CSCI 4560/6560 Computational Geometry

https://www.cs.rpi.edu/~cutler/classes/computationalgeometry/S22/

Lecture 23: Robot Motion Planning

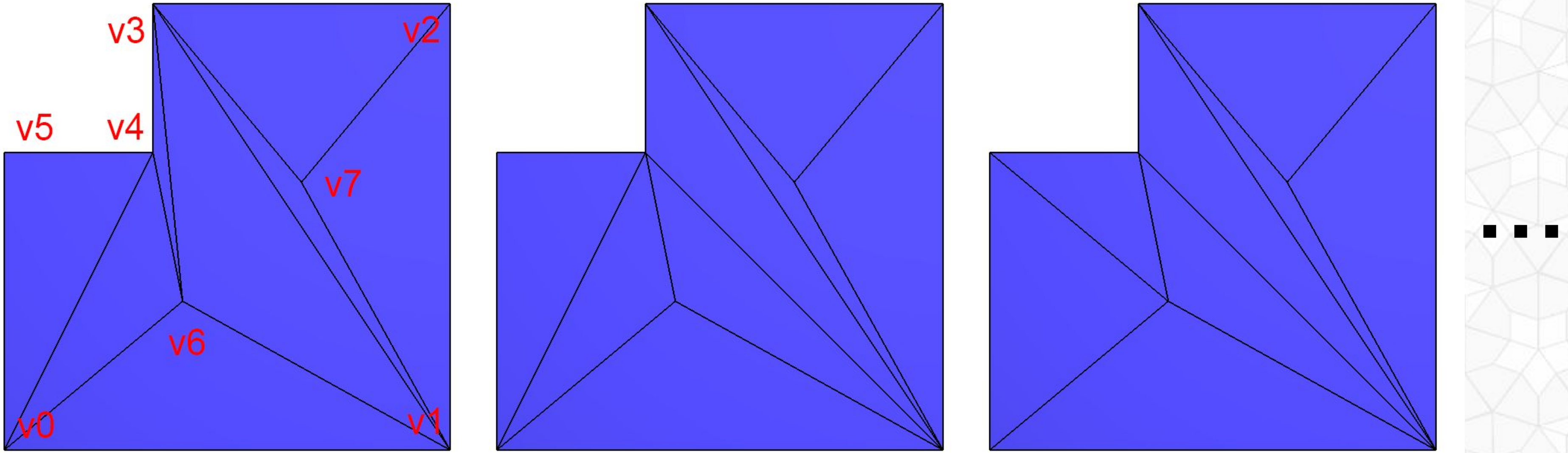
Outline for Today

- Homework 6 Due Soon & Homework 7 Part 1 Posted
- Last Time: General Position, Robustness, & Exact Computation
- Motivation: Robot Motion Planning
- Previous Lecture: Voronoi Diagram of Segments for Motion Planning
- Degrees of Freedom & Configuration Space
- Trapezoid Map for Motion Planning
- Non-Point, Non-Rotating Robots & Minkowski Sums
- Related Operations: Convolution, Morphology, Accessible Surfaces
- Rotations & Higher Dimensional Configuration Space
- Next Time: ?

Homework 6

- Ok if your solution is *not the shortest path*
  (e.g., it has unnecessary edits that are later reverted)

```
v3-v6  ->  v1-v4
v0-v4  ->  v5-v6
v1-v3  ->  v4-v7
v1-v4  ->  v6-v7
v6-v7  ->  v1-v4
v5-v6  ->  v0-v4
v3-v7  ->  v2-v4
```



Outline for Today

- Homework 6 Due Soon & Homework 7 Part 1 Posted
- Last Time: General Position, Robustness, & Exact Computation
- Motivation: Robot Motion Planning
- Previous Lecture: Voronoi Diagram of Segments for Motion Planning
- Degrees of Freedom & Configuration Space
- Trapezoid Map for Motion Planning
- Non-Point, Non-Rotating Robots & Minkowski Sums
- Related Operations: Convolution, Morphology, Accessible Surfaces
- Rotations & Higher Dimensional Configuration Space
- Next Time: ?

Factorization by Gaussian Elimination

- Divide by zero is not the only concern…
- We should also avoid division by very small values, e.g., epsilon:

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{A} = \begin{bmatrix} -\epsilon & 1 \\ 1 & -1 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 1-\epsilon \\ 0 \end{bmatrix},$$

Correct answer: x_1=1
But we will have robustness problems if ε is very small!

$$\begin{bmatrix} -\epsilon & 1 & 1-\epsilon \\ 0 & -1+\epsilon^{-1} & \epsilon^{-1}-1 \end{bmatrix} \Rightarrow \begin{aligned} x_2 &= 1 \\ x_1 &= \frac{(1-\epsilon)-1}{-\epsilon} \end{aligned}$$

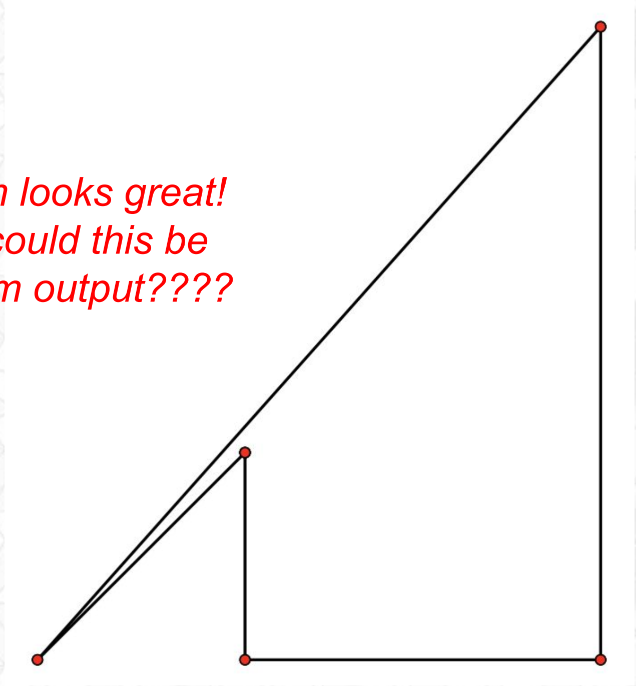
It's better to pivot / swap rows for the row with the largest value in this column

Fundamentals of Numerical Computation, Driscoll & Braun, 2017
https://fncbook.github.io/v1.0/linsys/pivoting.html

Incremental Convex Hull Construction

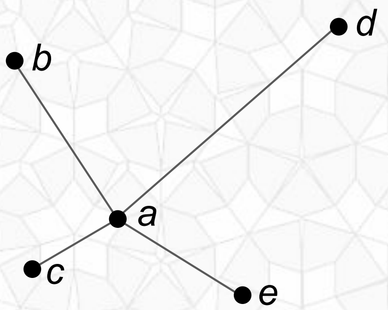
- Make a triangle with the first 3 points
- For each additional point *r*
  - Find an outside edge that is "visible" from *r*
  - Expand to a sequence of connected edges $v_i \to v_{i+1} \to v_{i+2} \; (\to \ldots \;) \to v_j$
  - Remove middle points (e.g., v_{i+1} & v_{i+2}) from hull, add point *r* to hull

Algorithm looks great! So how could this be a program output????



"Geometric Computing: The Science of Making Geometric Algorithms Work", Kurt Mehlhorn
https://people.mpi-inf.mpg.de/~mehlhorn/ftp/SoCG09.pdf

Avoid Creating Irrational Numbers

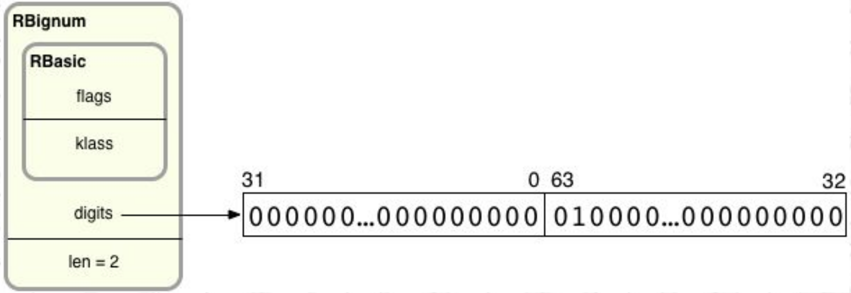


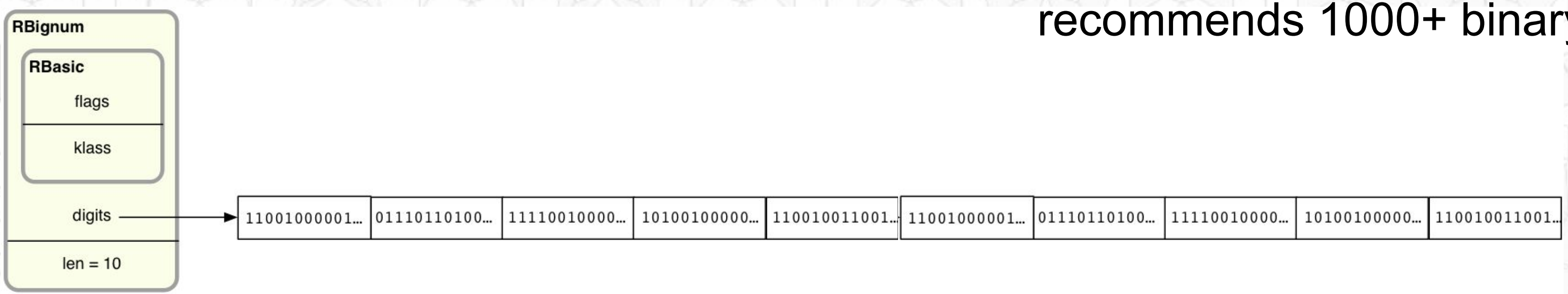
- Problem: Given 5 points with integer coordinates, find the nearest neighbor to point *a*
- Compute the length of lines *ab, ac, ad, ae*
  - length(*ab*) = sqrt ($(x_a-x_b)*(x_a-x_b) + (y_a-y_b)*(y_a-y_b)$)
  - Note: the sqrt, will likely create irrational numbers!
- Sort the lengths, return endpoint for shortest line length

- Instead… compute & sort the squares of the line lengths
  - squared_length(*ab*) = $(x_a-x_b)*(x_a-x_b) + (y_a-y_b)*(y_a-y_b)$
  - This is an integer!
- This will always return the correct answer to the original question. *WITHOUT creating irrational numbers!*

Arbitrary Precision Arithmetic

- If we do not have irrational numbers in our program…
- We can store integers using a “BigNum” infinite precision integer type





- 64 bit binary integer = ~19 bit base 10 integer

- RSA Security requires at least 100 binary digits, but recommends 1000+ binary digits

https://patshaughnessy.net/2014/1/9/how-big-is-a-bignum

Arbitrary Precision Arithmetic

- If we do not have irrational numbers in our program…
- We can store rational numbers as a ratio of two BigNums
- Reduce fractions whenever possible to minimize storage:

$$\frac{49578291287491495151508905425869578}{74367436931237242727263358138804367} \rightarrow \frac{2}{3}$$

https://algorist.com/problems/Arbitrary-Precision_Arithmetic.html

Outline for Today

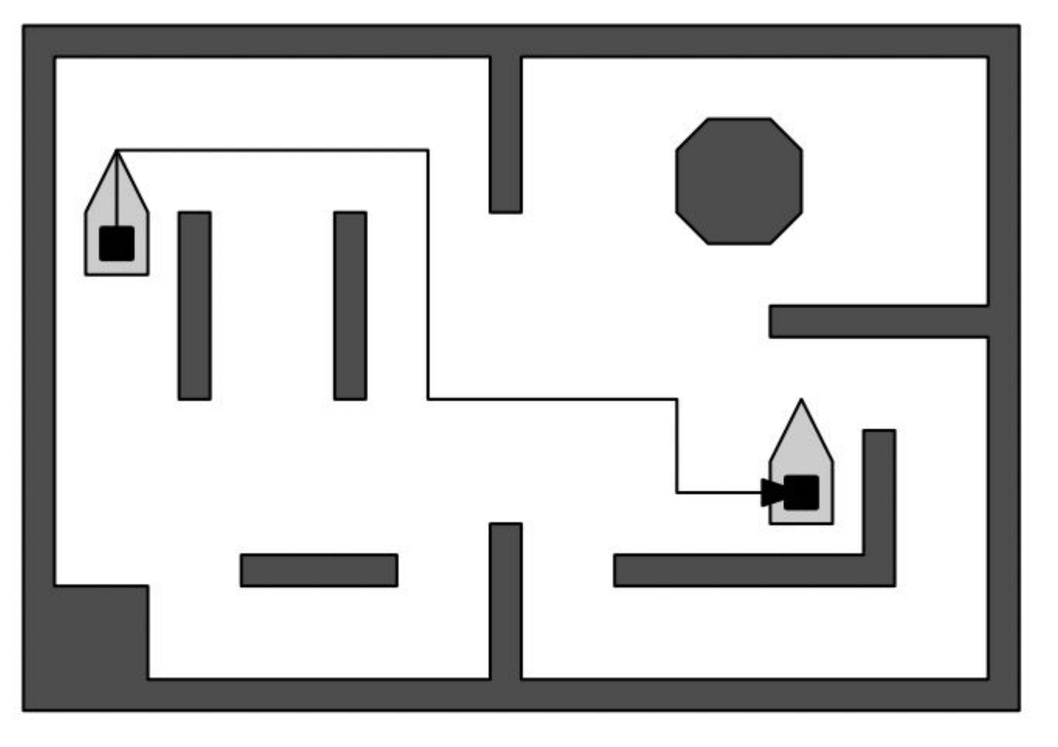
- Homework 6 Due Soon & Homework 7 Part 1 Posted
- Last Time: General Position, Robustness, & Exact Computation
- Motivation: Robot Motion Planning
- Previous Lecture: Voronoi Diagram of Segments for Motion Planning
- Degrees of Freedom & Configuration Space
- Trapezoid Map for Motion Planning
- Non-Point, Non-Rotating Robots & Minkowski Sums
- Related Operations: Convolution, Morphology, Accessible Surfaces
- Rotations & Higher Dimensional Configuration Space
- Next Time: ?

Motivation: Robot Motion Planning

- 2D (or 3D)
- Navigate from starting location to end location

- Avoid all obstacles
- Touching/sliding along the obstacles may be allowed (or disallowed)

- Rotation may be allowed (or disallowed)



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 13

Outline for Today

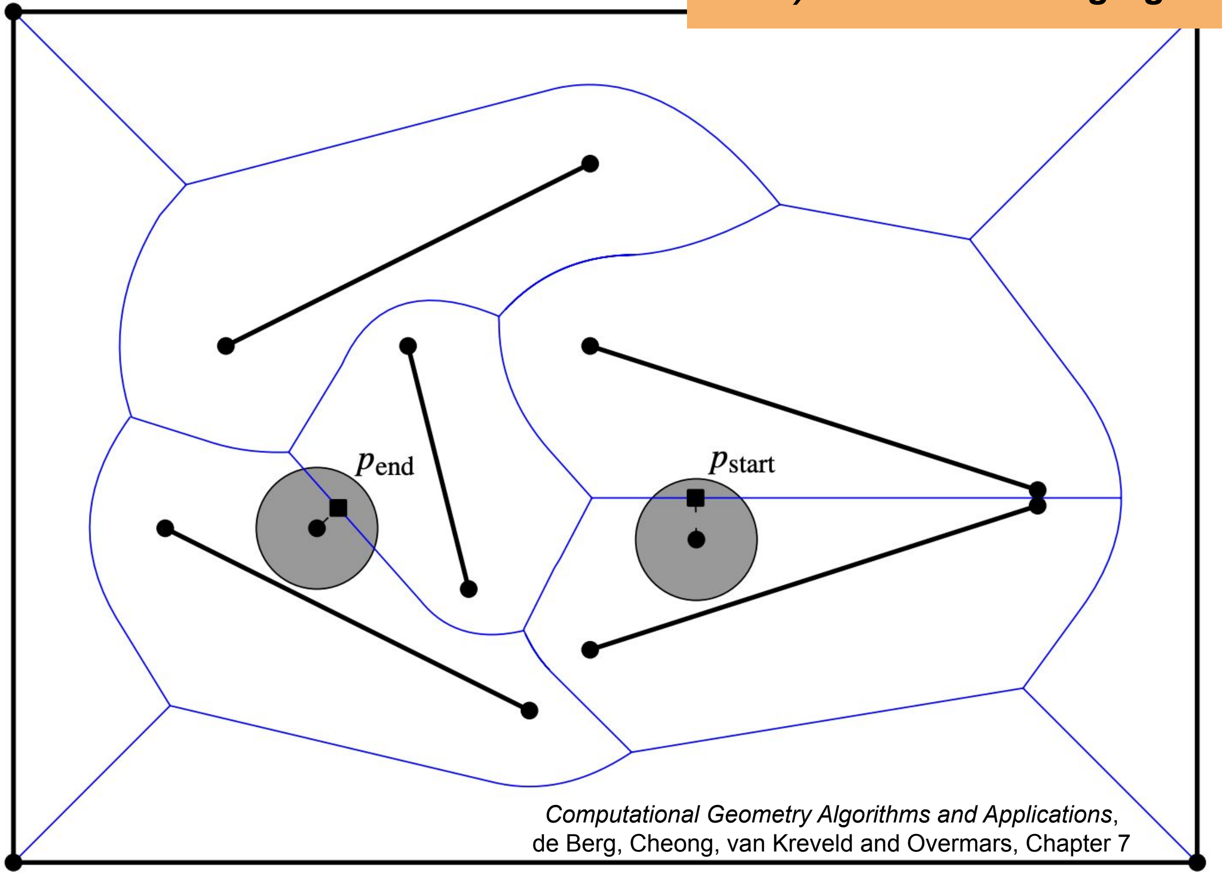
- Homework 6 Due Soon & Homework 7 Part 1 Posted
- Last Time: General Position, Robustness, & Exact Computation
- Motivation: Robot Motion Planning
- Previous Lecture: Voronoi Diagram of Segments for Motion Planning
- Degrees of Freedom & Configuration Space
- Trapezoid Map for Motion Planning
- Non-Point, Non-Rotating Robots & Minkowski Sums
- Related Operations: Convolution, Morphology, Accessible Surfaces
- Rotations & Higher Dimensional Configuration Space
- Next Time: ?

Voronoi Diagram of Line Segments

Proper implementation (robustness, floating point, etc) is extra challenging

- Voronoi Diagram w/ segments has parabolic curved segments
- But is still *O(n)* in complexity - (# of segments)
- And can be computed in *O(n log n)*

- *But why is this useful?*

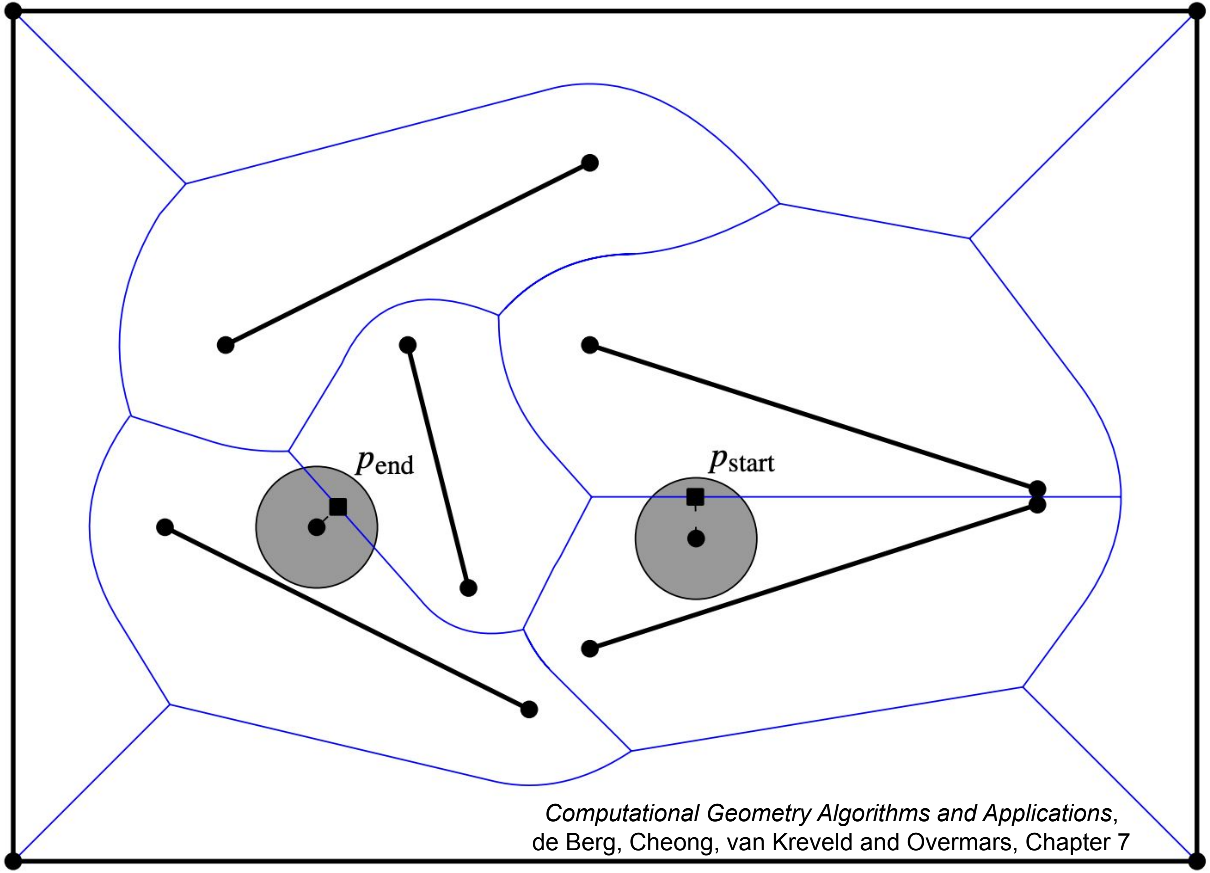


Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 7

Application: Robotics & Motion Planning

- Let's move a circular/disk robot from the start position to the end position.

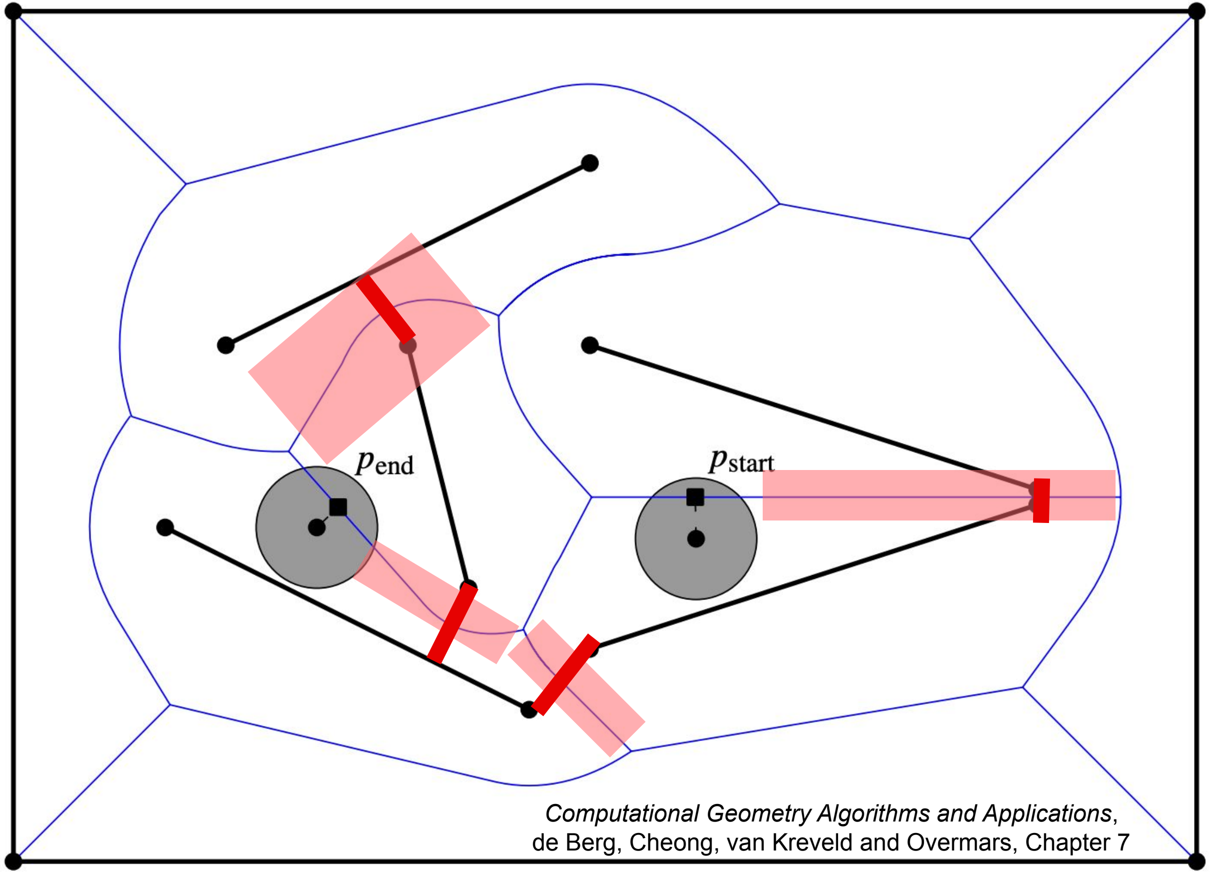
- Step 1: Project the robot center to the closest Voronoi edge (line segment or parabolic curve)



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 7

Application: Robotics & Motion Planning

- Step 1: Project the robot center to the closest Voronoi edge (line segment or parabolic curve)
- Step 2: Remove edges from the diagram graph with smallest distance to segment < radius.

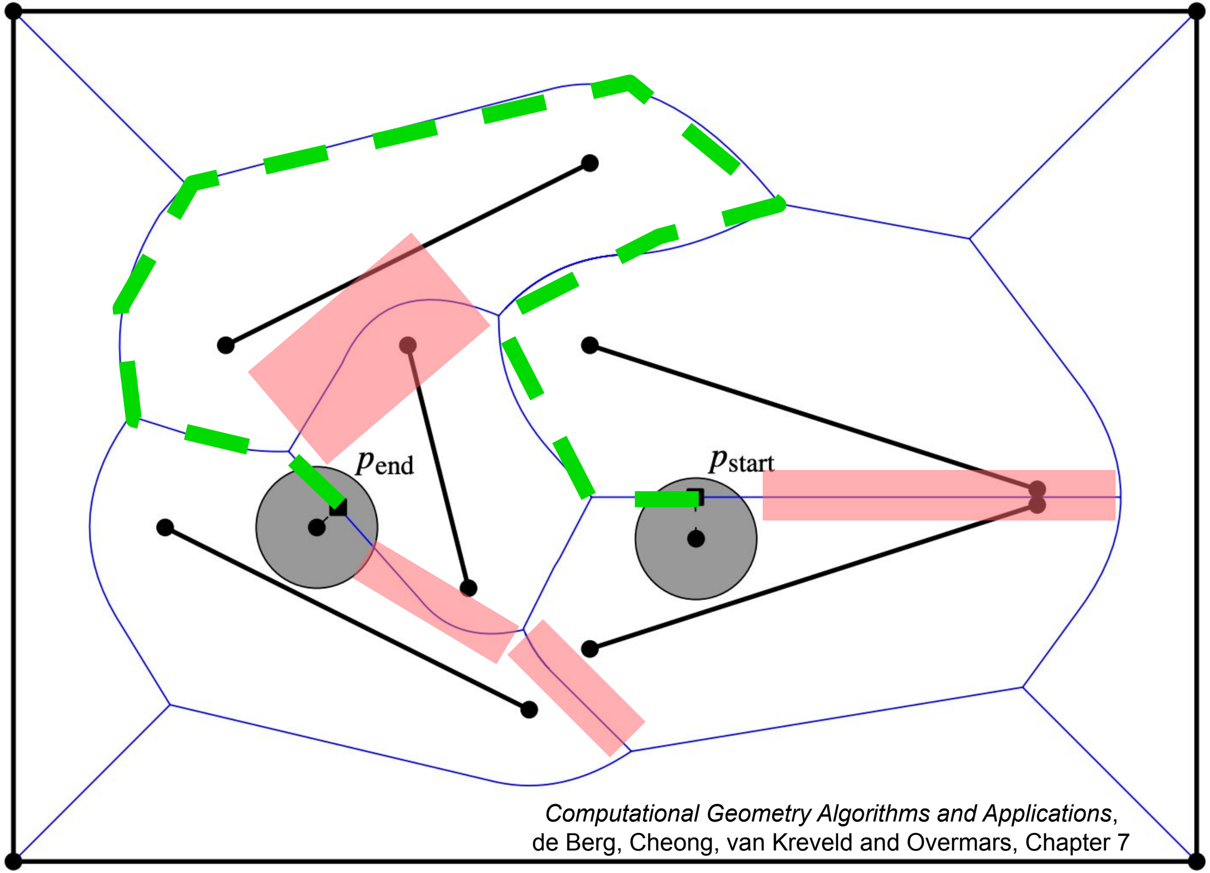


Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 7

Application: Robotics & Motion Planning

- Step 2: Remove edges from the diagram graph with smallest distance to segment < radius.

- Step 3: Search the remaining graph for a connected path from start to end.



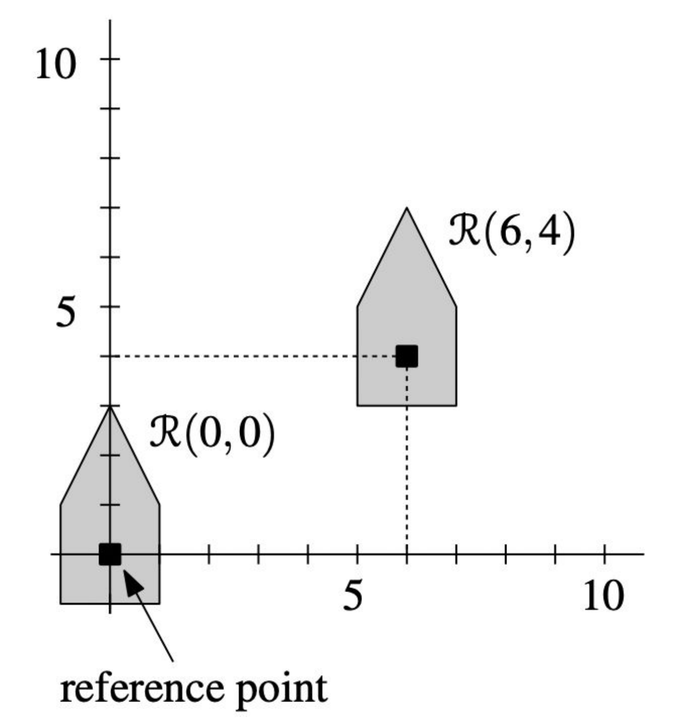
Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 7

Outline for Today

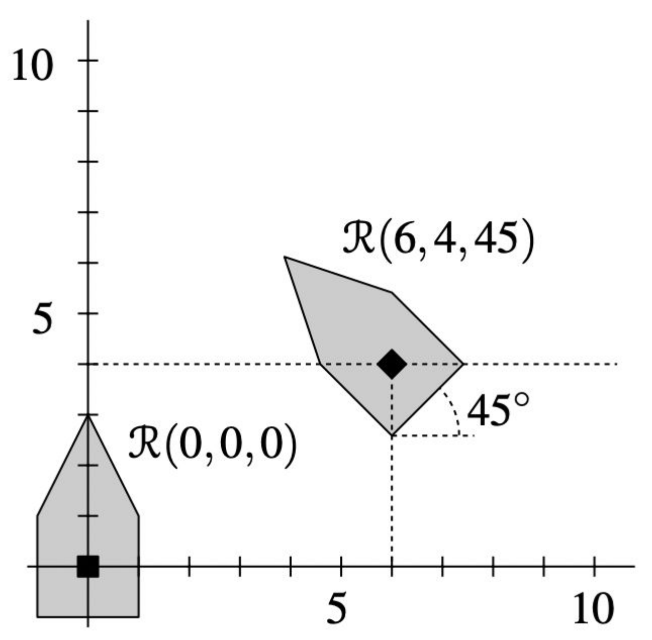
- Homework 6 Due Soon & Homework 7 Part 1 Posted
- Last Time: General Position, Robustness, & Exact Computation
- Motivation: Robot Motion Planning
- Previous Lecture: Voronoi Diagram of Segments for Motion Planning
- Degrees of Freedom & Configuration Space
- Trapezoid Map for Motion Planning
- Non-Point, Non-Rotating Robots & Minkowski Sums
- Related Operations: Convolution, Morphology, Accessible Surfaces
- Rotations & Higher Dimensional Configuration Space
- Next Time: ?

Robot Degree of Freedom (DOF)

2D w/ Translation only → 2 DOF



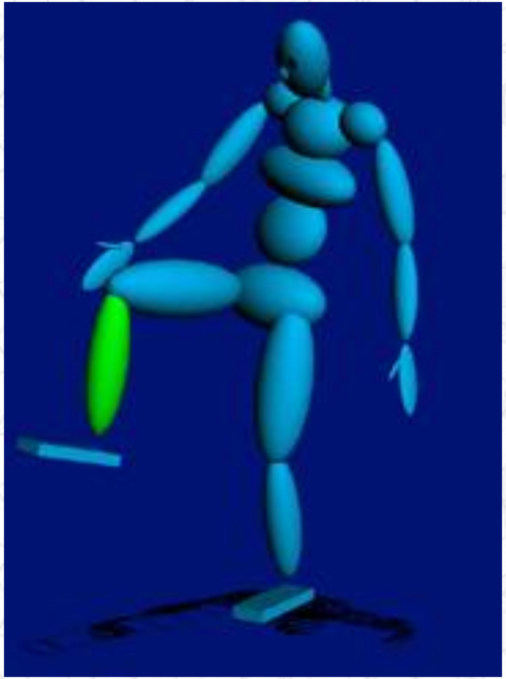
2D w/ Translation & Rotation → 3 DOF



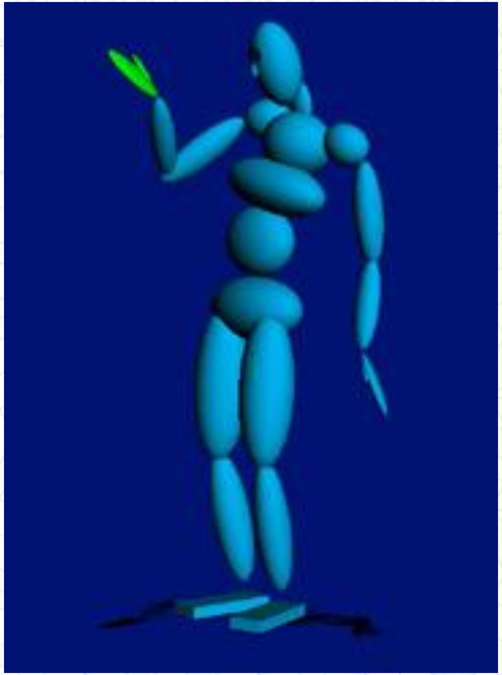
Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 13

Degree of Freedom (DOF)

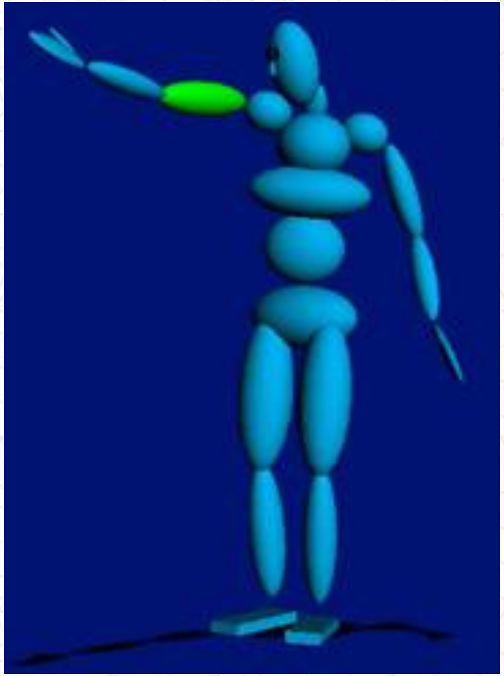
- 3D w/ Translation & up to 3 Rotational DOF → up to 6 total DOF



1 Rotational DOF: knee



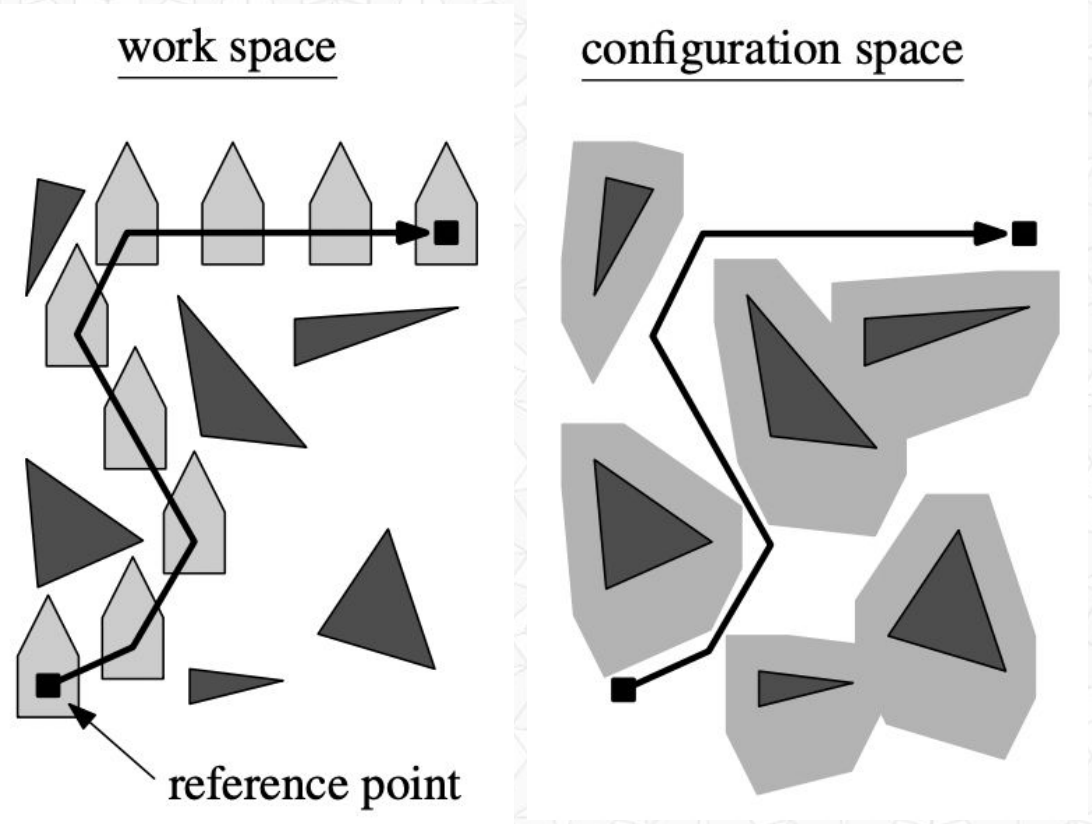
2 Rotational DOF: wrist



3 Rotational DOF: arm

Configuration Space

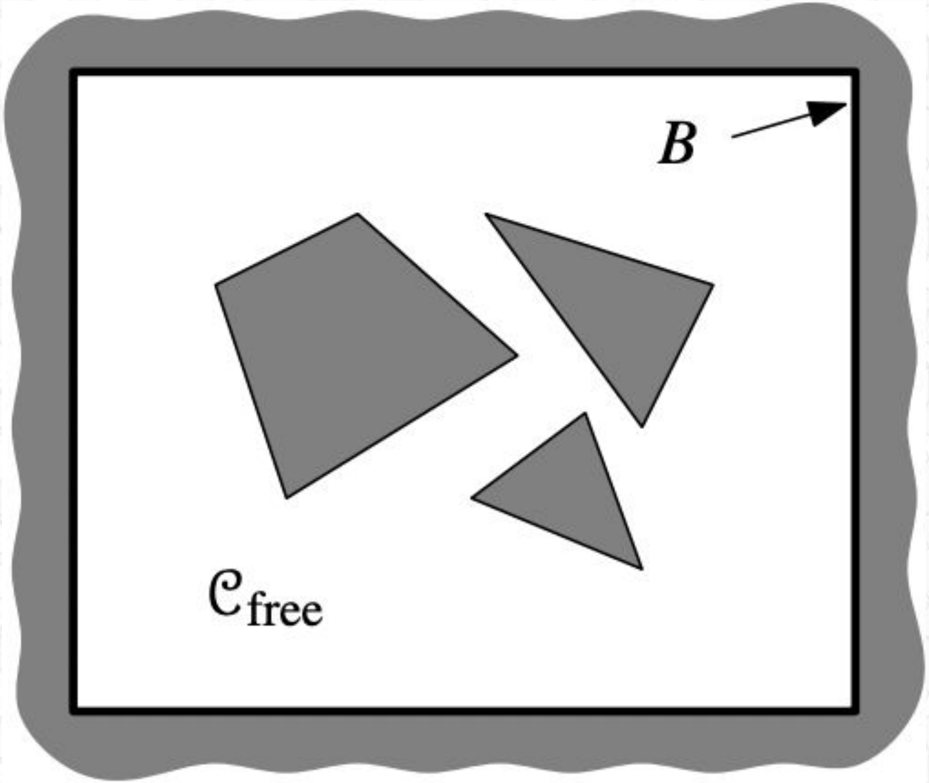
- The dimensions of configuration space match the DOF of the robot
- Usually configuration space is higher dimensional than the environment/workspace
- It is often useful to construct, visualize, and even solve the problem in “configuration space”



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 13

Determine the Boundaries of the Free Space

- Initially assume a point robot (rotation is thus irrelevant)
- How do we efficiently represent & plan within this free space?



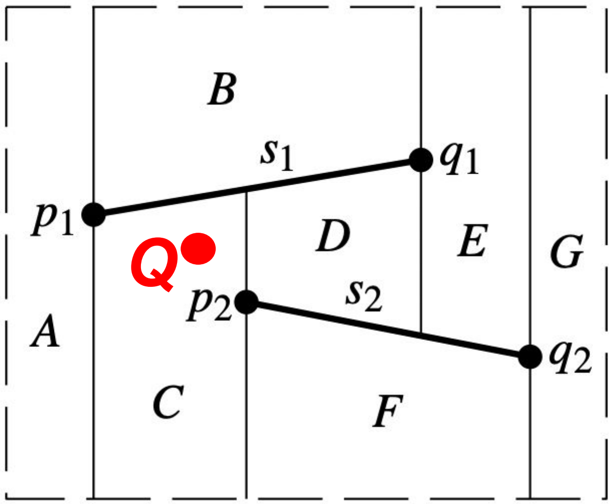
Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 13

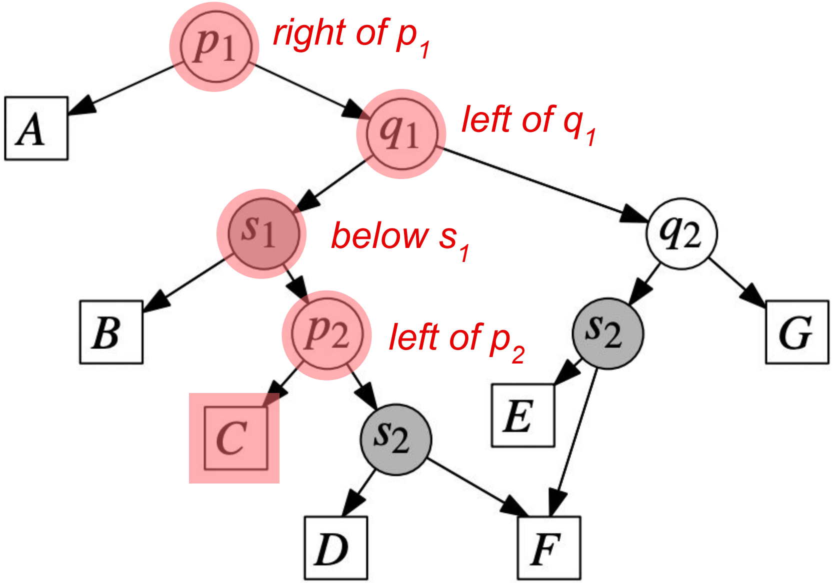
Outline for Today

- Homework 6 Due Soon & Homework 7 Part 1 Posted
- Last Time: General Position, Robustness, & Exact Computation
- Motivation: Robot Motion Planning
- Previous Lecture: Voronoi Diagram of Segments for Motion Planning
- Degrees of Freedom & Configuration Space
- Trapezoid Map for Motion Planning
- Non-Point, Non-Rotating Robots & Minkowski Sums
- Related Operations: Convolution, Morphology, Accessible Surfaces
- Rotations & Higher Dimensional Configuration Space
- Next Time: ?

Trapezoidal Map & Directed Acyclic Graph

- n = # of segments
- size (# of nodes) = O(n)
- height = O(log n) expected (using Randomized Incremental Construction)

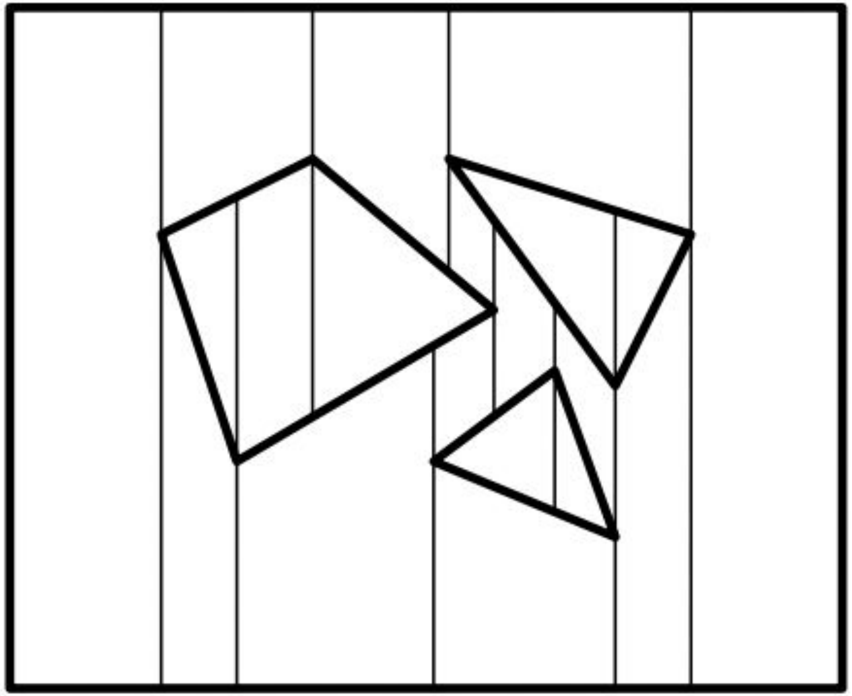




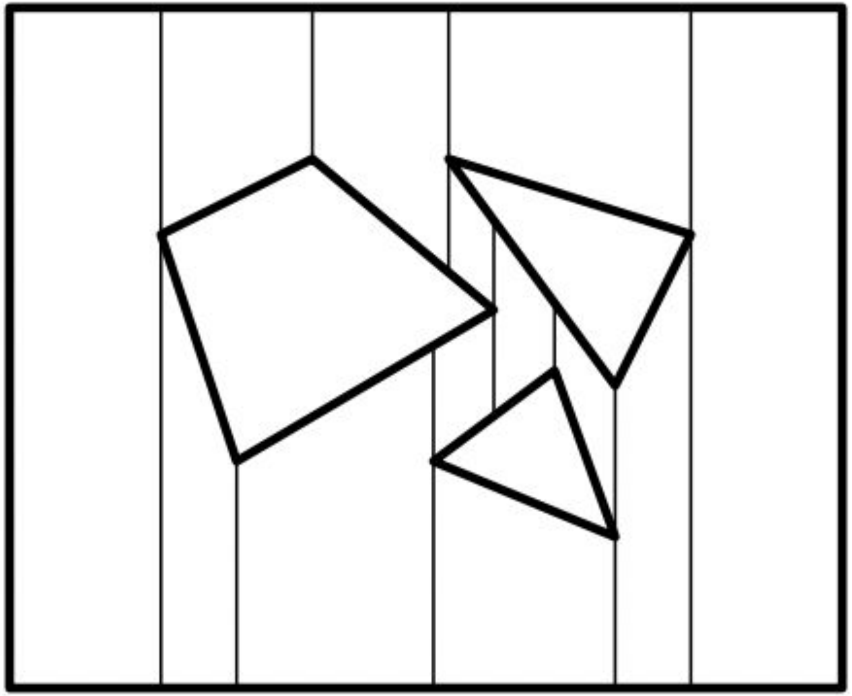
Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 6

Build Trapezoidal Map of Free Space

Insert all obstacle boundaries



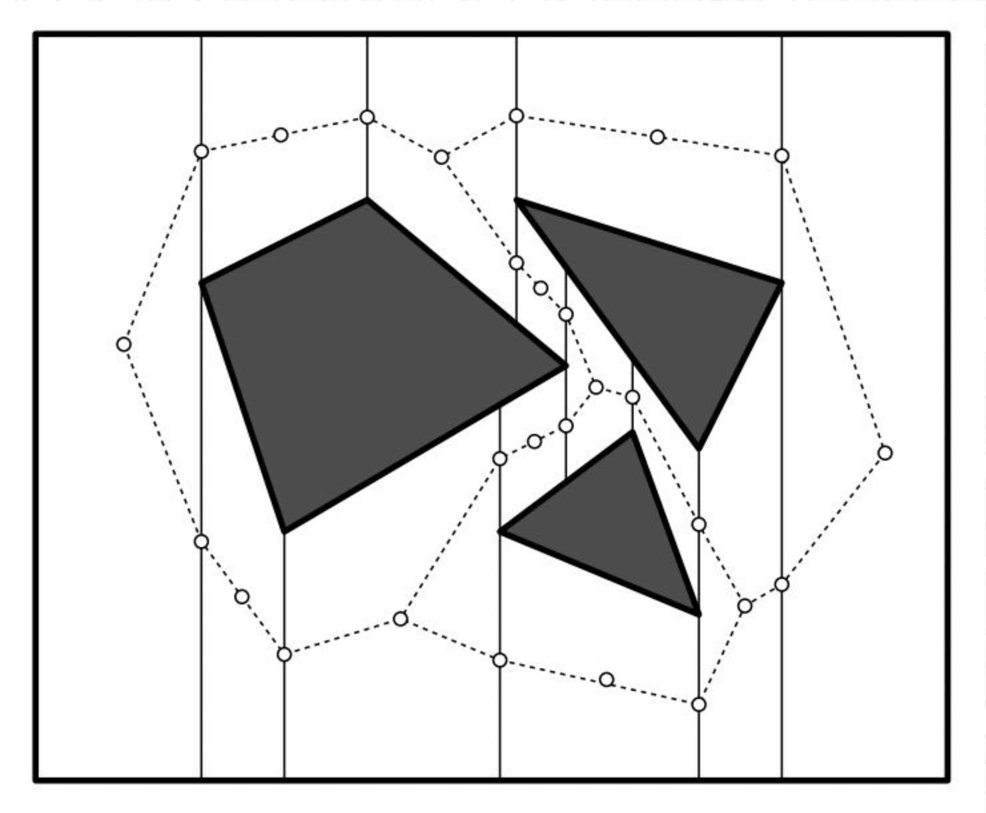
Remove trapezoids inside obstacles



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 13

Motion Planning Graph

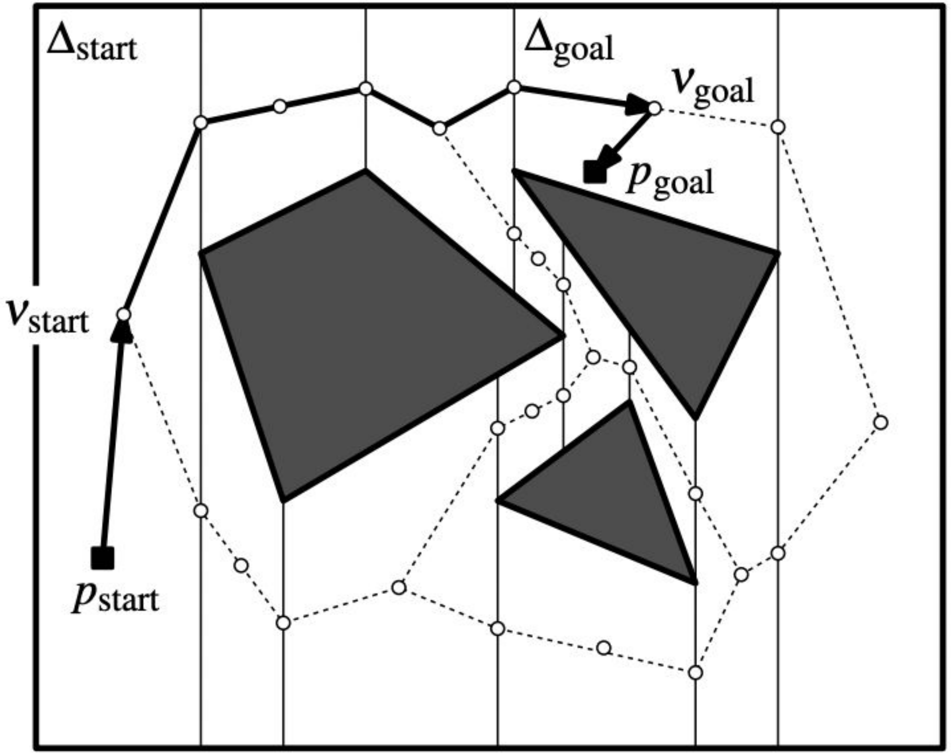
- Add graph node within each trapezoid
- Add graph node at midpoint of each vertical edge
- Connect two graph nodes if they share a vertical edge



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 13

Motion Planning Graph

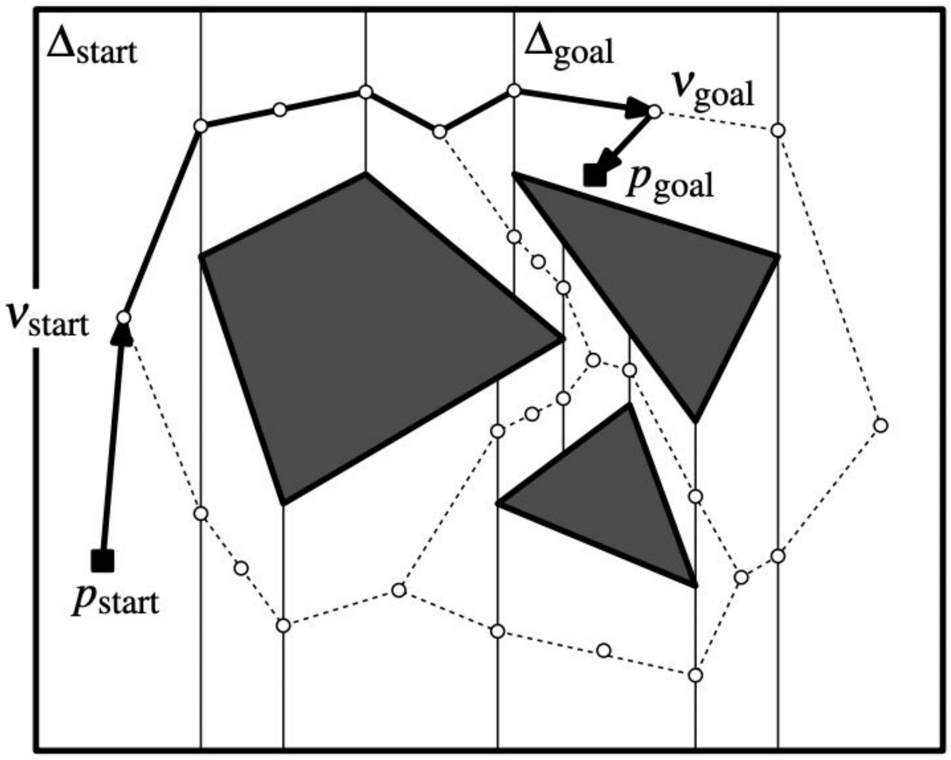
- Locate which trapezoid contains the start & end points
- Follow a straight line path from the start point to the graph node within the containing trapezoid
- Perform breadth first search on the graph to find a path from start to end (if any exists)



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 13

Motion Planning Graph - Analysis

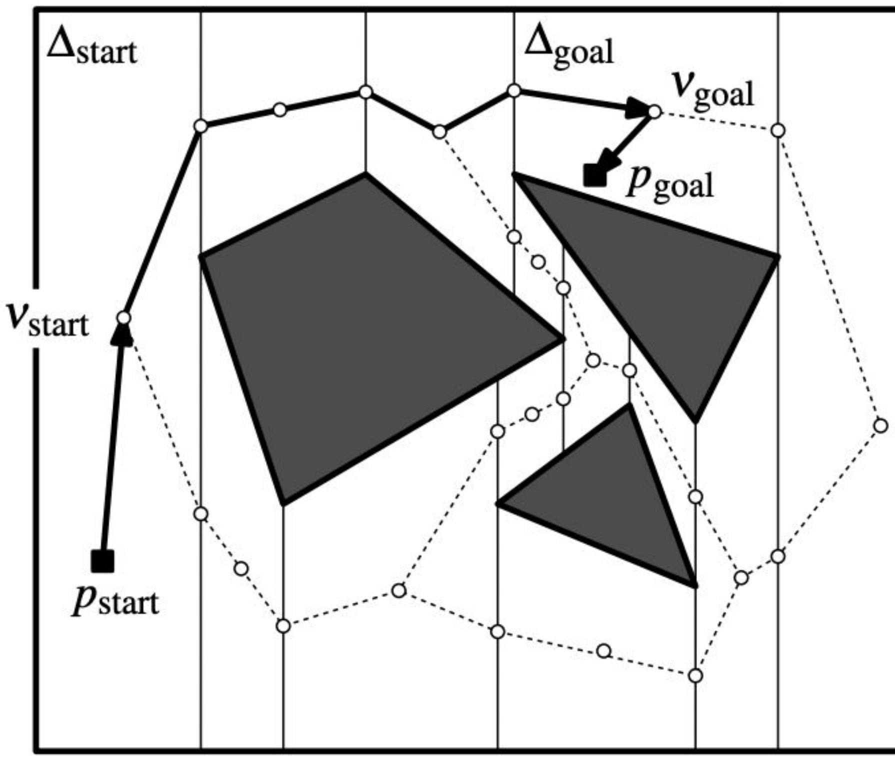
- Size of Trapezoid Map
- Build Trapezoid Map
- Locate start/end trapezoid
- Breadth first search



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 13

Motion Planning Graph - Analysis

- Size of Trapezoid Map
  → O(n)
- Build Trapezoid Map
  → O(n log n)
- Locate start/end trapezoid
  → O(log n)
- Breadth first search
  → O(n)



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 13

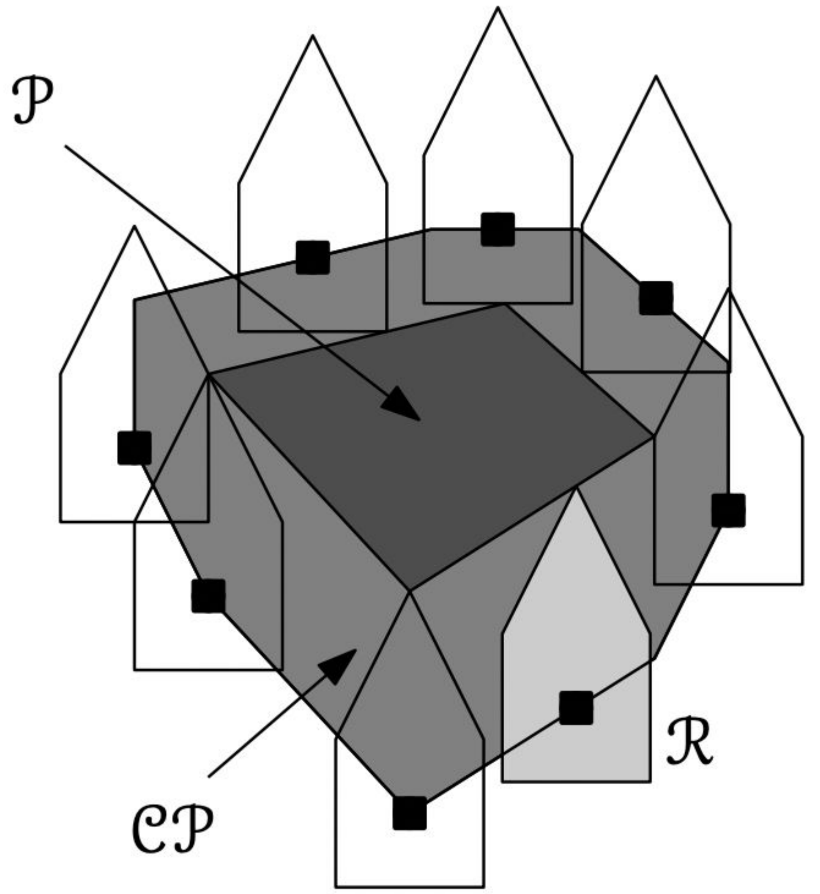
Outline for Today

- Homework 6 Due Soon & Homework 7 Part 1 Posted
- Last Time: General Position, Robustness, & Exact Computation
- Motivation: Robot Motion Planning
- Previous Lecture: Voronoi Diagram of Segments for Motion Planning
- Degrees of Freedom & Configuration Space
- Trapezoid Map for Motion Planning
- Non-Point, Non-Rotating Robots & Minkowski Sums
- Related Operations: Convolution, Morphology, Accessible Surfaces
- Rotations & Higher Dimensional Configuration Space
- Next Time: ?

Non-Point Robots

- Initially, let's ignore rotation
- How close can the robot get to the obstacle?

- The obstacle boundaries in configuration space will be expanded
- The origin / reference point of the robot is important

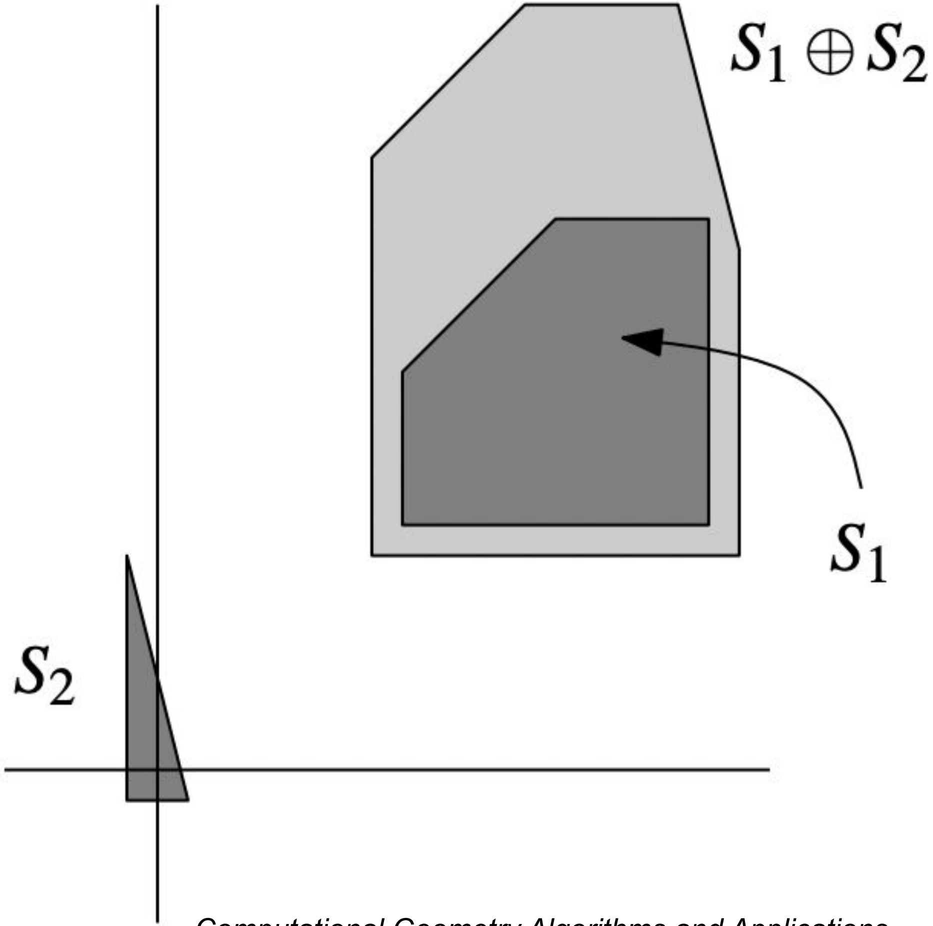


Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 13

Minkowski Sum

Related to:

- Convolution
- Morphology
    - Dilation
    - Erosion
    - Opening
    - Closing
- Accessible Surfaces

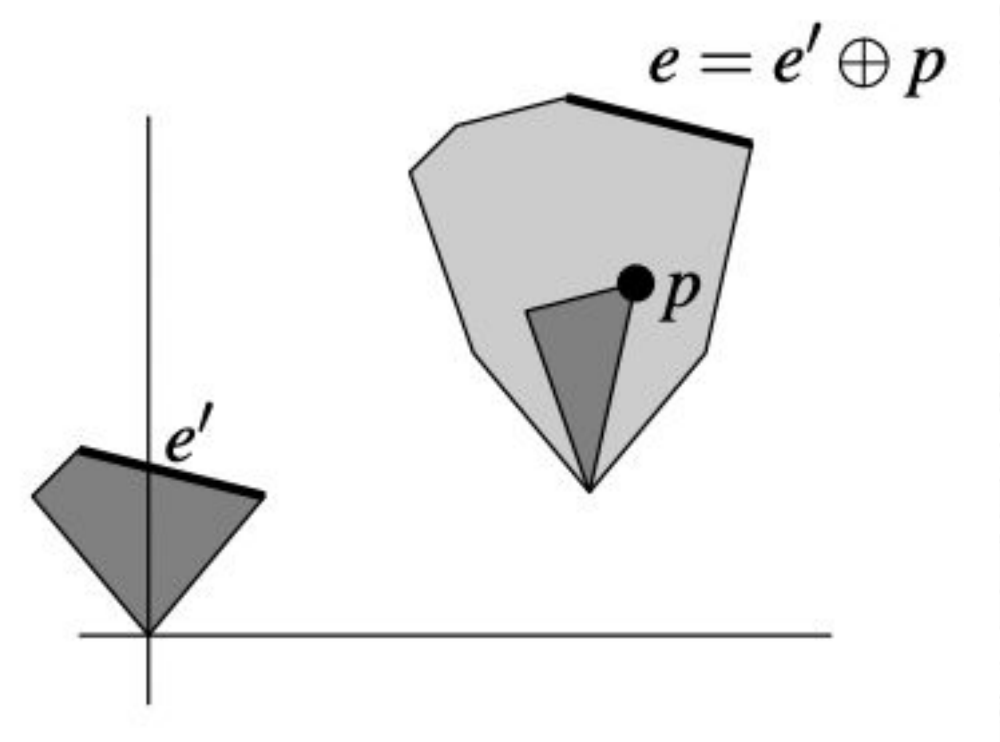


Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 13

Complexity of Minkowski Sum

- Given:
  - Robot with $n = 4$ edges
  - Obstacle with $m = 3$ edges
- How many edges does the resulting shape hape?
  - $n+m = 7$ edges

- Each edge in the Minkowski sum is defined by an edge on one shape and a point on the other shape
- If two or more edges of the robot and obstacle are parallel, it will have fewer than n+m edges



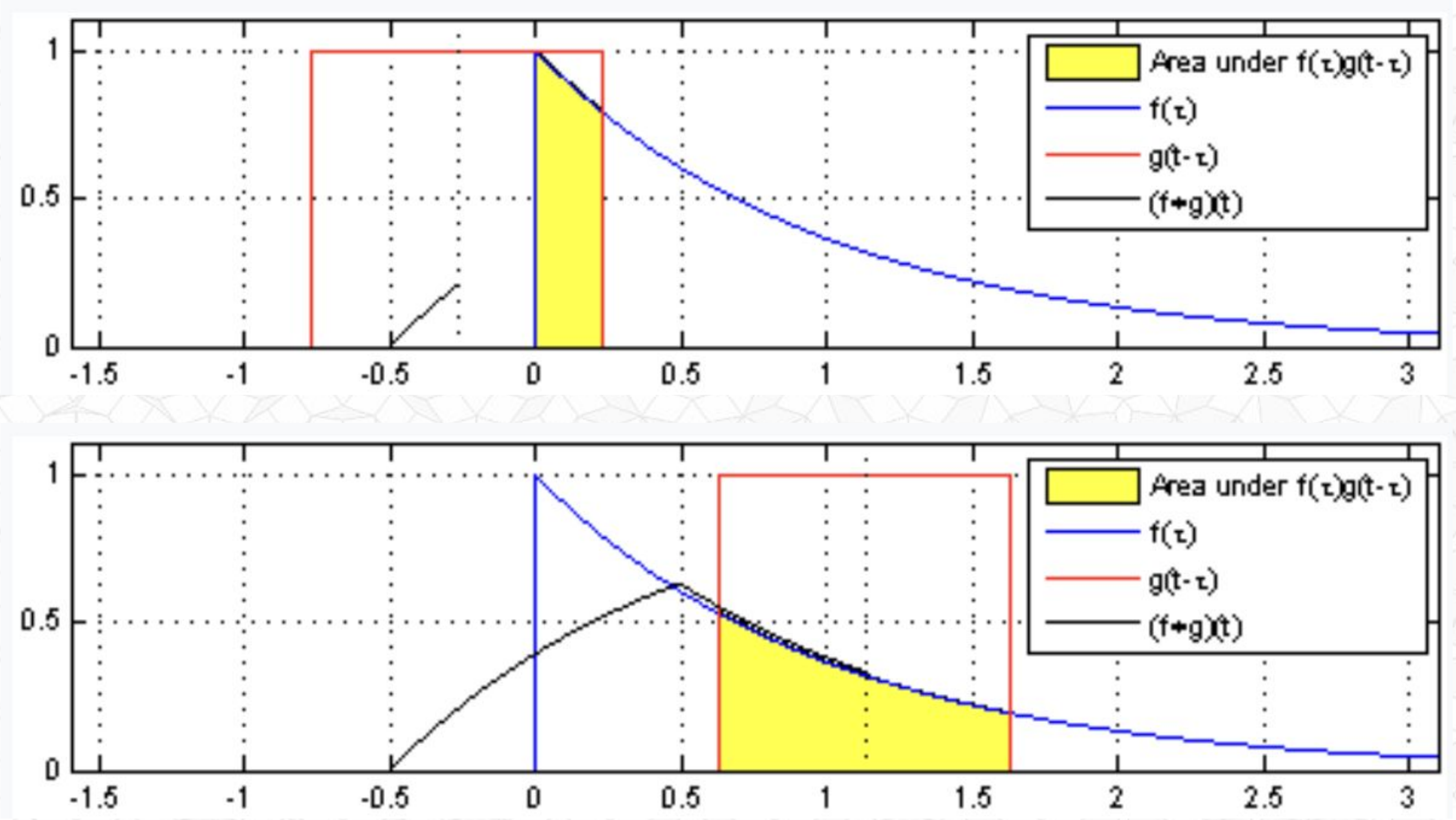
Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 13

Outline for Today

- Homework 6 Due Soon & Homework 7 Part 1 Posted
- Last Time: General Position, Robustness, & Exact Computation
- Motivation: Robot Motion Planning
- Previous Lecture: Voronoi Diagram of Segments for Motion Planning
- Degrees of Freedom & Configuration Space
- Trapezoid Map for Motion Planning
- Non-Point, Non-Rotating Robots & Minkowski Sums
- Related Operations: Convolution, Morphology, Accessible Surfaces
- Rotations & Higher Dimensional Configuration Space
- Next Time: ?

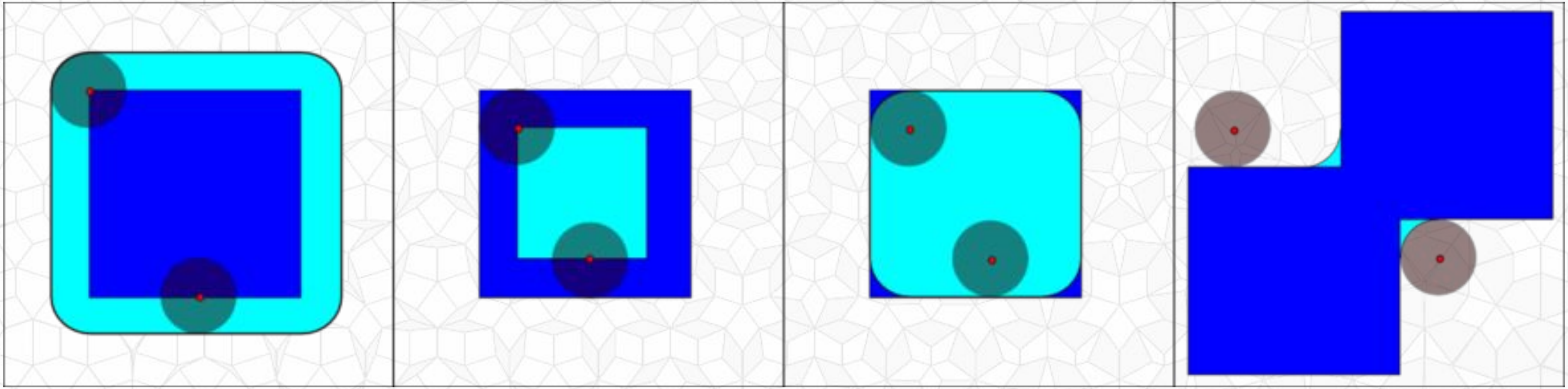
https://en.wikipedia.org/wiki/Convolution

Convolution - “Flip & Slide” from Signals & Systems



Morphology for Computer Vision

- For Noise Removal and Other Image Processing Tasks



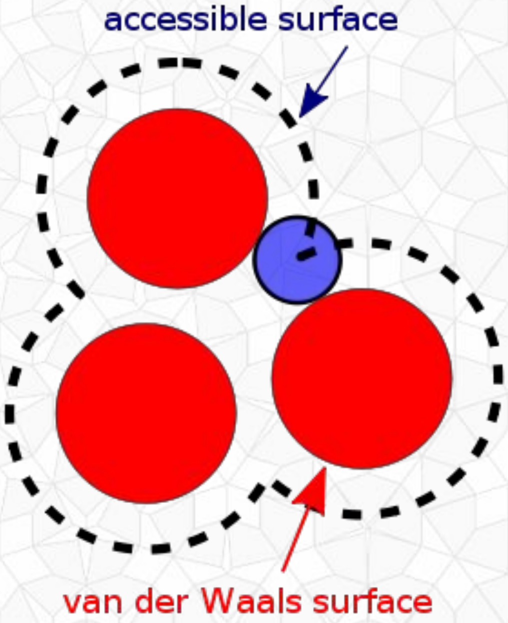
Dilation | Erosion | Opening | Closing

https://en.wikipedia.org/wiki/Mathematical_morphology

Weathering, Accessible Surfaces

https://en.wikipedia.org/wiki/Accessible_surface_area

- Simulate water flow & removal of surface dirt



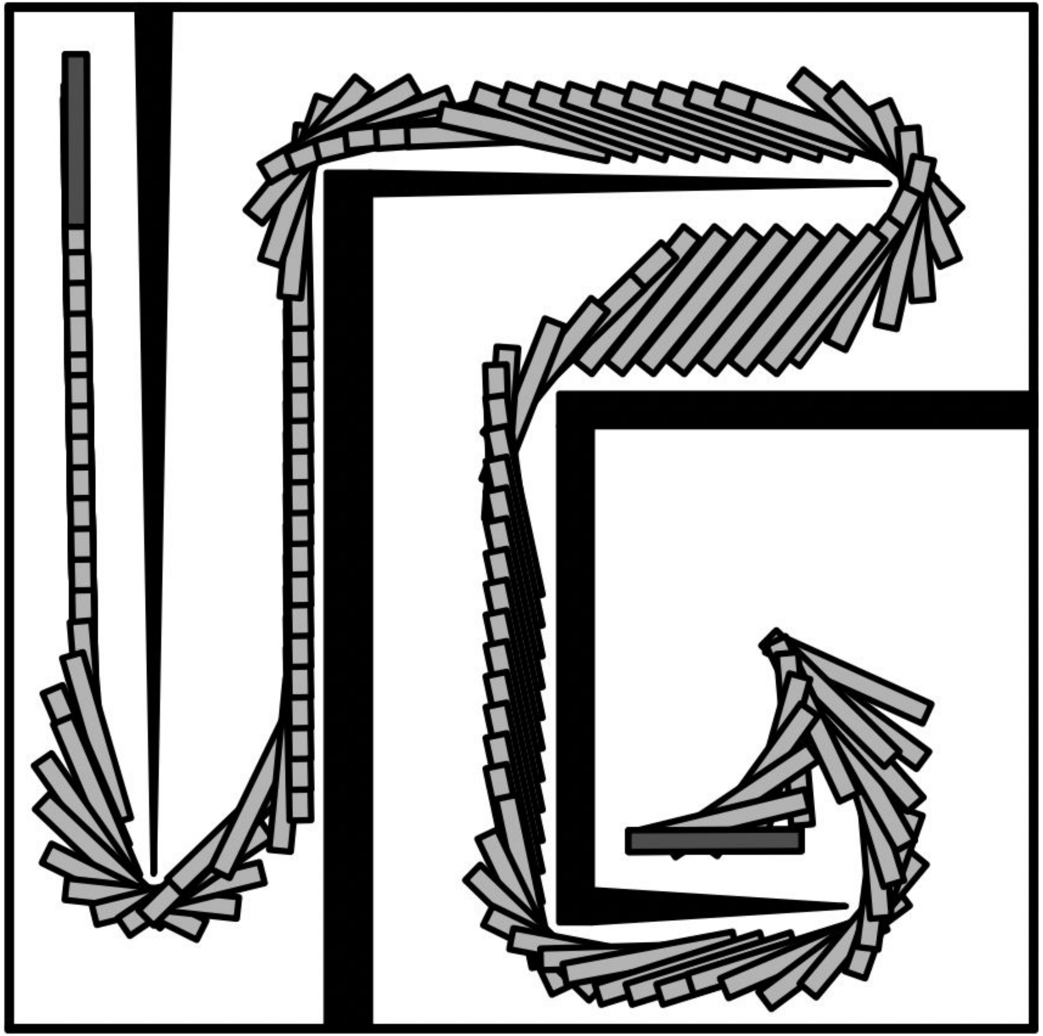
"Flow and Changes in Appearance" Dorsey, Pedersen, & Hanrahan, SIGGRAPH 1996

Outline for Today

- Homework 6 Due Soon & Homework 7 Part 1 Posted
- Last Time: General Position, Robustness, & Exact Computation
- Motivation: Robot Motion Planning
- Previous Lecture: Voronoi Diagram of Segments for Motion Planning
- Degrees of Freedom & Configuration Space
- Trapezoid Map for Motion Planning
- Non-Point, Non-Rotating Robots & Minkowski Sums
- Related Operations: Convolution, Morphology, Accessible Surfaces
- Rotations & Higher Dimensional Configuration Space
- Next Time: ?

What about Rotating Robots?

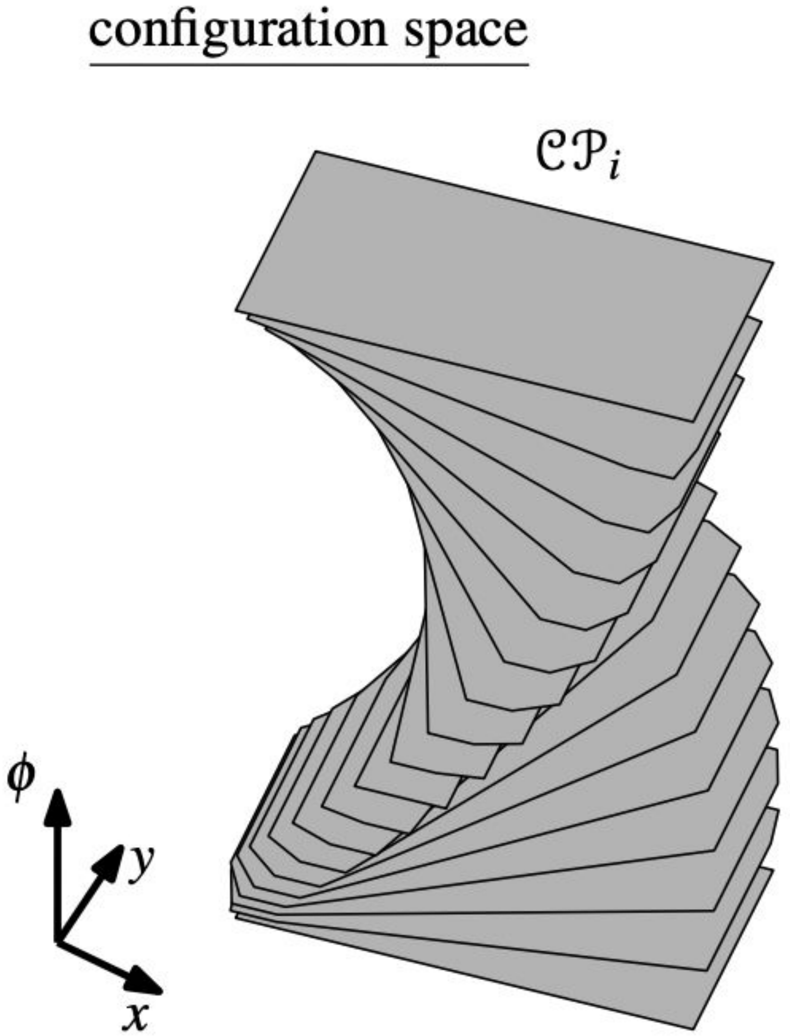
- Rotation may be necessary to complete the task



Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 13

Discretization

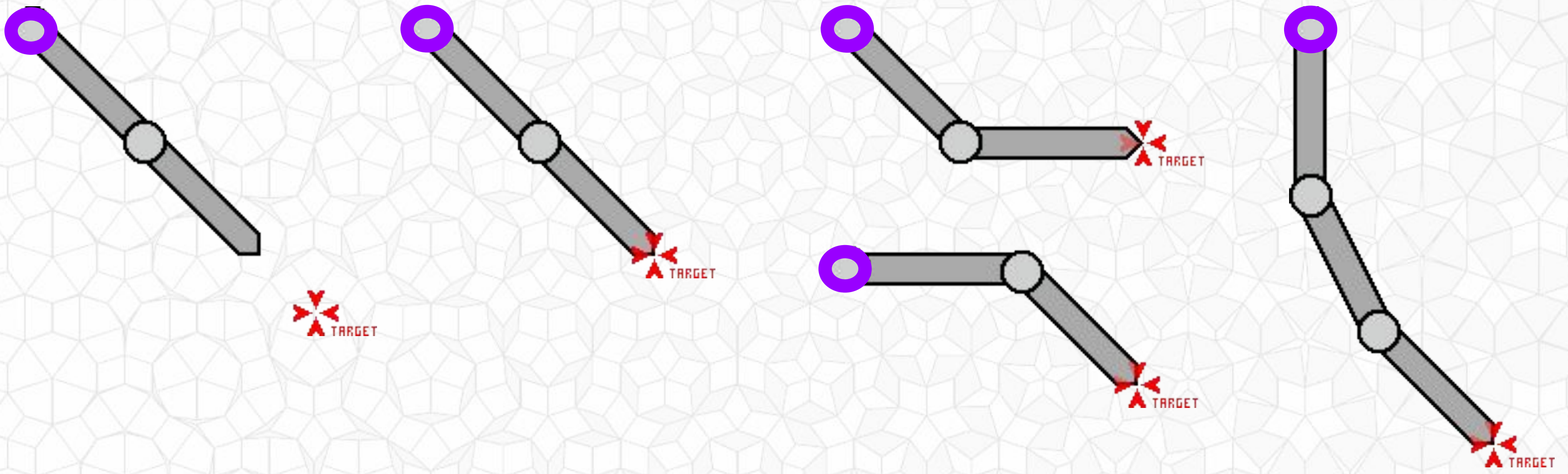
- Discretize the problem into fixed step sizes in rotation
- Search within a single 2D configuration space layer
- Step up or down a layer
- Because error has been introduced, add extra padding around obstacles



Searching Configuration Space

Application:
Robot Motion Planning

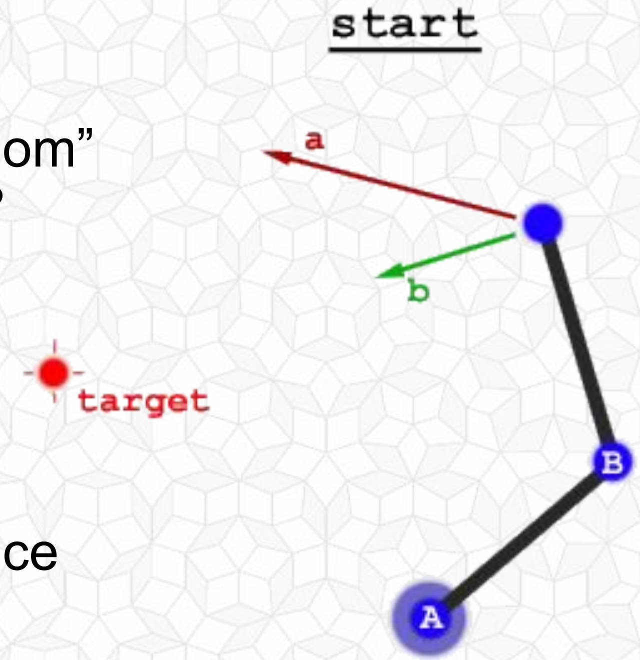
No solutions *One solution* *Two solutions (2D)* *Many solutions*



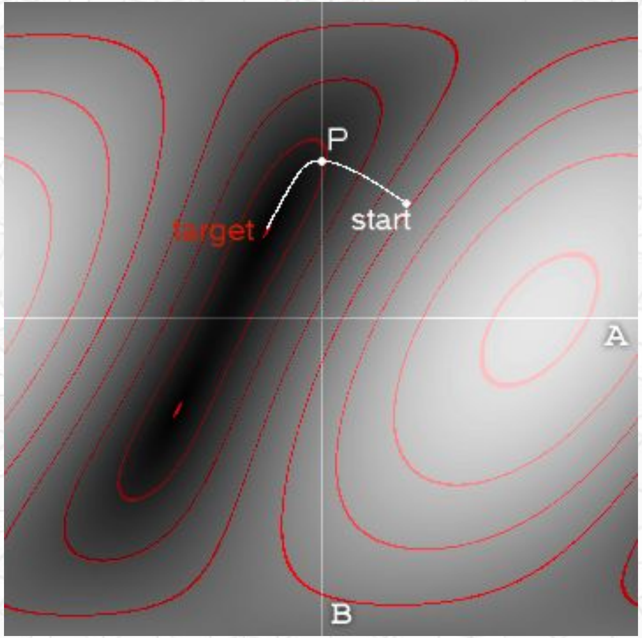
"The good-looking textured light-sourced bouncy fun smart and stretchy page"
Hugo Elias, **http://freespace.virgin.net/hugo.elias/models/m_ik.htm**

Searching Configuration Space

- What are the unknowns?
- What are the "degrees of freedom" of our robot arm?
- More degrees of freedom = higher dimensional configuration space



pose space shaded
by distance to target

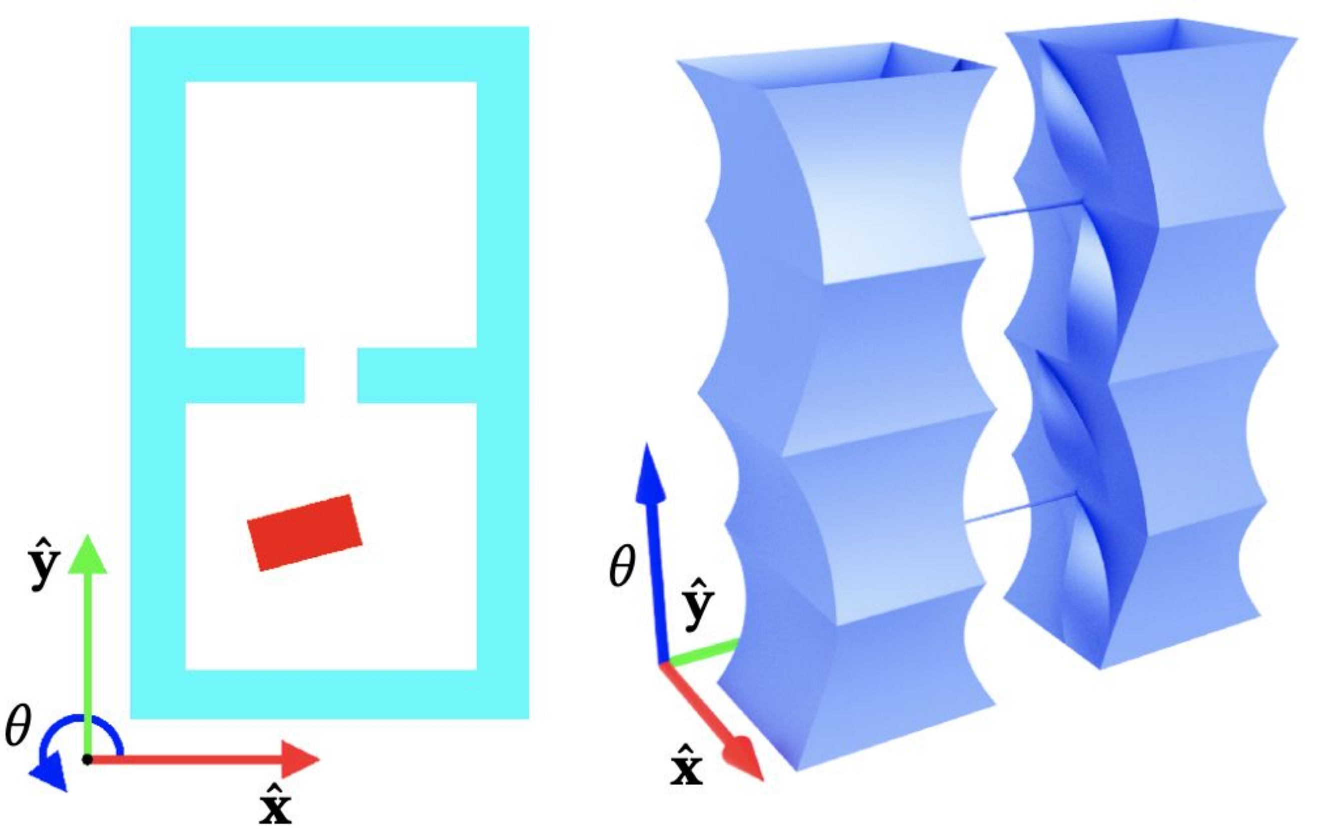


"The good-looking textured light-sourced bouncy fun smart and stretchy page"
Hugo Elias, **http://freespace.virgin.net/hugo.elias/models/m_ik2.htm**

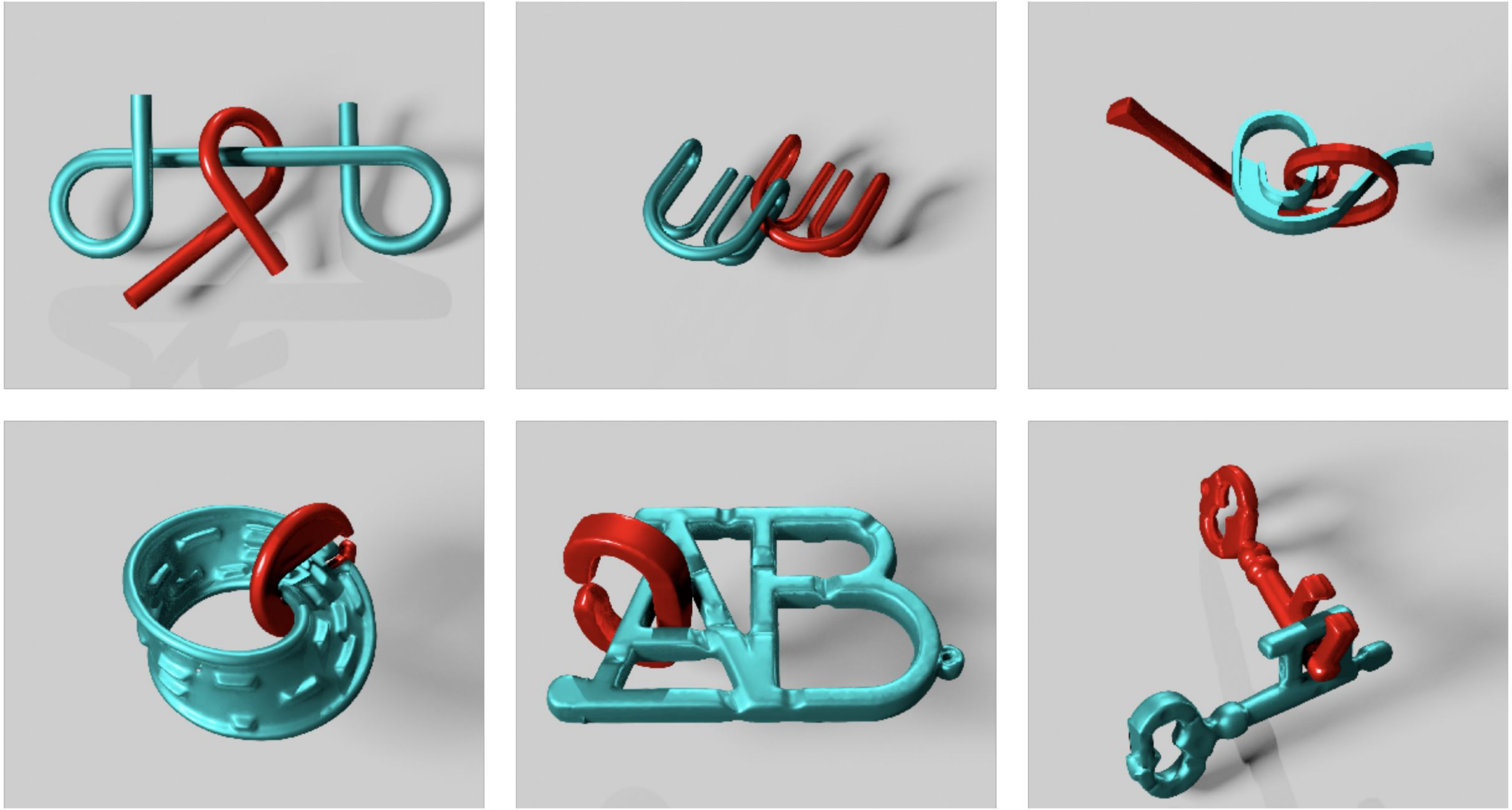
Searching Configuration Space

"C-Space Tunnel Discovery for Puzzle Path Planning",
Zhang, Belfer, Kry, & Voucha,
SIGGRAPH 2020.

- Dimensionality becomes infeasible to construct & exhaustively search
- Randomized search is necessary



"C-Space Tunnel Discovery for Puzzle Path Planning", Zhang, Belfer, Kry, & Voucha, SIGGRAPH 2020.



"Design of Part Feeding and Assembly Processes with Dynamics", Song,Trinkle, Kumar, & Pang, MEAM 2004.

Robotics: Automatic Part Sorting & Orienting

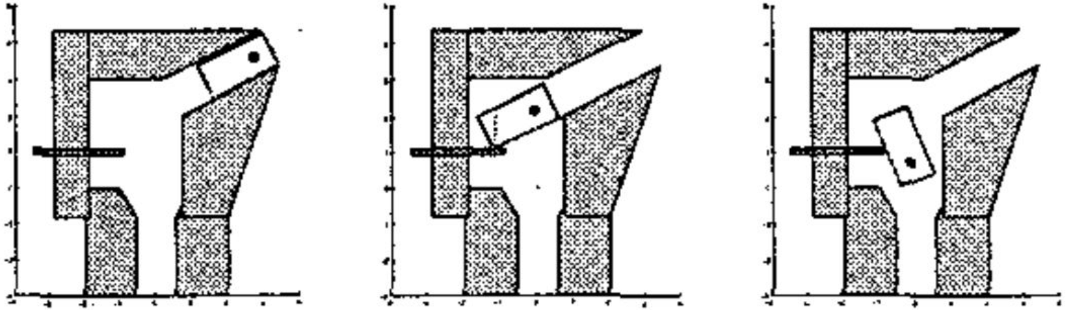


Fig. 9. Peg able to pass through the device with optimal design parameters with center of gravity starting on the right.

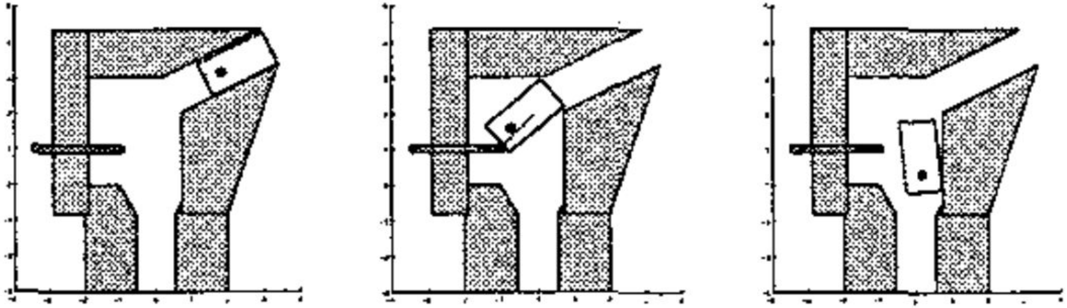


Fig. 10. Peg able to pass through the device with optimal design parameters with center of gravity starting on the left.

Robotics: Automatic Part Sorting & Orienting

“Using Simulation for Planning and Design of Robotic Systems with Intermittent Contact”,
Stephen Berard,
RPI PhD 2009.

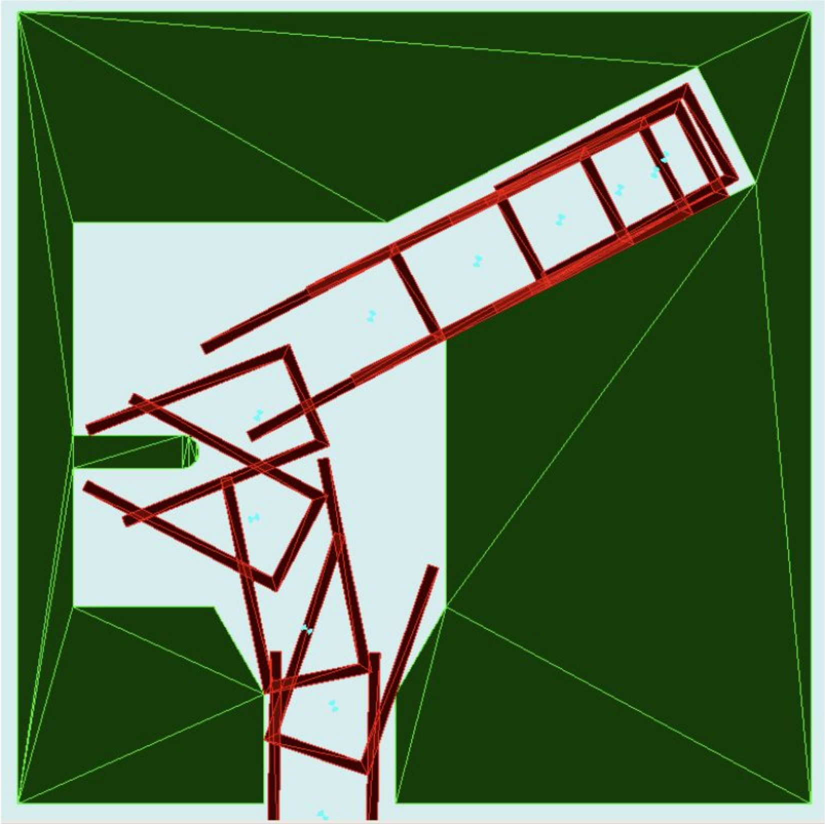
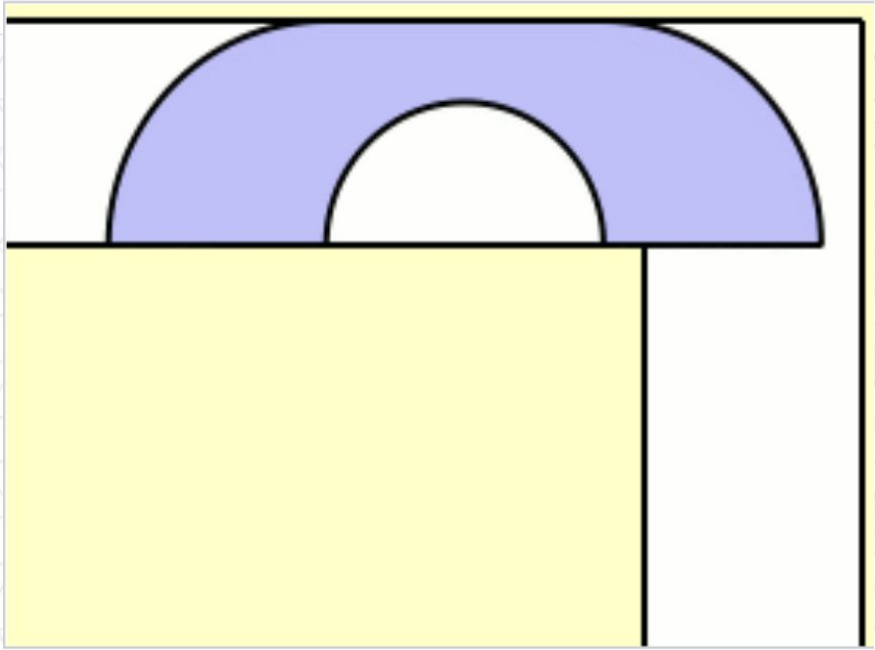
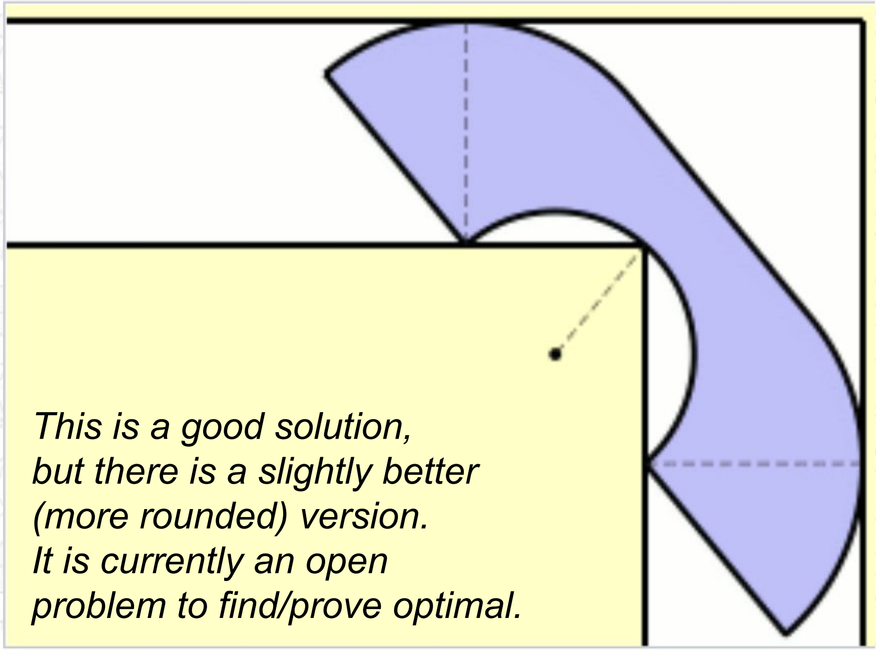


Figure 4.2: Snapshots of the gravity-fed part in the feeder.

Moving Sofa Problem

- Find the largest rigid shape (by area) that can navigate a 90° corner





https://en.wikipedia.org/wiki/Moving_sofa_problem

Outline for Today

- Homework 6 Due Soon & Homework 7 Part 1 Posted
- Last Time: General Position, Robustness, & Exact Computation
- Motivation: Robot Motion Planning
- Previous Lecture: Voronoi Diagram of Segments for Motion Planning
- Degrees of Freedom & Configuration Space
- Trapezoid Map for Motion Planning
- Non-Point, Non-Rotating Robots & Minkowski Sums
- Related Operations: Convolution, Morphology, Accessible Surfaces
- Rotations & Higher Dimensional Configuration Space
- Next Time: ?