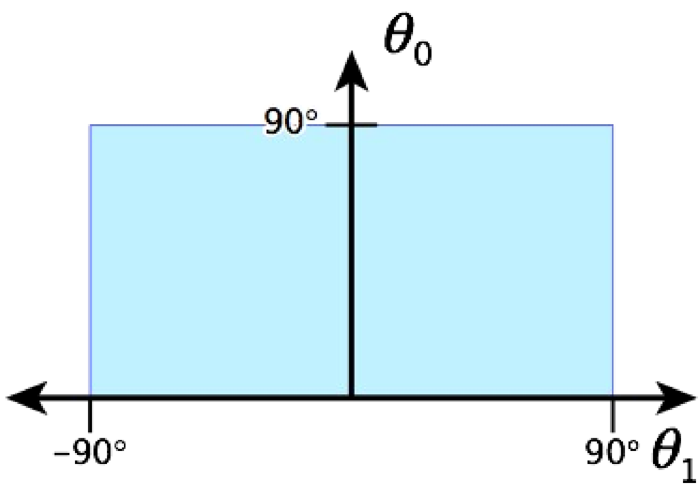
1) Navigation path planning (move the body)
2) Manipulation planning (move some other object, typically using the arm)

Manipulation Overview

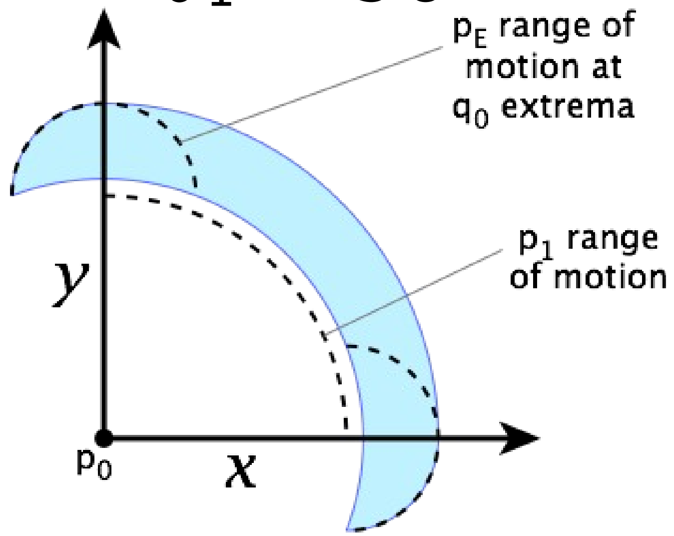
- Configuration space vs. work space
- Constraints
  - Form Closure vs. Force Closure
  - Grasp Analysis (Reuleaux’s Method)
- Path planning
  - Cspace, visibility graph, best first, RRT

Configuration Space vs. Work Space

- Consider a 2-link arm, with joint constraints:
  $0° < \theta_0 < 90°$, $-90° < \theta_1 < 90°$



Configuration Space: robot's internal state space (e.g. joint angles)



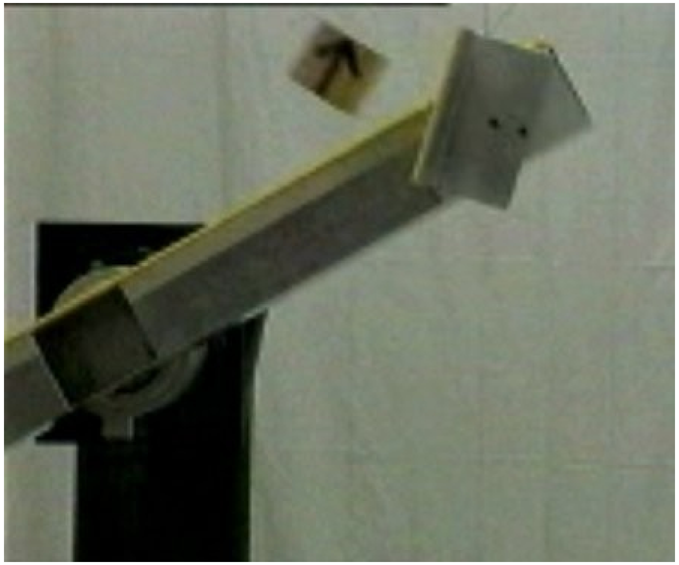
Work Space: set of all possible end-effector positions

Constraints

- Constraints can be your friend!
- Example: Use friction, gravity constraints to produce desired part trajectories
  - Upside: Exploit characteristics of the environment and the object itself to your advantage.
  - Downside: Requires planning and *accurate* modeling

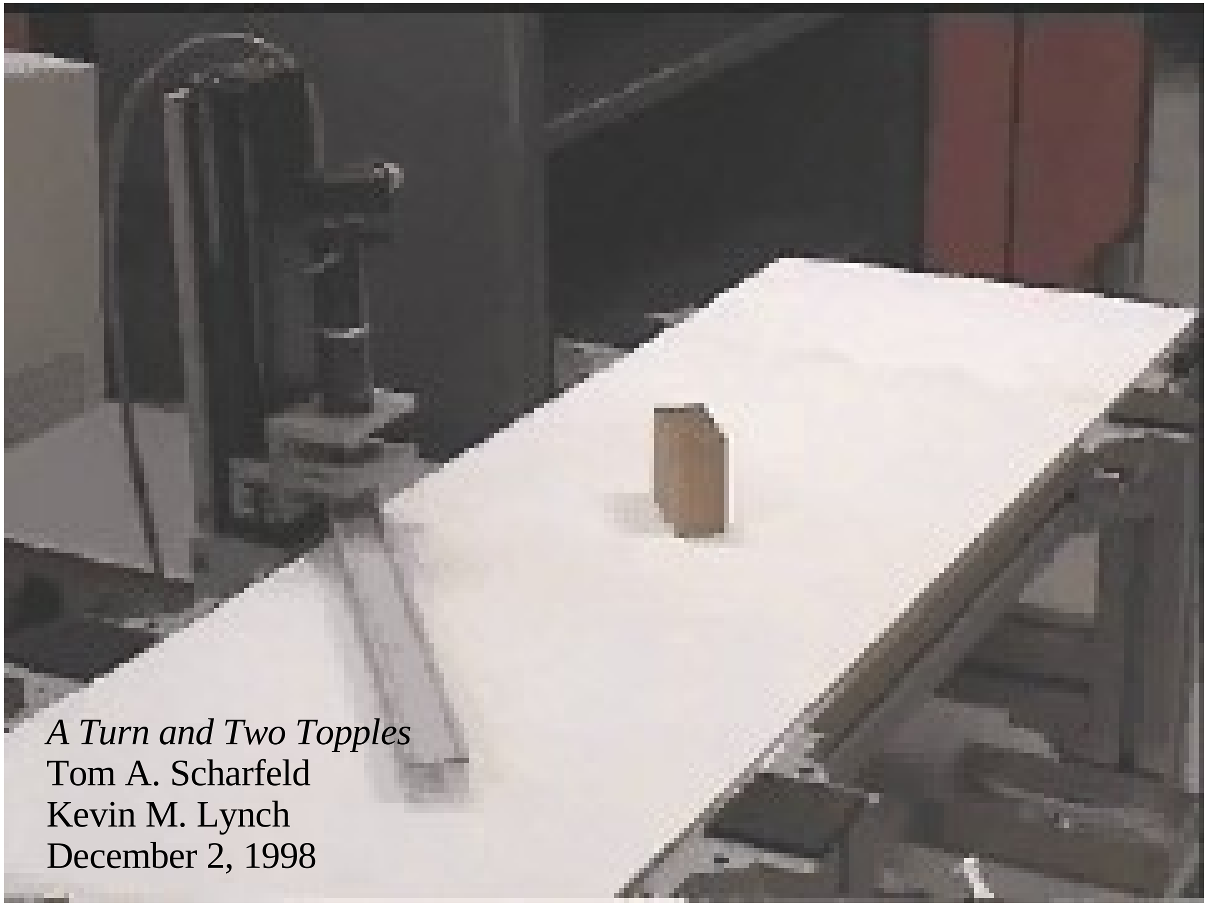
Constraints Are Your Friend

- Example: Throwing (Kevin Lynch)



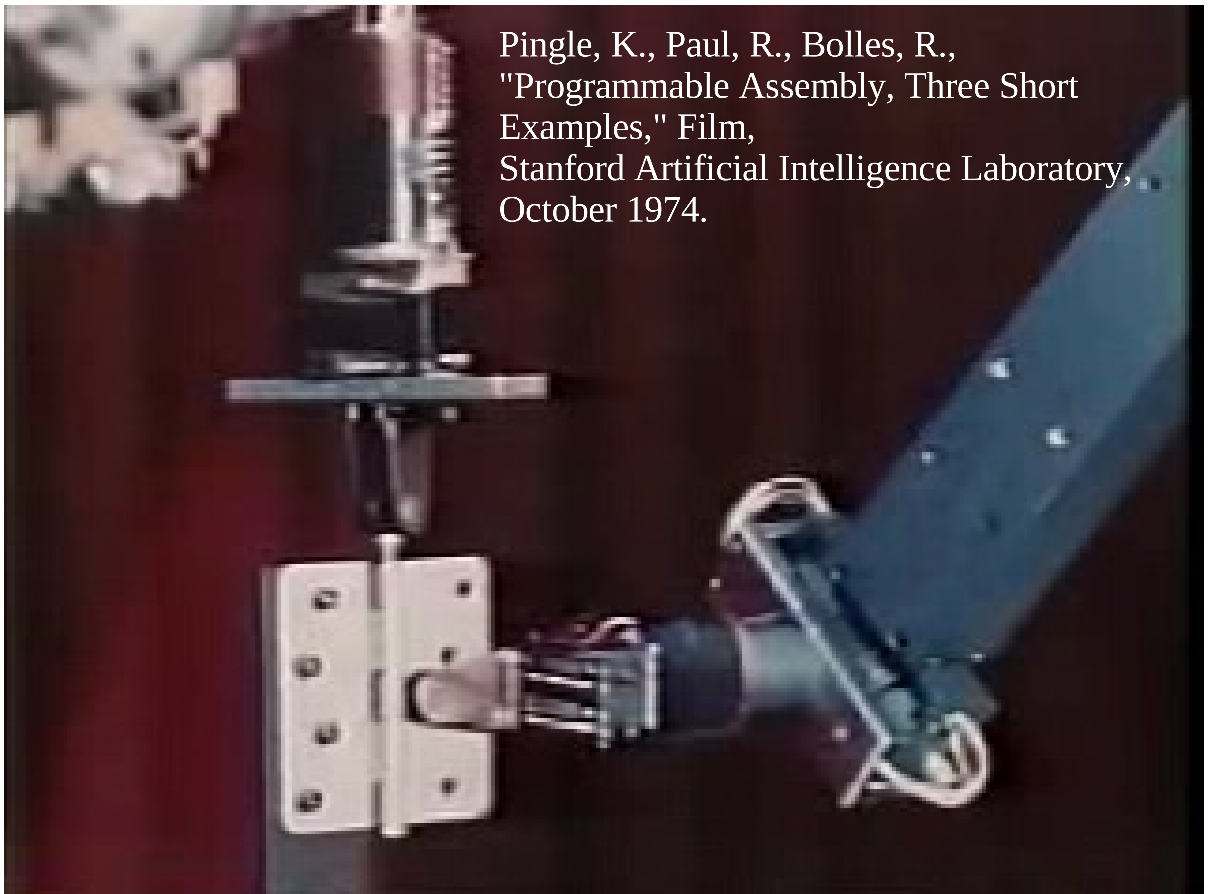
Constraints Are Your Friend

- 2 DOF Arm over a conveyor belt (2JOC)



Constraints Are Your Friend

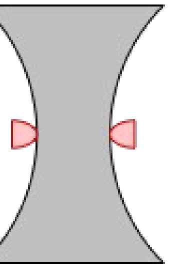
- Example: Hinge Assembly

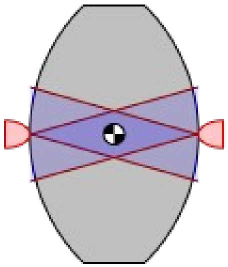


Grasping

- What does it mean to “hold” something?
  - *Form closure*: object is “secure” — can’t move without moving a contact point
  - *Force closure*: can apply any desired force
- Not necessarily the same thing — depends on your friction model (next lecture)

No friction:
Form closure, but
no force closure

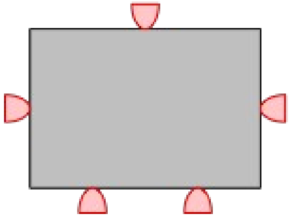




With friction:
Force closure, but
no form closure

Grasping

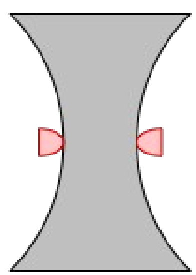
- Form closure is defined in increasing *orders*: position, velocity, acceleration, etc.
- Force closure does not have orders (you have it or you don't)
- Frictionless force closure equates to *first-order* (positional) form closure



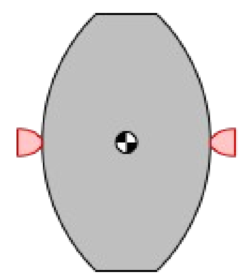
Example grasp with both force closure and first-order form closure, regardless of frictional model

Grasping

- Original examples do not have force closure
- Left figure can be moved infinitesimally up or down, although cannot be in motion vertically (so it has second-order form closure)



With no friction, neither example has force closure nor first-order form closure



Grasping

- What does it mean to “hold” something?
  - *Form closure*: object is “secure” — can’t move without moving a contact point
  - *Force closure*: can apply any desired force
  - *Equilibrium*: can resist environmental forces (gravity)
  - *Stability*: how much variance from the environment can be tolerated and still maintain equilibrium

Taxonomy of Contacts

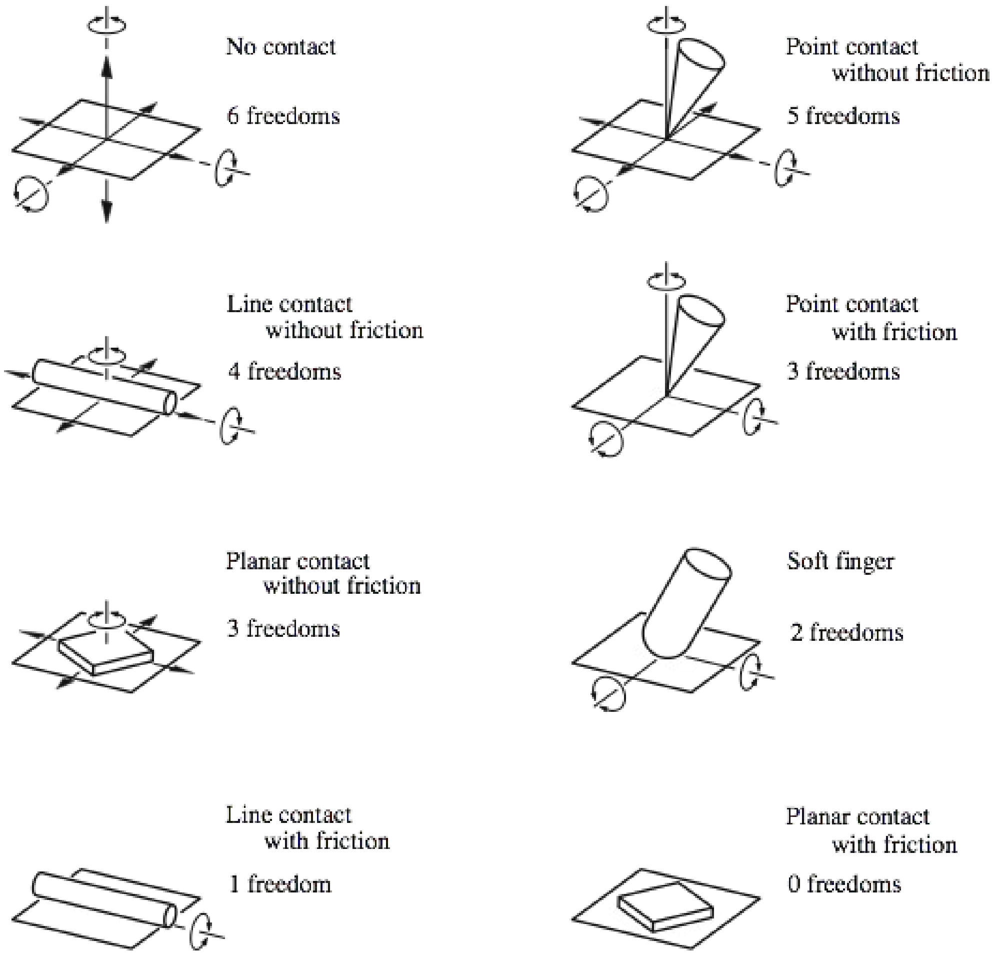
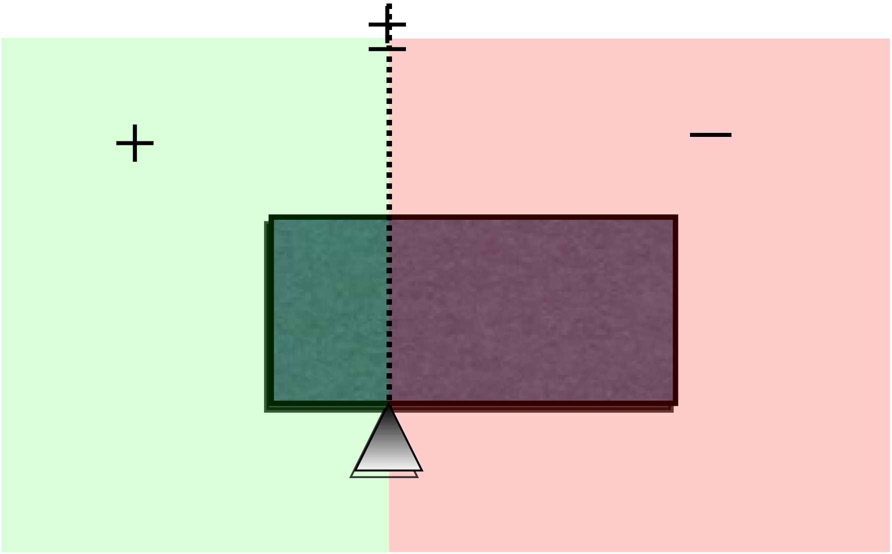


Figure 4.8 - Mason, Mechanics Of Robotic Manipulation

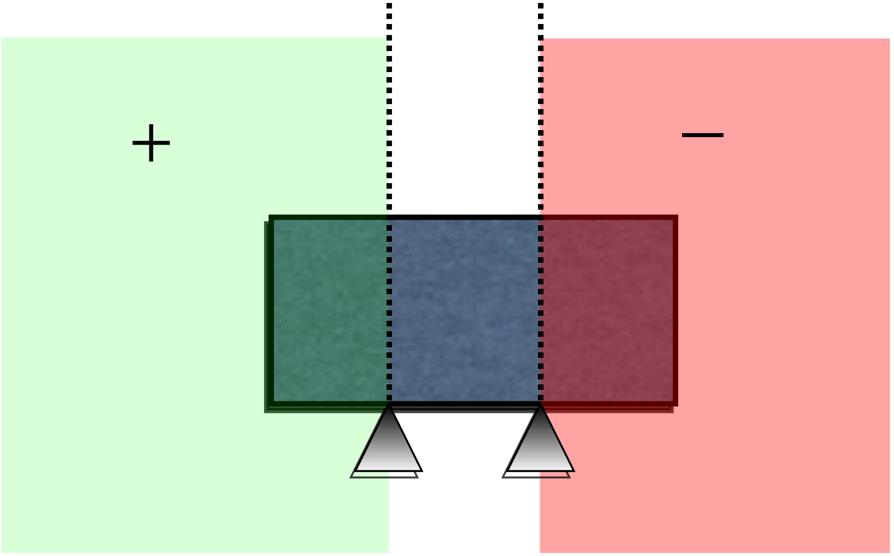
Grasp Analysis: Reuleaux's Method

- For each constraint, divide the plane into areas which can hold positive or negative centers of rotation (IC's - instantaneous centers)



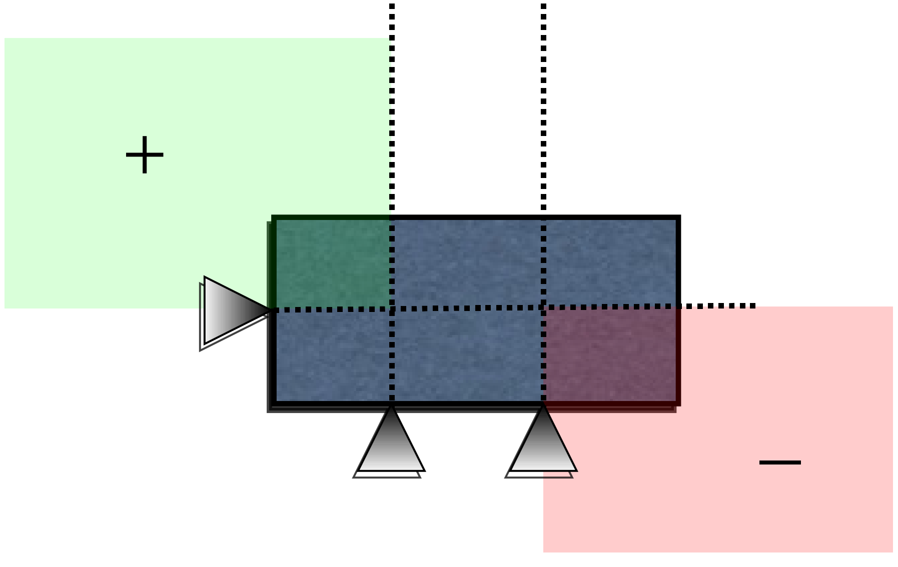
Grasp Analysis: Reuleaux's Method

- Intersect common regions



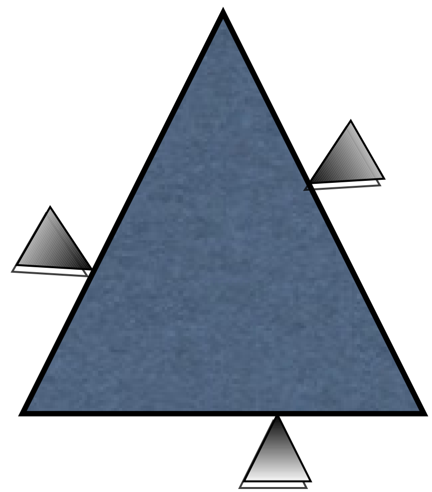
Grasp Analysis: Reuleaux's Method

- Intersect common regions



Grasp Analysis: Reuleaux's Method

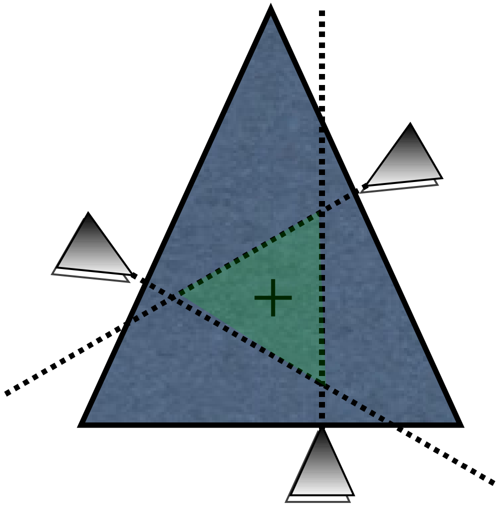
- Another example:



- Is this completely constrained?

Grasp Analysis: Reuleaux's Method

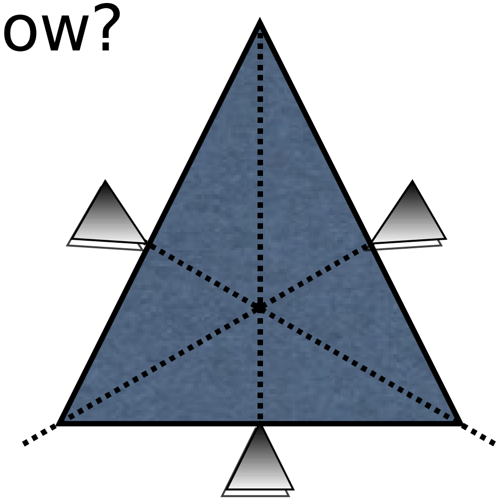
- Another example:



- Can spin counter-clockwise around area in the middle — but not clockwise!

Grasp Analysis: Reuleaux's Method

- How about now?



- Common intersections may indicate, but *do not guarantee*, that rotation is possible

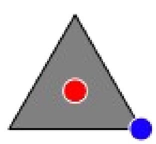
Grasp Analysis: Reuleaux’s Method

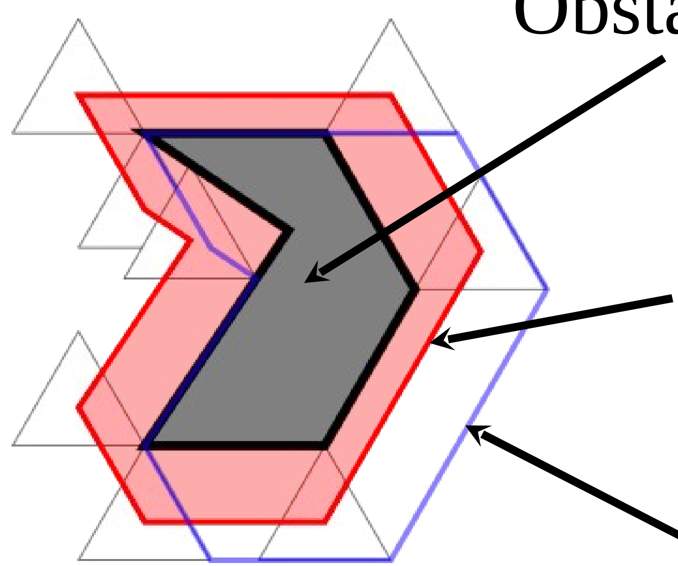
- Reuleaux’s Method is good for humans, not so good for machines
- Doesn’t extend to three dimensions
- Analytical solution would require a lecture unto itself
  - 16-741: Mechanics of Manipulation
  - Learn about screws, twists, wrenches, and moments

Motion Path Planning

- The Cspace Transform: the set of configuration points around obstacles which would cause a collision

Notice how the Cspace formed by defining the origin of the robot in its center (red dot and outline) is merely a translated version of the Cspace formed by placing the origin at one of the robot's corners (blue dot and outline).





Motion Path Planning

- The Cspace Transform: the area around obstacles which would cause a collision with the robot

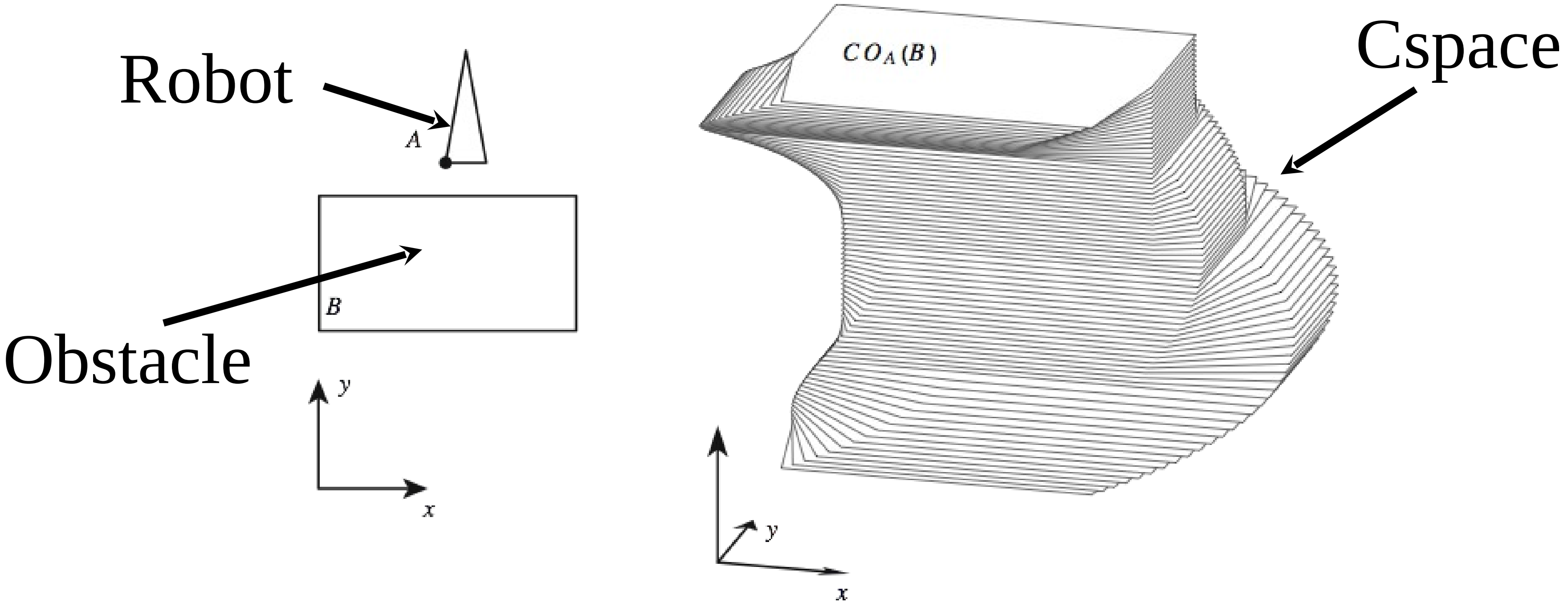


Figure 4.4 - Mason, Mechanics Of Robotic Manipulation

Motion Path Planning

- The Cspace Transform is not just for mobile robots' outer hulls!

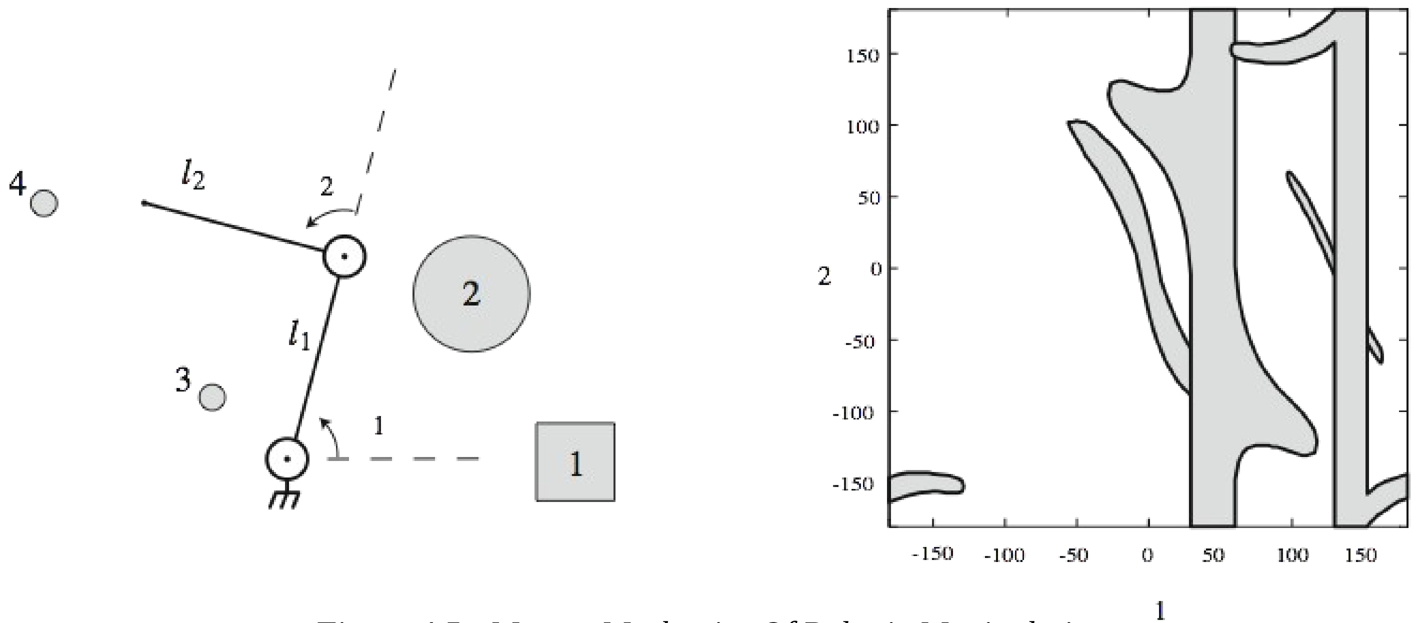


Figure 4.5 - Mason, Mechanics Of Robotic Manipulation

Motion Path Planning

- So, we know where we can't go, but how do we avoid it?
- Approach 1: Visibility Graph
  - Connect visible corners together, search the graph of connected edges

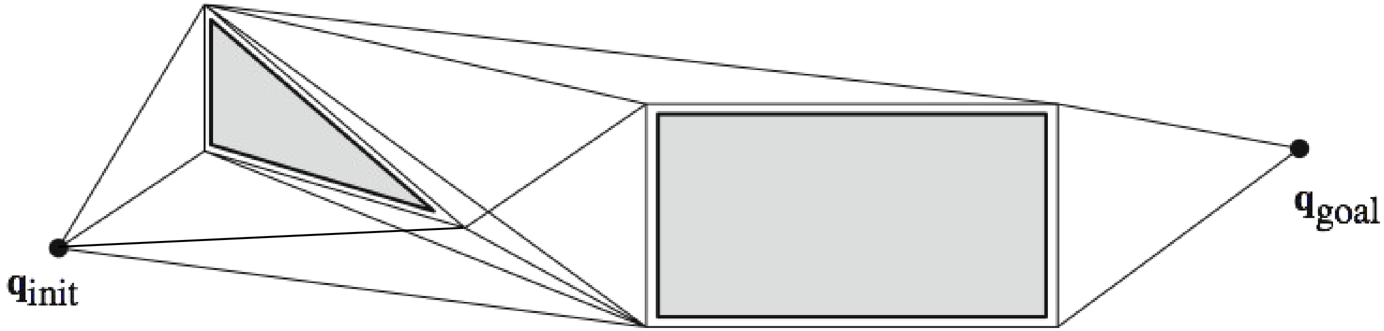
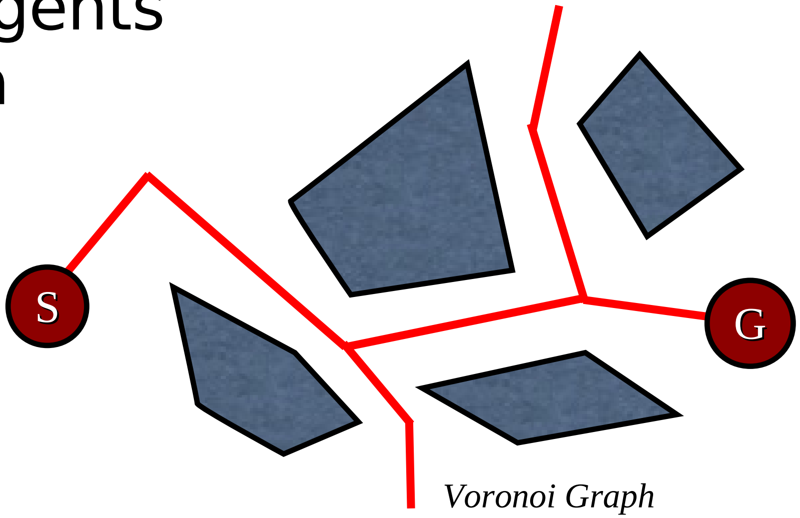


Figure 4.1 - Mason, Mechanics Of Robotic Manipulation

Motion Path Planning: Visibility Graph

- Great for 2 dimensions, but not for more
- Voronoi graphs are similar, and *have* been generalized to higher dimensions (Choset)
  - Instead of a graph of tangents between obstacles, use a graph of the midpoints
  - Fast search, safe path, but suboptimal distance

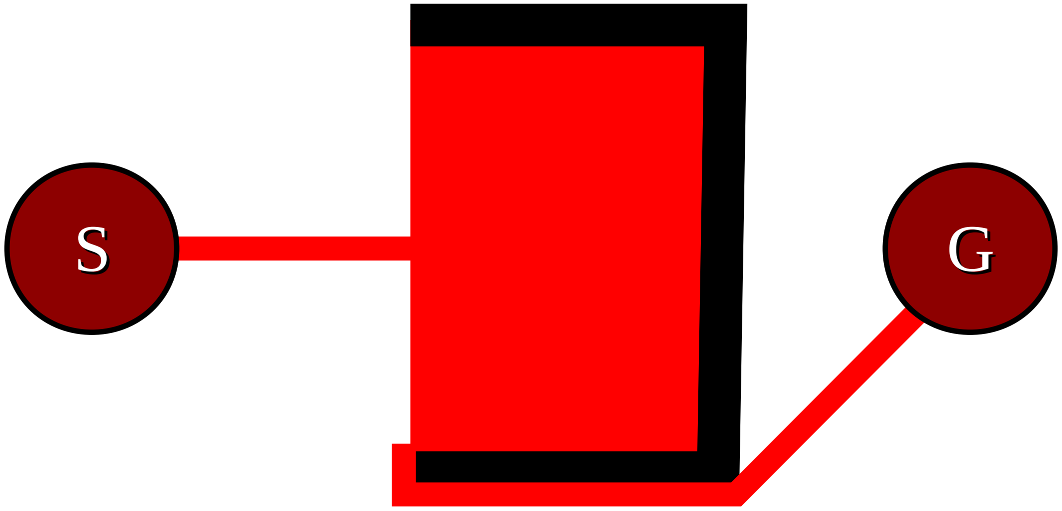


Voronoi Graph

Motion Path Planning: Best First Search (& Friends)

- Don’t explicitly solve all of Cspace before searching
- Basically, keep a priority queue of unevaluated nodes, sorted by “score” (e.g. distance to goal, or distance to goal plus distance so far)
- Each iteration, expand the current “best” node
- Choice of scoring heuristic (if you have a choice!) can make tradeoffs between search speed and optimality of solution found.

Motion Path Planning: Best First Search (& Friends)



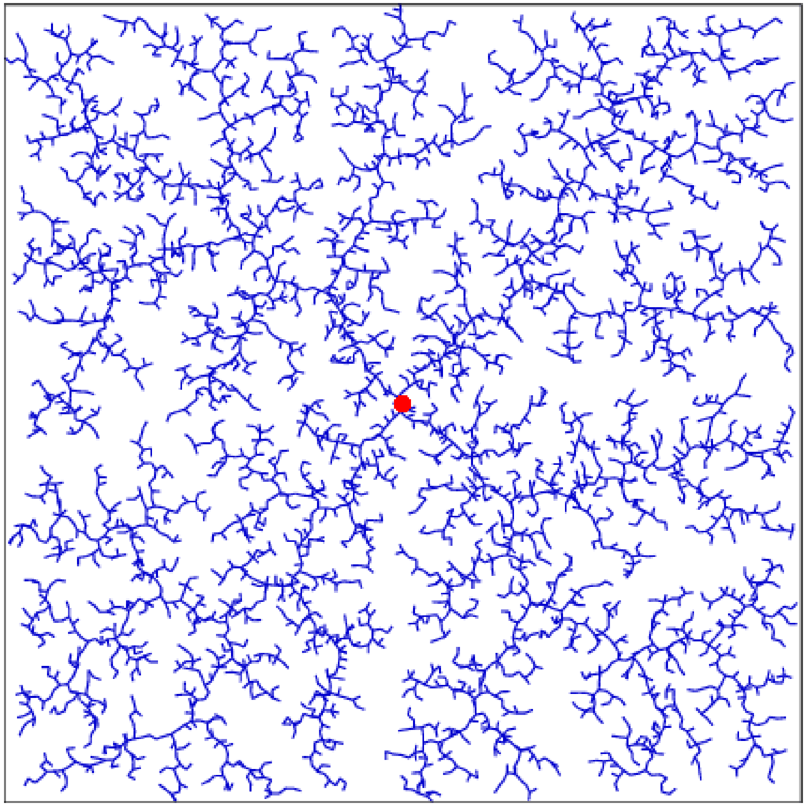
Trapped in the cul de sac for a long time.

Random search might be faster.

Rapidly-exploring Random Trees (RRTs)

- LaValle 1998
- Repeat *K* times:
  - Pick a random point *P* in <u>configuration</u> space
  - Find *N*, the closest tree node to *P*
  - Add new node *N'*, some distance Δ from *N* toward *P*
- Back to exploring entire configuration space?
- Not necessarily — instead of always picking a random target P, pick the goal some of the time

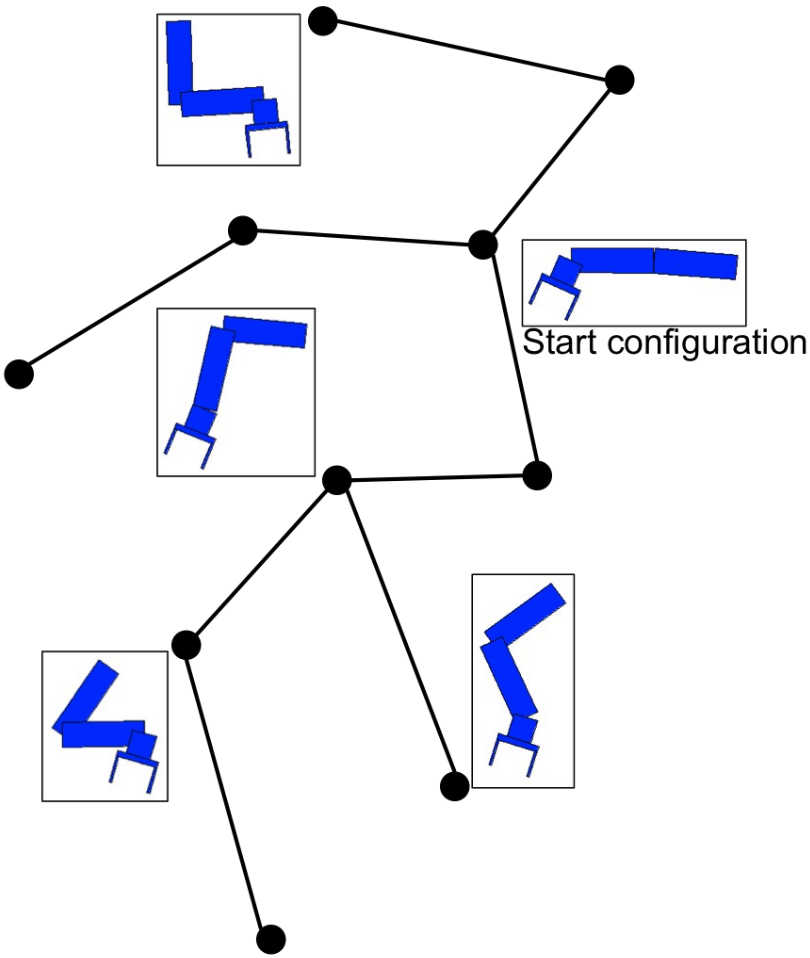
Rapidly-exploring Random Trees: Animation



http://msl.cs.uiuc.edu/rrt/treemovie.gif

RRTs for Arm Path Planning

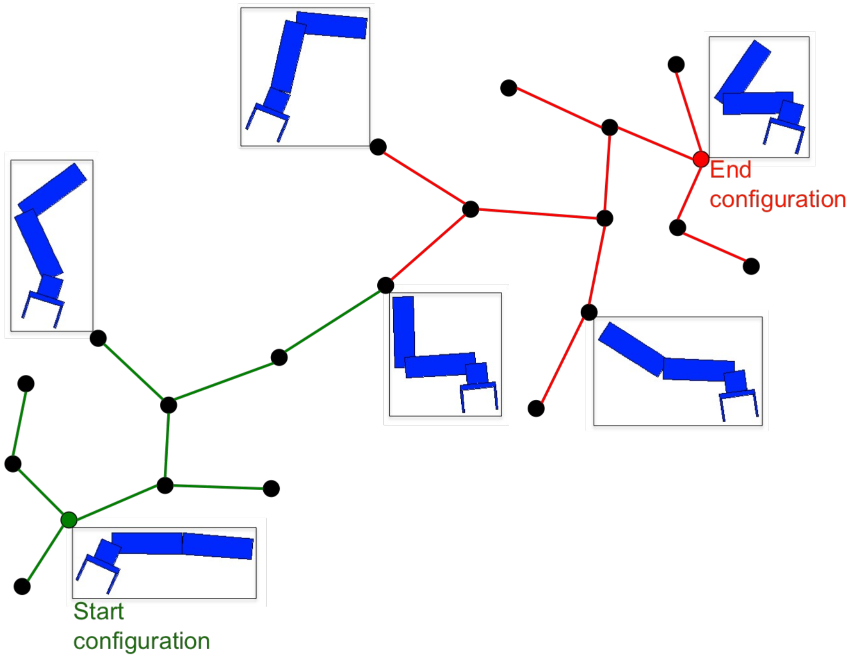
- Each node encodes an arm configuration.
- Only add nodes that don't cause collisions (with self or obstacles).
- The RRT grows by alternately extending the tree in random directions and moving toward the goal configuration.



Slide courtesy of Glenn Nickens

RRT-Connect Algorithm

- Kuffner and Lavalle, 2000
- RRT-Connect grows two RRTs, one from the start and one from the goal configuration, and biases the trees to grow toward each other.
- Once the RRTs connect, the path is extracted using backtracking.



Slide courtesy of Glenn Nickens

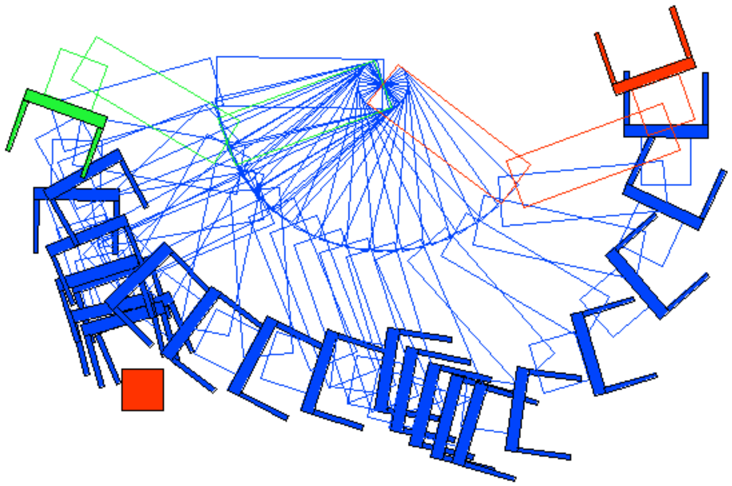
Path Smoothing

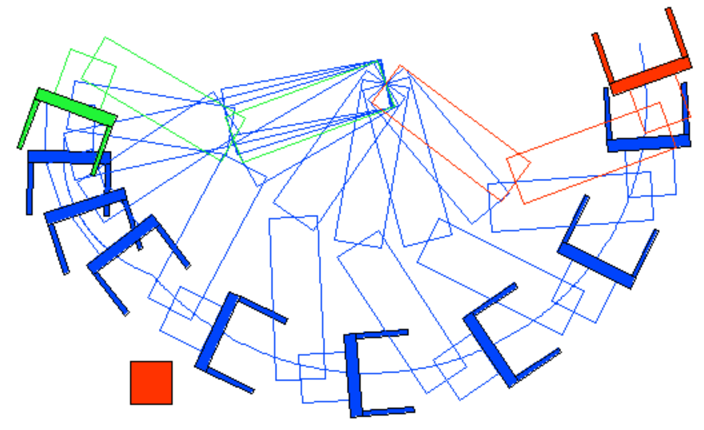
- The random component of the RRT-Connect search often results in a jerky and meandering solution.
- Therefore a smoothing algorithm is applied to the path.
- Smoothing is accomplished by selecting random segments to be snipped from the path.

Slide courtesy of Glenn Nickens

Arm Paths

- The pictures to the right show the arm's trajectory along a path from the start (green) to the end (red) configuration.
- The first image shows a path constructed by the path planner.
- The second image shows the same path, but after the smoothing algorithm has been applied to it.

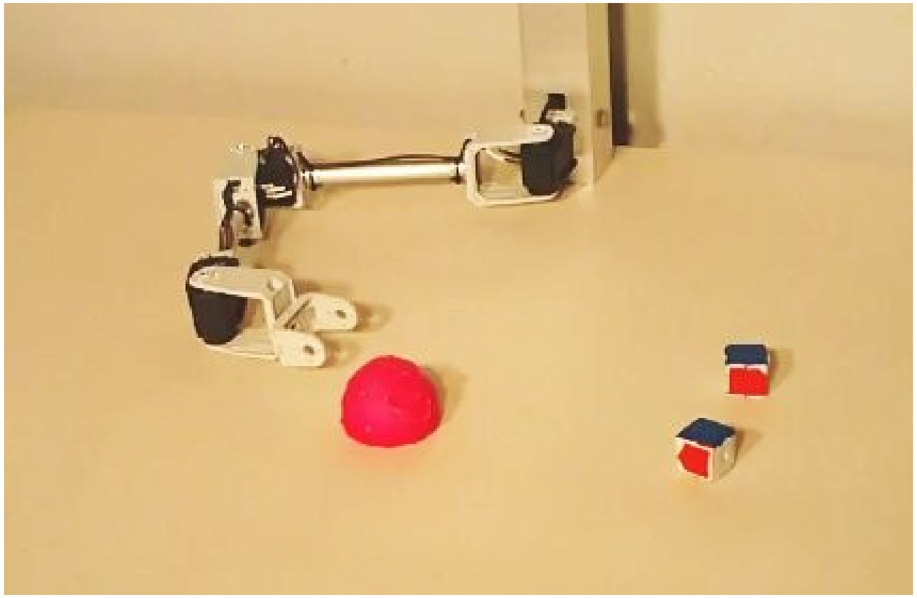




Slide courtesy of Glenn Nickens

Additional Path Planning Constraint

- With no closeable fingers, arm motion is constrained to be within about 60º of finger direction or we'll lose the object.



(video)

The Grasper

- Handles manipulation in Tekkotsu.
- Grasp planning: getting the fingers around an object.
- Path planning: moving the hand from one position to another while respecting physical constraints (joint limits, self-collisions) and avoiding obstacles.
- Many possible primitive operations (grasp, move, sweep, throw, etc.)

Motion Path Planning: Potential Fields

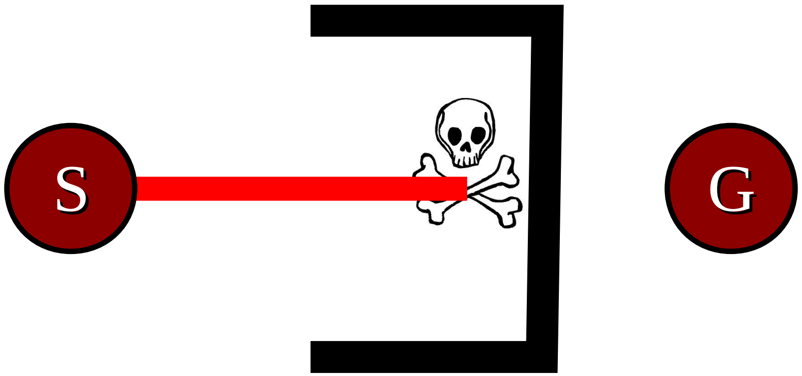
- So far we've been assuming we already know the environment, and there aren't other agents changing things around!
- Constant replanning is costly
  - replan only when something is amiss
  - replan only affected parts of existing plan (open research problem!)
- Or… don't make a plan in the first place

Motion Path Planning: Potential Fields

- Define a function *f* mapping from a specified configuration to a score value
  - e.g. distance to goal plus distance to obstacles
- Essentially just running heuristic from before:
  - Evaluate each of the currently available moves
  - Pick the one which maximizes score (or in example above, minimizes cost)

Motion Path Planning: Potential Fields

- Downside: can get stuck in local minima



- Workaround: follow edges (“bug” method)
- Upside: extremely quick and reactive
  - Popular in robosoccer for navigating to ball

Motion Path Planning: Summary

- Known Environment, Deterministic Actions
  - Road Maps (Visibility, Voronoi), A*, RRT, brushfire
- Unknown Environment, Deterministic Actions
  - Potential Field, “Bug”, D*
- Non-Deterministic and/or Unknown Environment
  - MDP, POMDP

Next Time:

Dynamics!
Friction, Forces, and Control

Thanks to:
16-741: Mechanics of Manipulation (Mason)
16-830: Planning, Execution, and Learning (Rizzi, Veloso)