Probabilistic Motion Planning

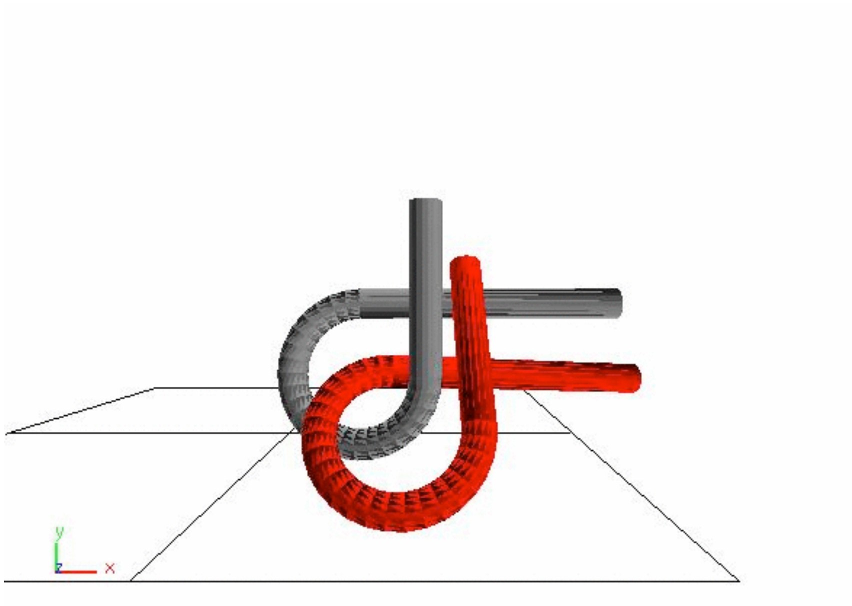
Jyh-Ming Lien

Department of Computer Science
George Mason University

Based on many people's lecture notes

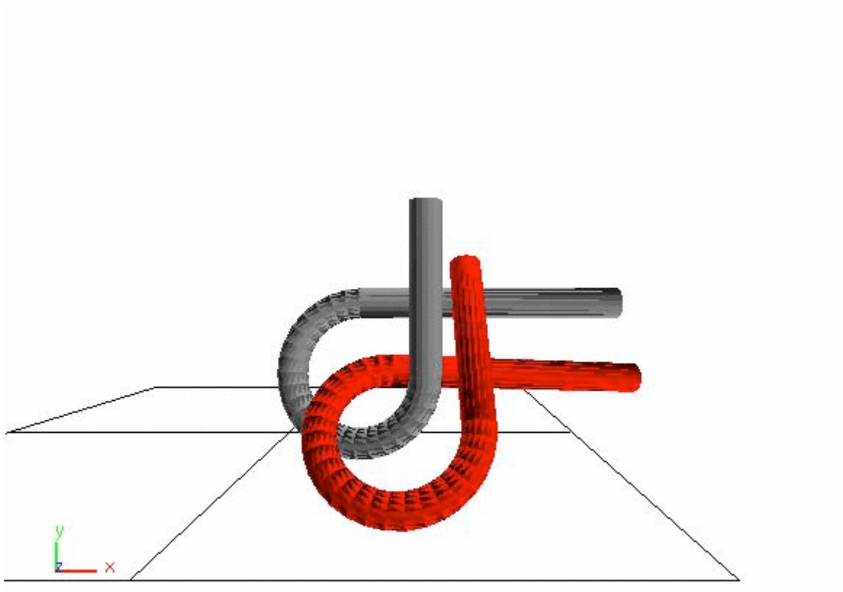
Seth Hutchinson at the University of Illinois at Urbana-Champaign, Leo Joskowicz at Hebrew University, Jean-Claude Latombe at Stanford University, Nancy Amato at Texas A&M University, Burchan Bayazit at Washington University in St. Louis

Hard Motion Planning Problems

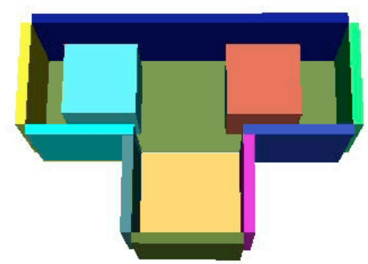


Hard Motion Planning Problems

The Alpha Puzzle

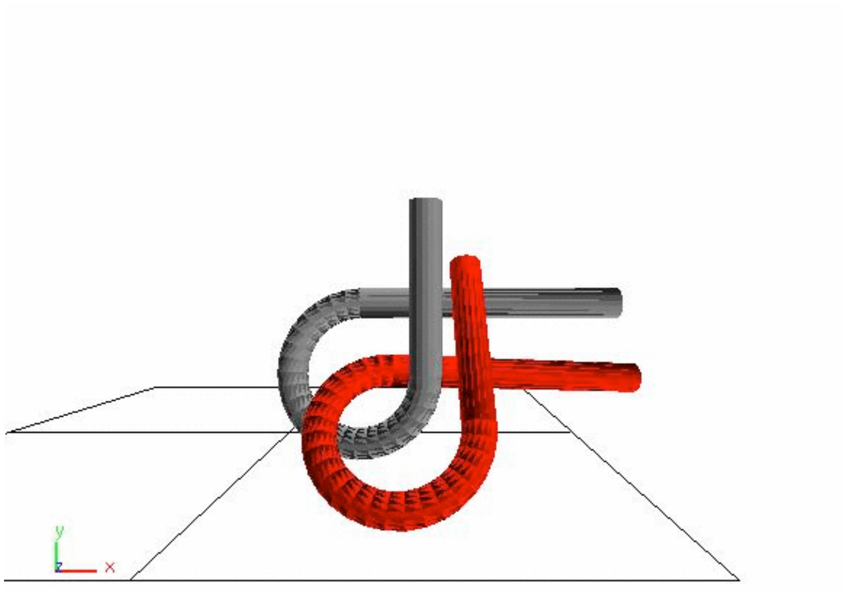


Swapping Cubes Puzzle

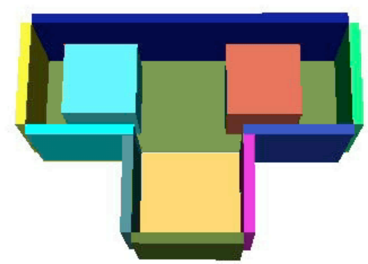


Hard Motion Planning Problems

The Alpha Puzzle



Swapping Cubes Puzzle



Hard Motion Planning Problems

Highly Articulated (Constrained) Systems

Paper Folding

Articulated robot

Polyhedron: 25 dof

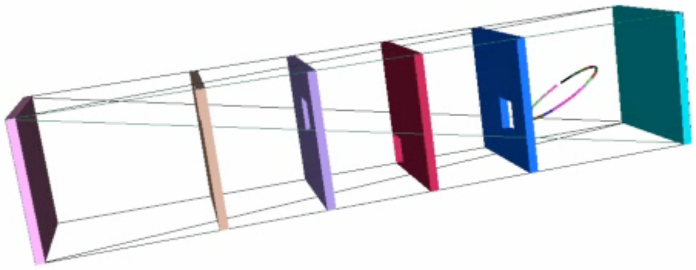
Line: 30 dof

Hard Motion Planning Problems

Highly Articulated (Constrained) Systems

Paper Folding

Articulated robot



Polyhedron: 25 dof

Line: 30 dof

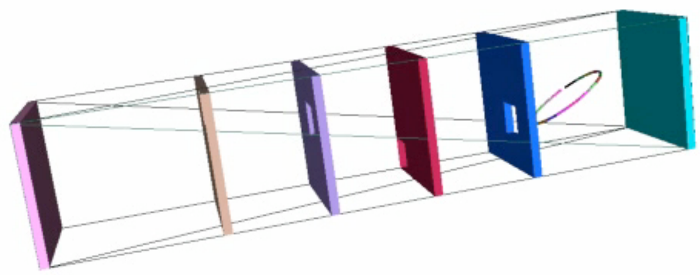
Hard Motion Planning Problems

Highly Articulated (Constrained) Systems

Paper Folding

Polyhedron: 25 dof

Articulated robot



Line: 30 dof

Hard Motion Planning Problems

Highly Articulated (Constrained) Systems

Digital Actors

Reaching and grasping

Hard Motion Planning Problems

Highly Articulated (Constrained) Systems

Digital Actors

Reaching and grasping

Hard Motion Planning Problems

Flocking: Covering, Homing, Shepherding

Motion for coordinated entities

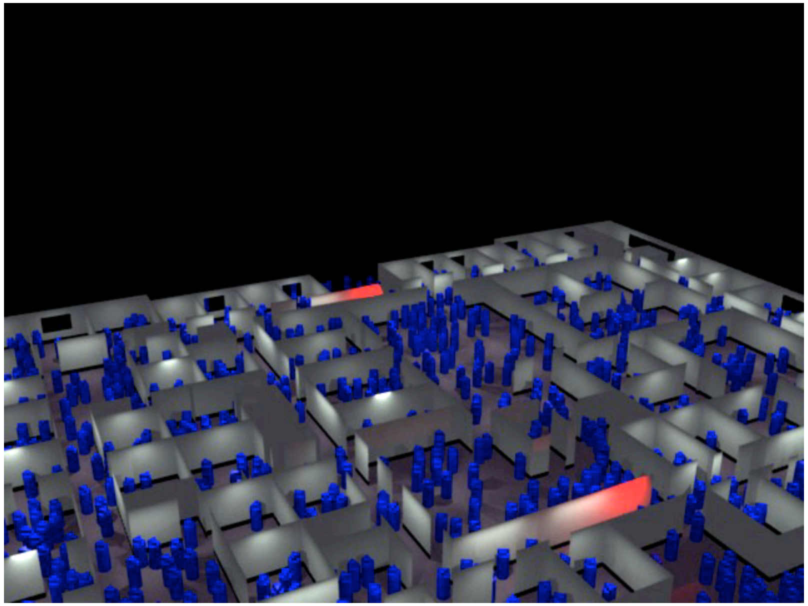
Interactive Navigation of Multiple Agents in Crowded Environments. Jur van den Berg, Sachin Patil, Jason Sewall, Dinesh Manocha, Ming Lin, i3D 2008

Control the motion of coordinated entities

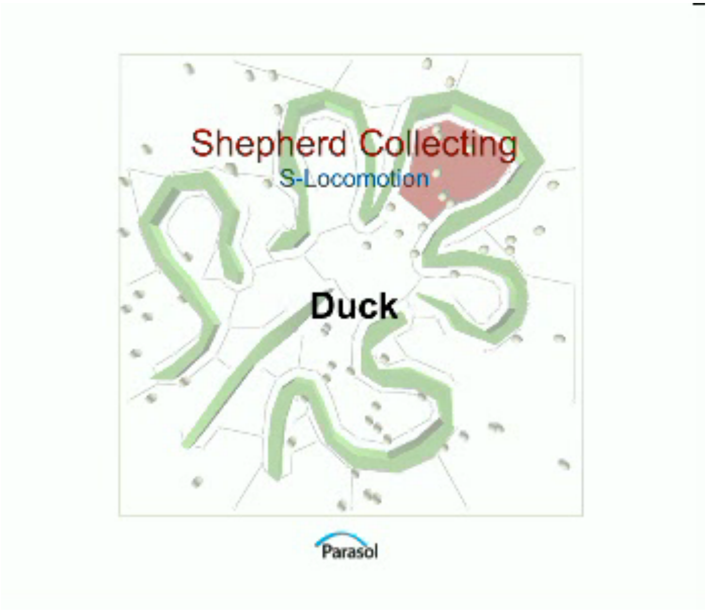
Hard Motion Planning Problems

Flocking: Covering, Homing, Shepherding

Motion for coordinated entities



Interactive Navigation of Multiple Agents in Crowded Environments. Jur van den Berg, Sachin Patil, Jason Sewall, Dinesh Manocha, Ming Lin, i3D 2008

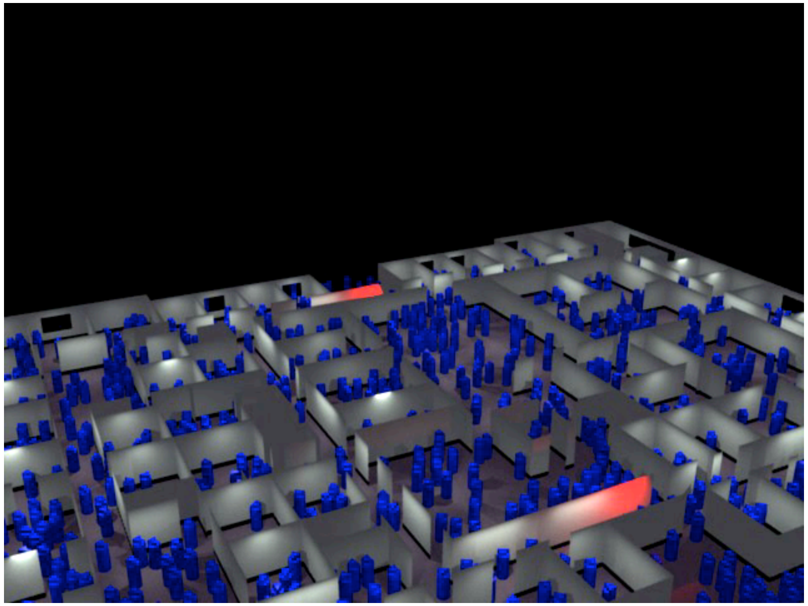


Control the motion of coordinated entities

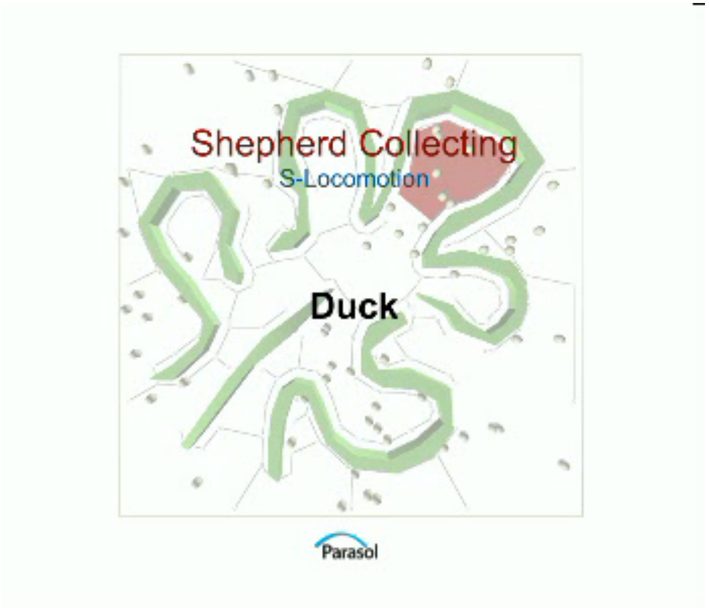
Hard Motion Planning Problems

Flocking: Covering, Homing, Shepherding

Motion for coordinated entities



Interactive Navigation of Multiple Agents in Crowded Environments. Jur van den Berg, Sachin Patil, Jason Sewall, Dinesh Manocha, Ming Lin, i3D 2008

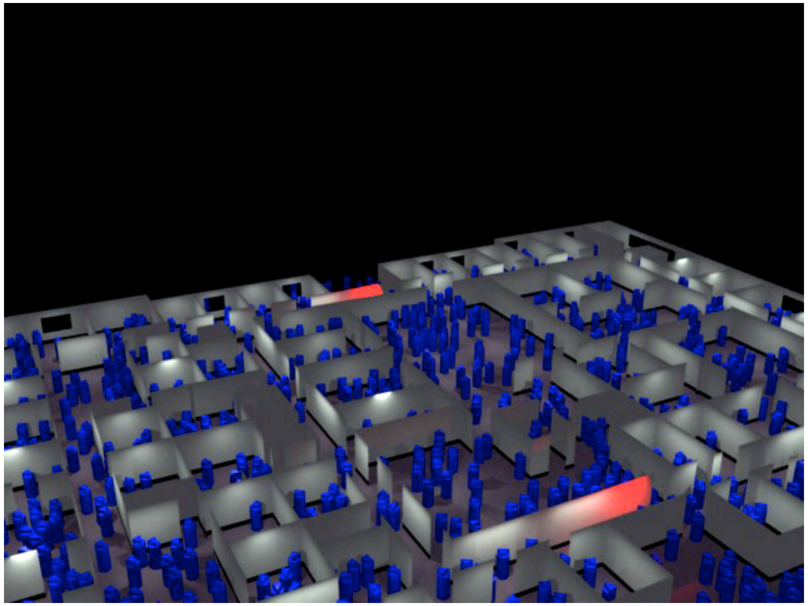


Control the motion of coordinated entities

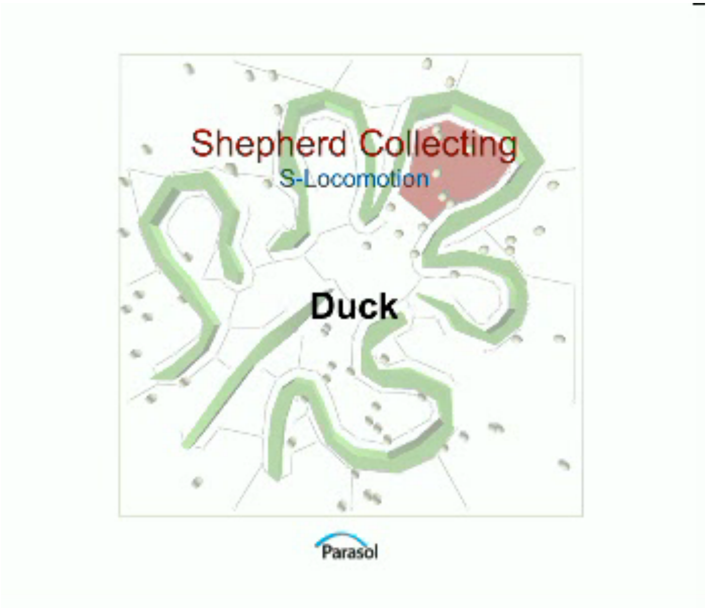
Hard Motion Planning Problems

Flocking: Covering, Homing, Shepherding

Motion for coordinated entities



Interactive Navigation of Multiple Agents in Crowded Environments. Jur van den Berg, Sachin Patil, Jason Sewall, Dinesh Manocha, Ming Lin, i3D 2008

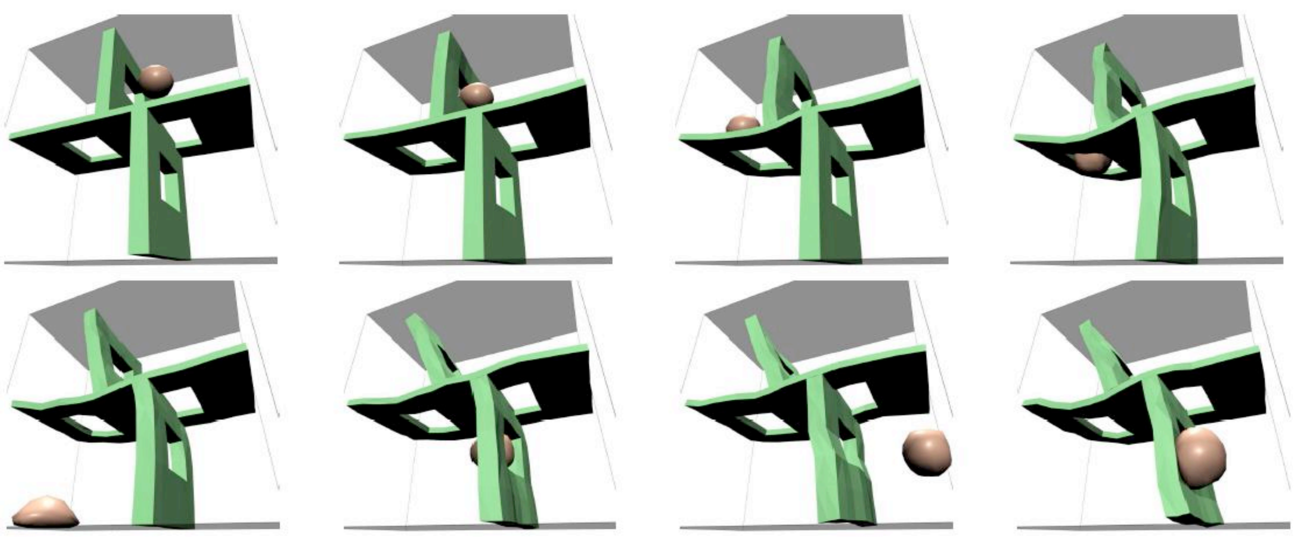


Control the motion of coordinated entities

Hard Motion Planning Problems

Deformable Objects

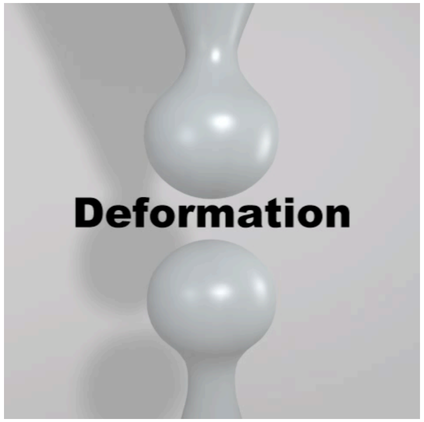
- **Find a path for a deformable object that can deform to avoid collision with obstacles**
  - move a mattress in a house, elastic or air-filled objects, metal sheets or long flexible tubes
  - virtual surgery applications
  - computer animation and games
  - Issue: difficult to find natural deformation efficiently

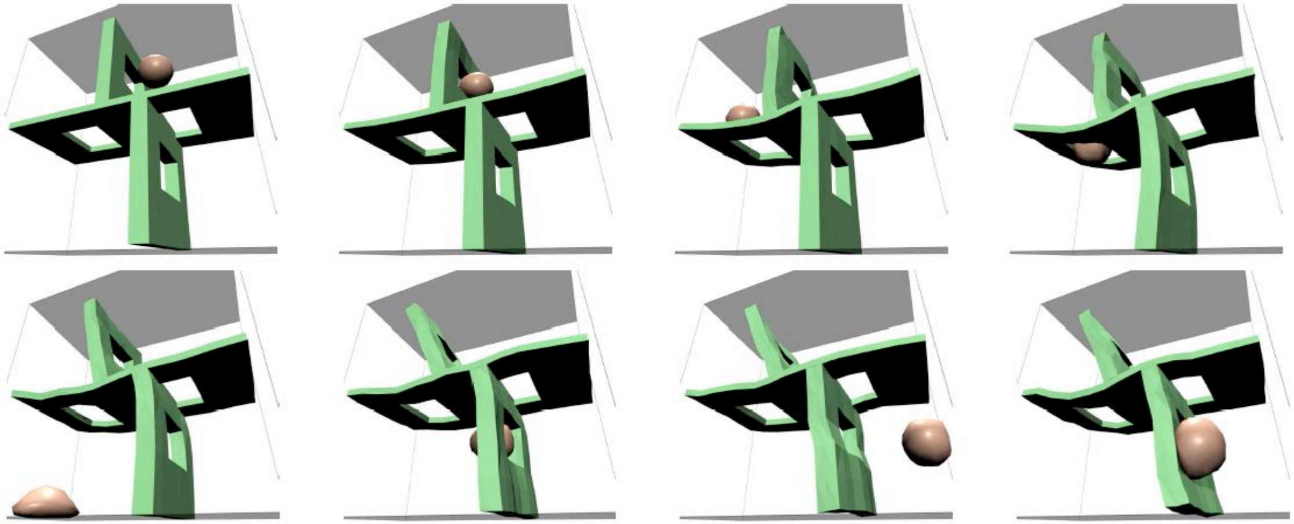


Hard Motion Planning Problems

Deformable Objects

- Find a path for a deformable object that can deform to avoid collision with obstacles
  - move a mattress in a house, elastic or air-filled objects, metal sheets or long flexible tubes
  - virtual surgery applications
  - computer animation and games
  - Issue: difficult to find natural deformation efficiently



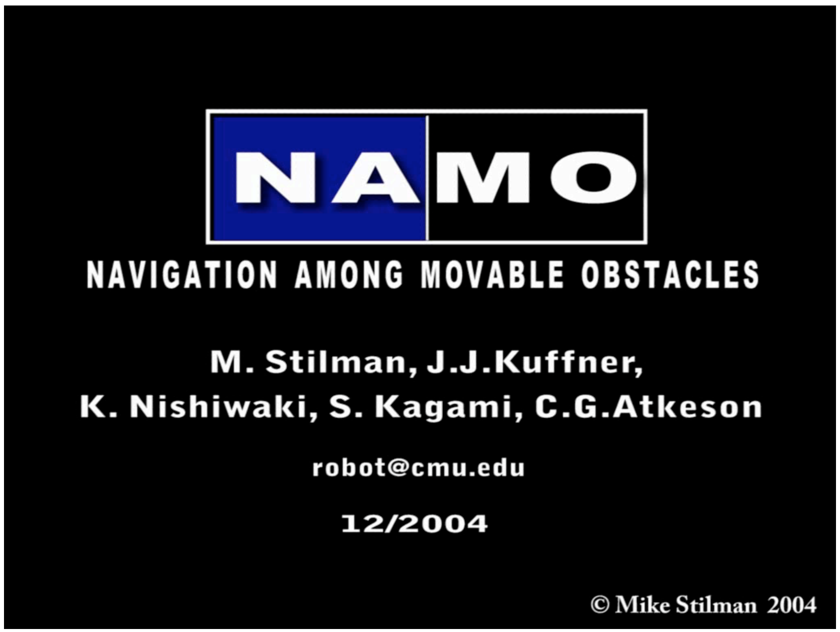


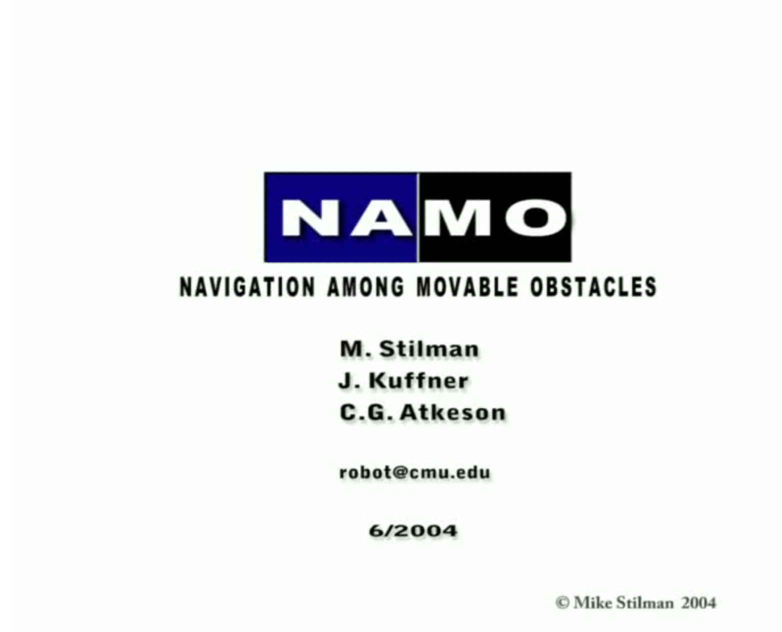
Hard Motion Planning Problems

Movable Objects

- **M. Stilman and J.J. Kuffner** Planning Among Movable Obstacles with Artificial Constraints
  Workshop on the Algorithmic Foundations of Robotics, July, 2006

Hard Motion Planning Problems
Movable Objects

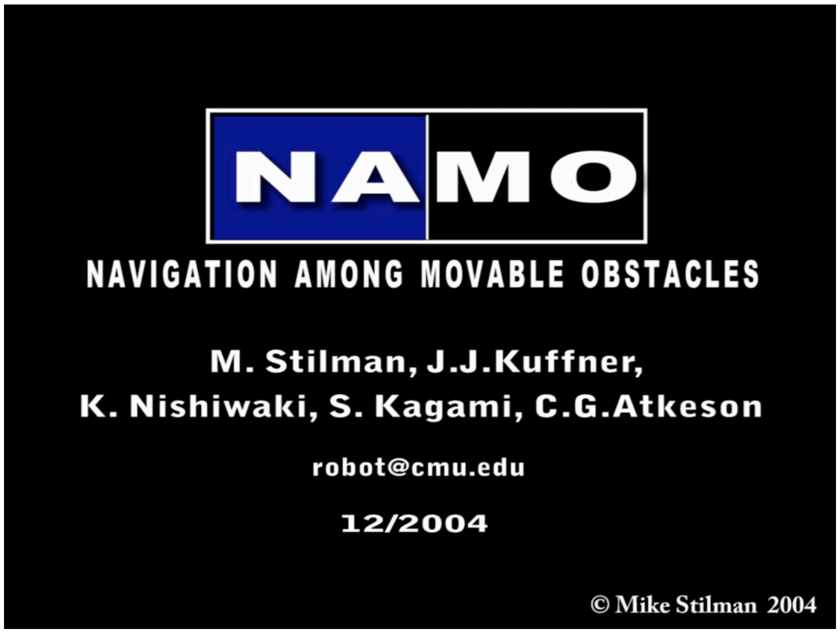


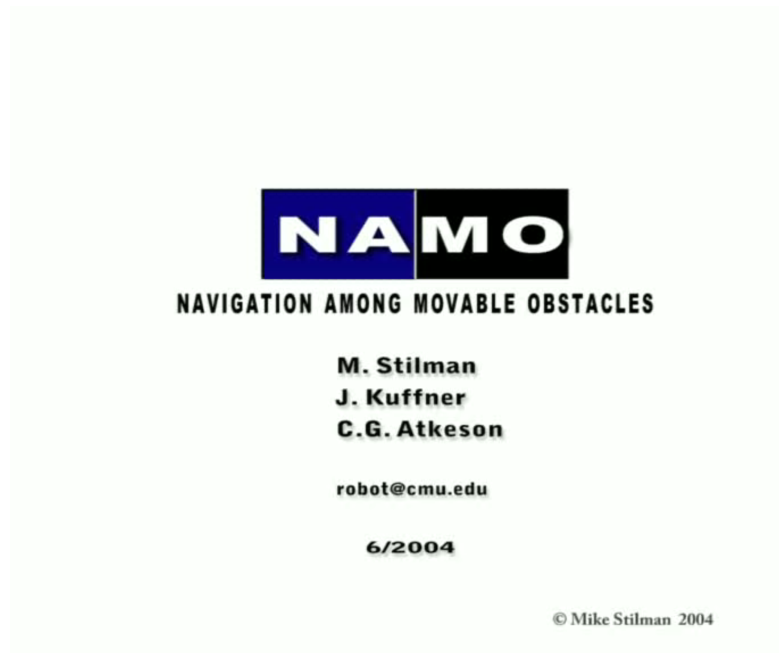


- **M. Stilman and J.J. Kuffner** Planning Among Movable Obstacles with Artificial Constraints
  Workshop on the Algorithmic Foundations of Robotics, July, 2006

Hard Motion Planning Problems

Movable Objects





- **M. Stilman and J.J. Kuffner** [Planning Among Movable Obstacles with Artificial Constraints]
  Workshop on the Algorithmic Foundations of Robotics, July, 2006

Hard Motion Planning Problems

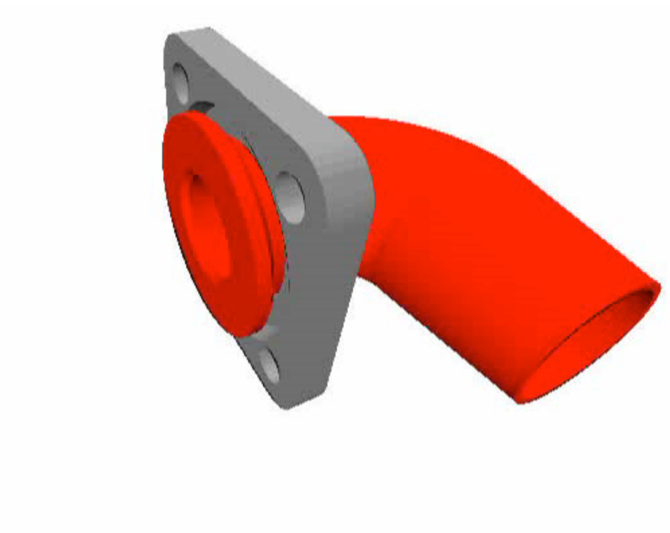
Intelligent CAD Applications

- **Using Motion Planning to Test Design Requirements**:
  - Accessibility for servicing/assembly tested on physical "mock ups"
  - Digital testing saves time and money, is more accurate, enables more extensive testing, and is useful for training (VR or e-manuals)

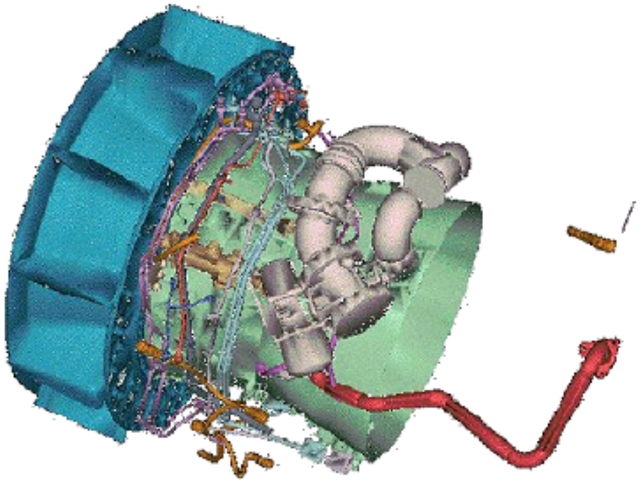
Hard Motion Planning Problems
Intelligent CAD Applications

- **Using Motion Planning to Test Design Requirements**:
  - Accessibility for servicing/assembly tested on physical "mock ups"
  - Digital testing saves time and money, is more accurate, enables more extensive testing, and is useful for training (VR or e-manuals)

Maintainability Problems:
Mechanical Designs from GE



flange



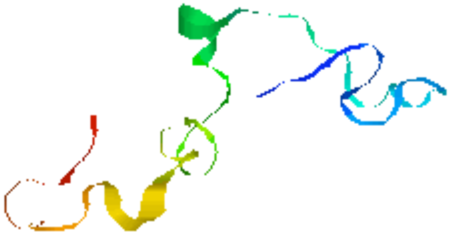
Airplane engine

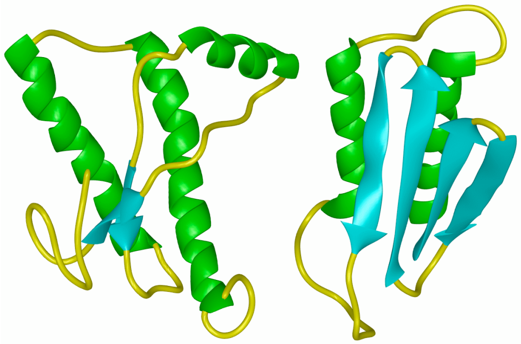
Hard Motion Planning Problems

computational biology & chemistry

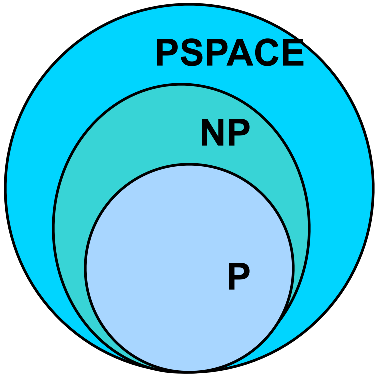
Motion of molecules

- help understand important interactions - protein structure/function prediction
- diseases such as Alzheimer's and Mad Cow are related to misfolded proteins





The Complexity of Motion Planning

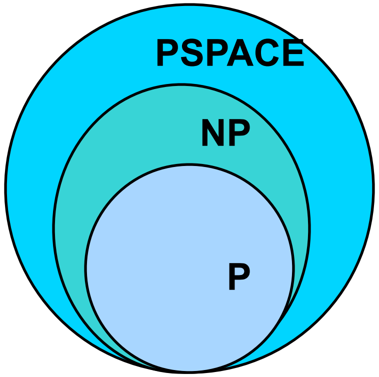


General motion planning problem is
PSPACE-hard **[Reif 79, Hopcroft et al. 84 & 86]**
PSPACE-complete **[Canny 87]**

The best deterministic algorithm known has running time that is exponential in the dimension of the robot's C-space
[Canny 86]

- C-space has high dimension - 6D for rigid body in 3-space
- simple obstacles have complex C-obstacles $\Rightarrow$ impractical to compute explicit representation of freespace for more than 4 or 5 dof

The Complexity of Motion Planning



General motion planning problem is
PSPACE-hard **[Reif 79, Hopcroft et al. 84 & 86]**
PSPACE-complete **[Canny 87]**

The best deterministic algorithm known has running time that is exponential in the dimension of the robot's C-space
[Canny 86]

- C-space has high dimension - 6D for rigid body in 3-space
- simple obstacles have complex C-obstacles → impractical to compute explicit representation of freespace for more than 4 or 5 dof

So … attention has turned to *randomized algorithms*

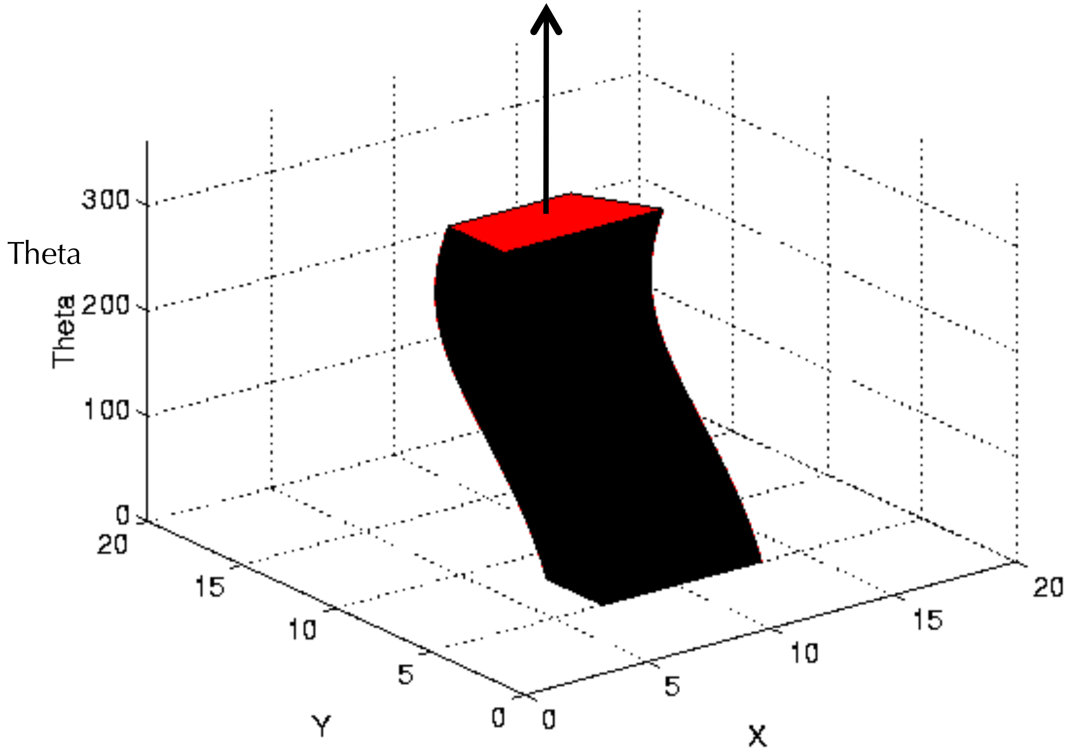
Probabilistic Methods

- Avoid computing C-obstacles
  - Too difficult to compute efficiently
- Idea: Sacrifice completeness to gain simplicity and efficiency
- Probabilistic Methods
  - Graph based
  - Tree based

Probabilistic Roadmap Method

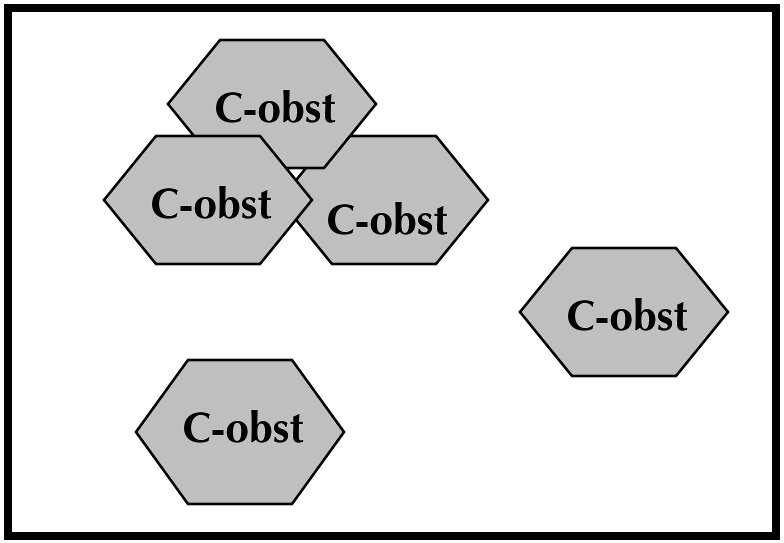
[Kavraki, Svestka, Latombe,Overmars 1996]

unknown



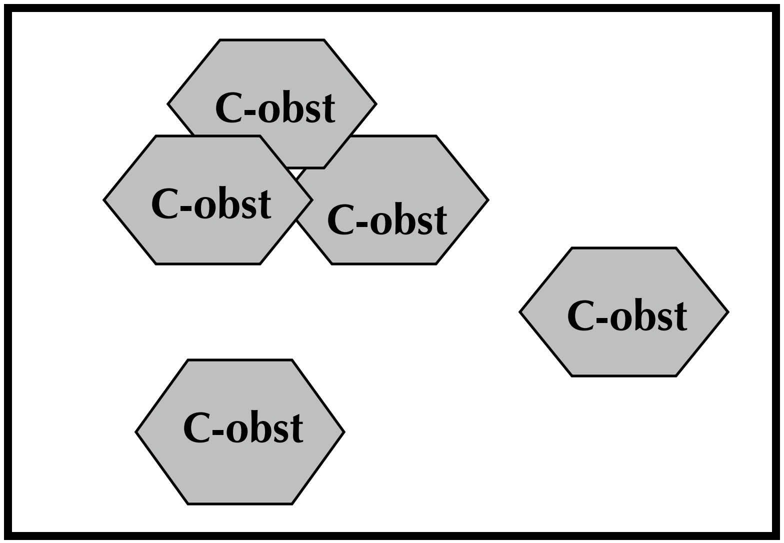
Probabilistic Roadmap Method

C-space



Probabilistic Roadmap Method

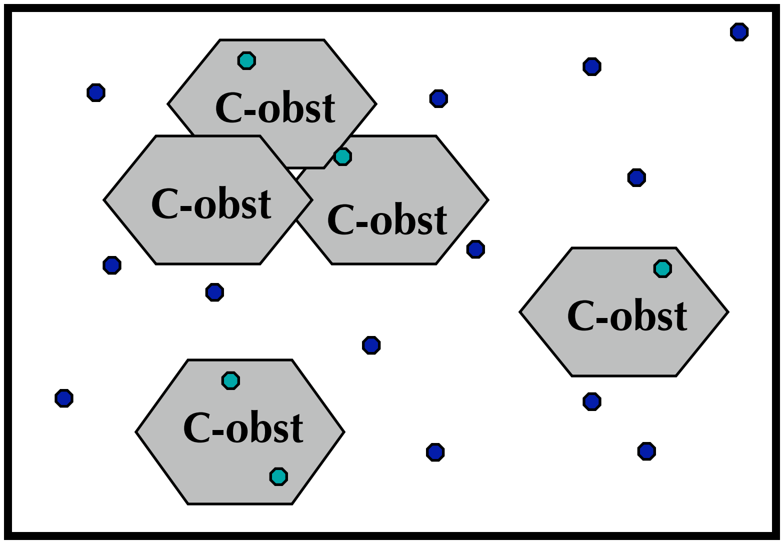
C-space



Roadmap Construction (Pre-processing)

Probabilistic Roadmap Method

C-space

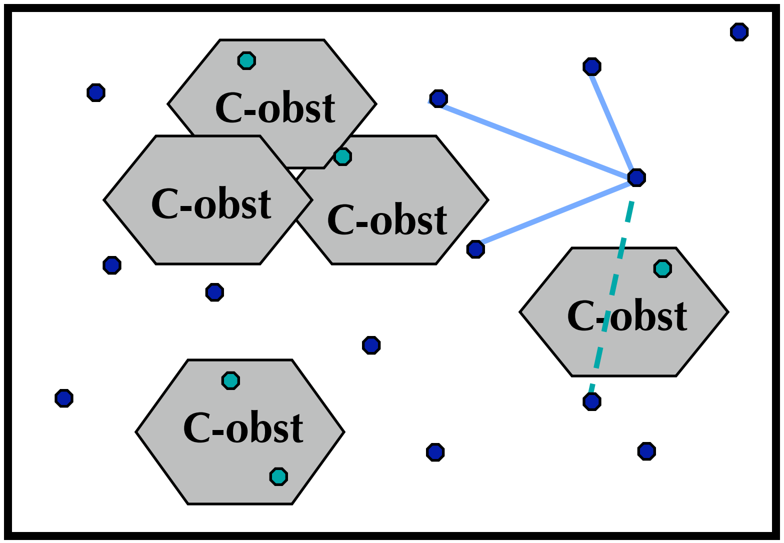


Roadmap Construction (Pre-processing)

1. Randomly generate robot configurations (nodes)
   - discard nodes that are invalid

Probabilistic Roadmap Method

C-space



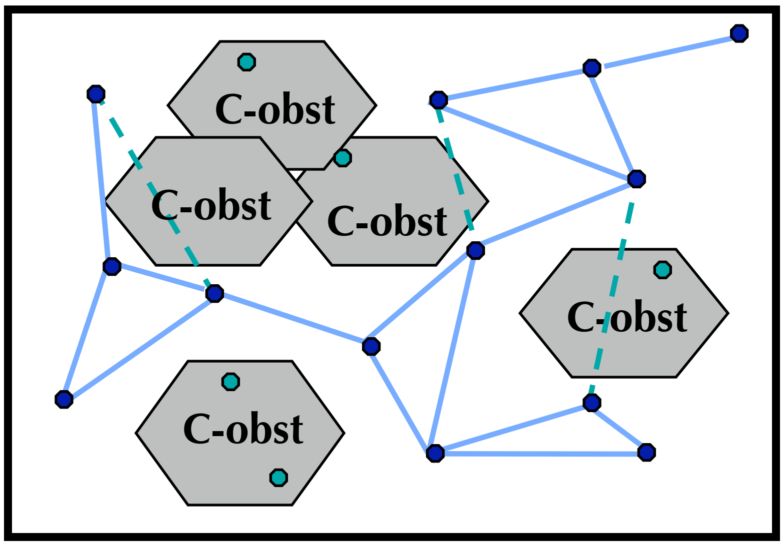
Roadmap Construction (Pre-processing)

1. Randomly generate robot configurations (nodes)
   - discard nodes that are invalid

2. Connect pairs of nodes to form **roadmap**
   - simple, deterministic *local planner* (e.g., straightline)
   - discard paths that are invalid

Probabilistic Roadmap Method

C-space

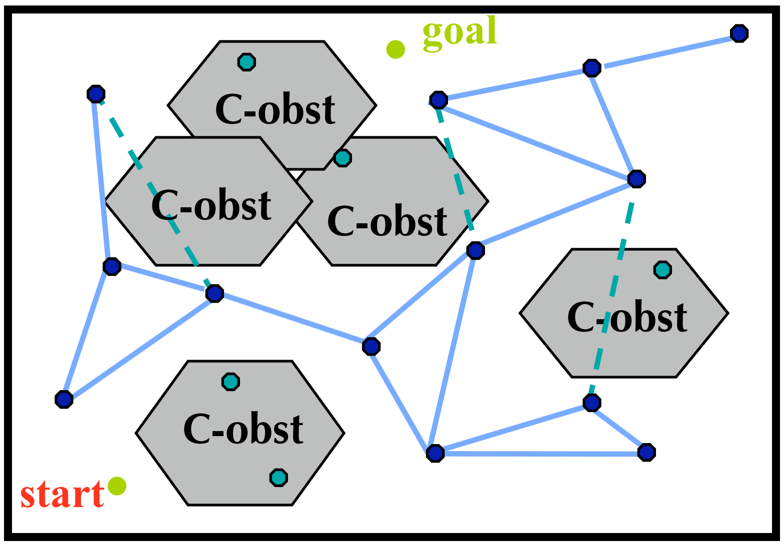


Roadmap Construction (Pre-processing)

1. Randomly generate robot configurations (nodes)
   - discard nodes that are invalid
2. Connect pairs of nodes to form **roadmap**
   - simple, deterministic *local planner* (e.g., straightline)
   - discard paths that are invalid

Probabilistic Roadmap Method

C-space



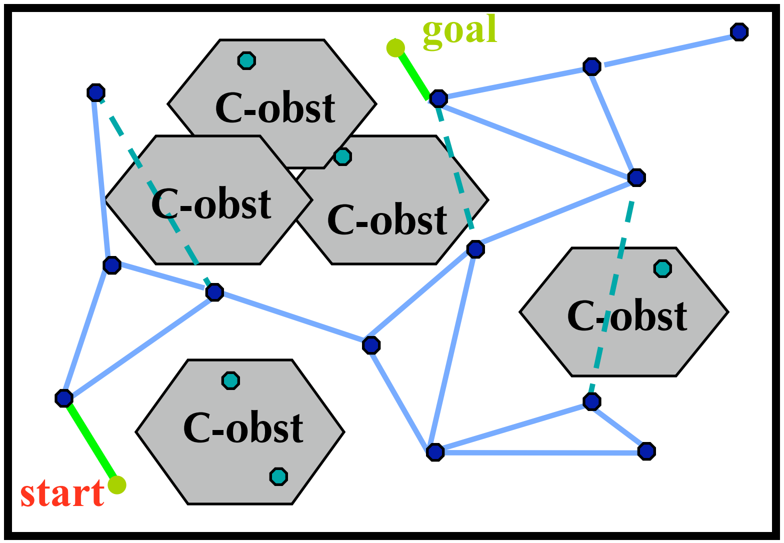
Roadmap Construction (Pre-processing)

1. Randomly generate robot configurations (nodes)
   - discard nodes that are invalid
2. Connect pairs of nodes to form **roadmap**
   - simple, deterministic *local planner* (e.g., straightline)
   - discard paths that are invalid

Query processing

Probabilistic Roadmap Method

C-space



Roadmap Construction (Pre-processing)

1. Randomly generate robot configurations (nodes)
   - discard nodes that are invalid

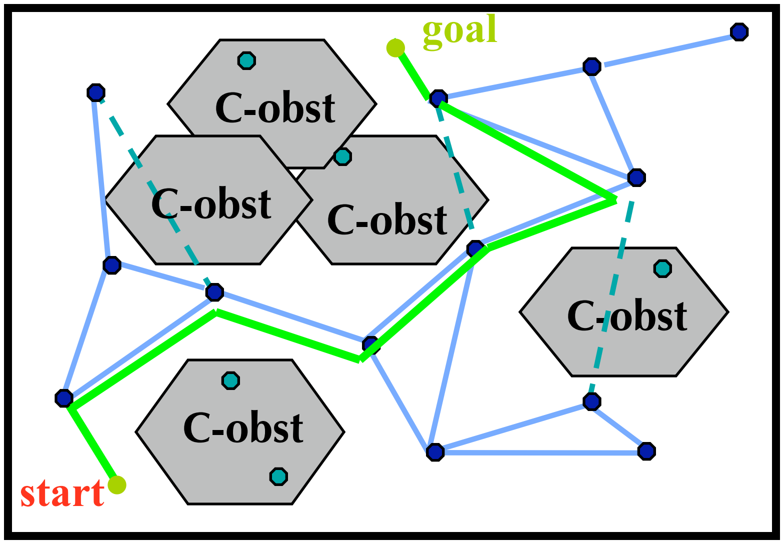
2. Connect pairs of nodes to form **roadmap**
   - simple, deterministic *local planner* (e.g., straightline)
   - discard paths that are invalid

Query processing

1. Connect *start* and *goal* to roadmap

Probabilistic Roadmap Method

C-space



Roadmap Construction (Pre-processing)

1. Randomly generate robot configurations (nodes)
   - discard nodes that are invalid
2. Connect pairs of nodes to form **roadmap**
   - simple, deterministic *local planner* (e.g., straightline)
   - discard paths that are invalid

Query processing

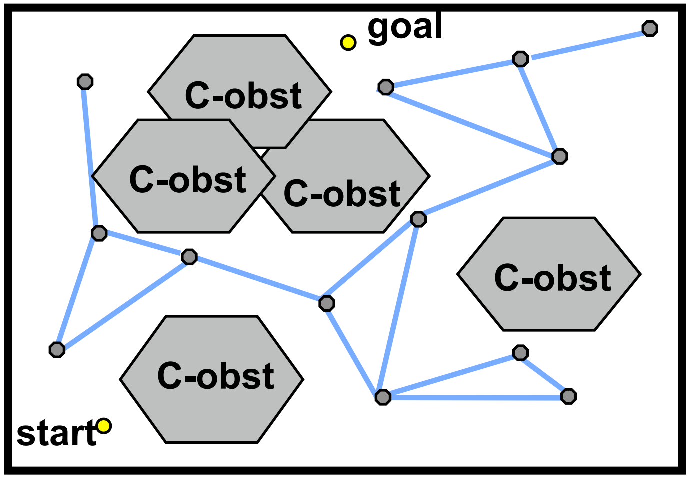
1. Connect *start* and *goal* to roadmap
2. Find path in roadmap between *start* and *goal*
   - regenerate plans for edges in roadmap

Probabilistic Roadmap Method

- Important sub-routines
  - Generate random configurations
  - Local planners
  - Distance metrics
  - Selecting k-nearest neighbors (becoming dominant in high dimensional space)
  - Collision detection (>80% computation)

Note: We don't store paths in the edges

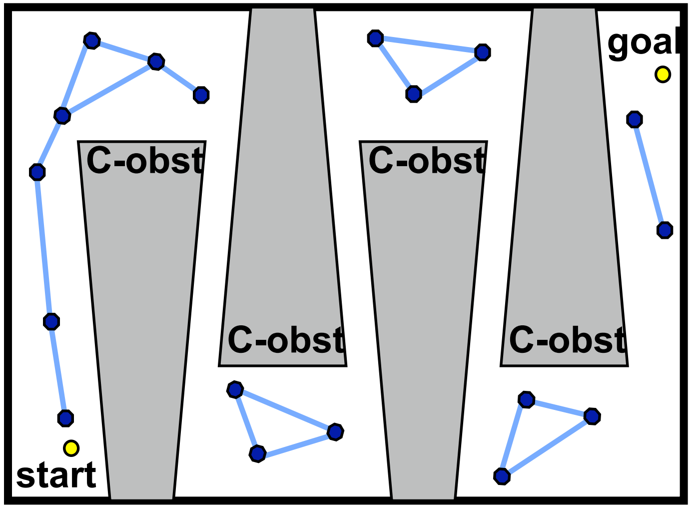
PRMs: Pros & Cons



PRMs: The Good News

1. PRMs are *probabilistically complete*
2. PRMs apply easily to high-dimensional C-space
3. PRMs support fast queries w/ enough preprocessing

Many success stories where PRMs solve previously unsolved problems



PRMs: The Bad News

1. PRMs don't work as well for some problems:
– unlikely to sample nodes in *narrow passages*
– hard to sample/connect nodes on constraint surfaces

Related Work (selected)

- **Probabilistic Roadmap Methods**
  - Uniform Sampling (original) [Kavraki, Latombe, Overmars, Svestka, 92, 94, 96]
  - Obstacle-based PRM (OBPRM) [Amato et al, 98]
  - PRM Roadmaps in Dilated Free space [Hsu et al, 98]
  - Gaussian Sampling PRMs [Boor/Overmars/van der Steppen 99]
  - Bridge test [Hsu et al 03]
  - Visibility Roadmaps [Laumond et al 99]
  - Using Medial Axis [Kavraki et al 99, Lien/Thomas/Wilmarth/Amato/Stiller 99, 03, Lin et al 00]
  - Generating Contact Configurations [Xiao et al 99]
  - Using workspace clues
  - …

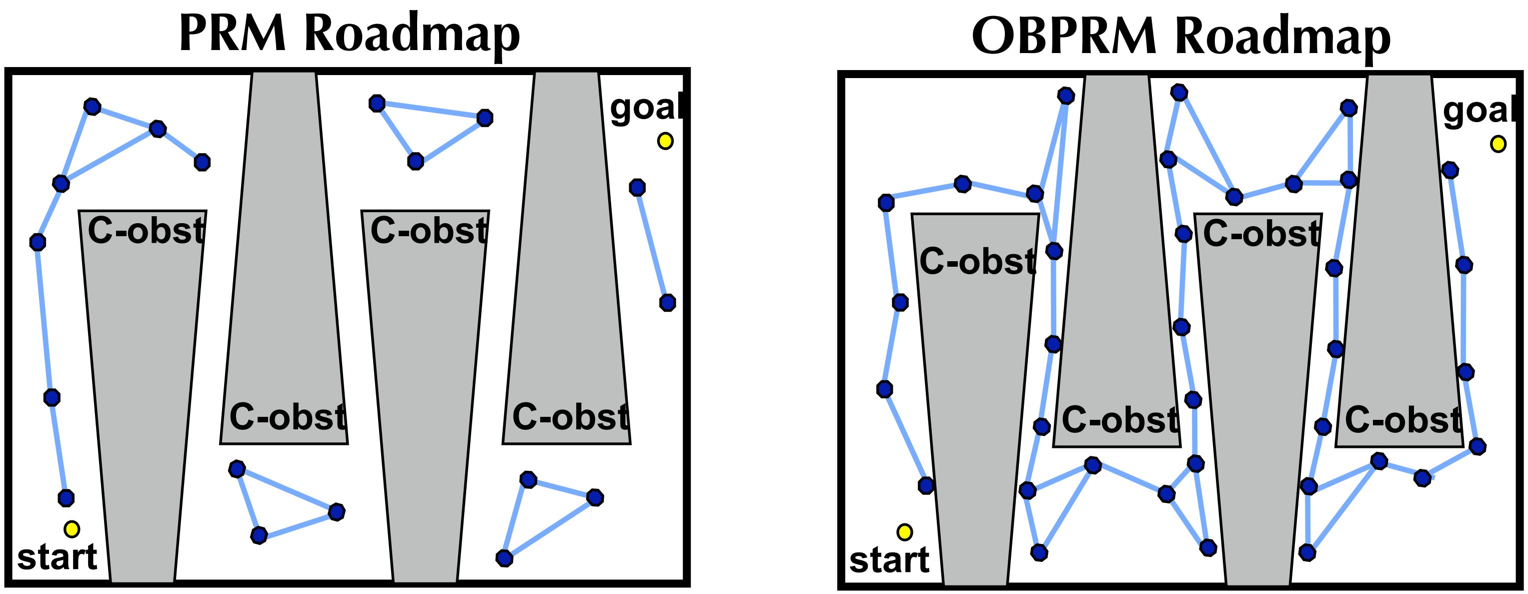
Related Work (selected)

- **Probabilistic Roadmap Methods**
  - Uniform Sampling (original) [Kavraki, Latombe, Overmars, Svestka, 92, 94, 96]
  - Obstacle-based PRM (OBPRM) [Amato et al, 98]
  - PRM Roadmaps in Dilated Free space [Hsu et al, 98]
  - Gaussian Sampling PRMs [Boor/Overmars/van der Steppen 99]
  - Bridge test [Hsu et al 03]
  - Visibility Roadmaps [Laumond et al 99]
  - Using Medial Axis [Kavraki et al 99, Lien/Thomas/Wilmarth/Amato/Stiller 99, 03, Lin et al 00]
  - Generating Contact Configurations [Xiao et al 99]
  - Using workspace clues
  - ...

An Obstacle-Based PRM

To Navigate Narrow Passages we must sample in them

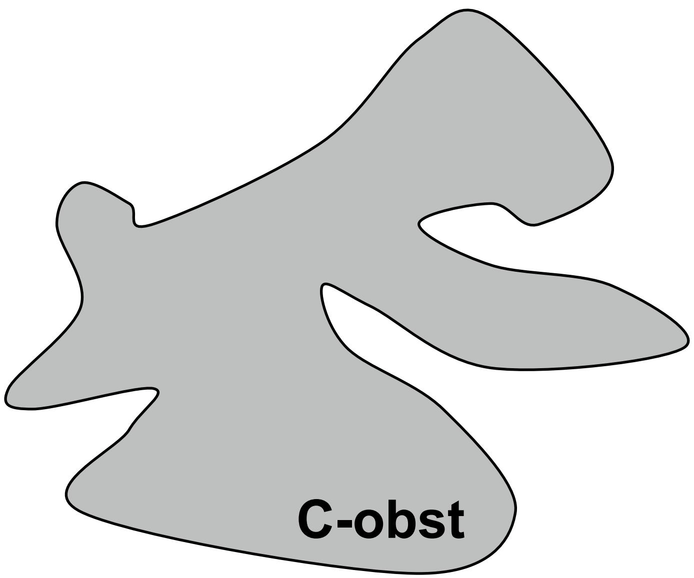
- most PRM nodes are where planning is easy (not needed)



Idea: Can we sample nodes near C-obstacle surfaces?

- we cannot explicitly construct the C-obstacles...
- we do have models of the (workspace) obstacles...

Finding Points on C-obstacles

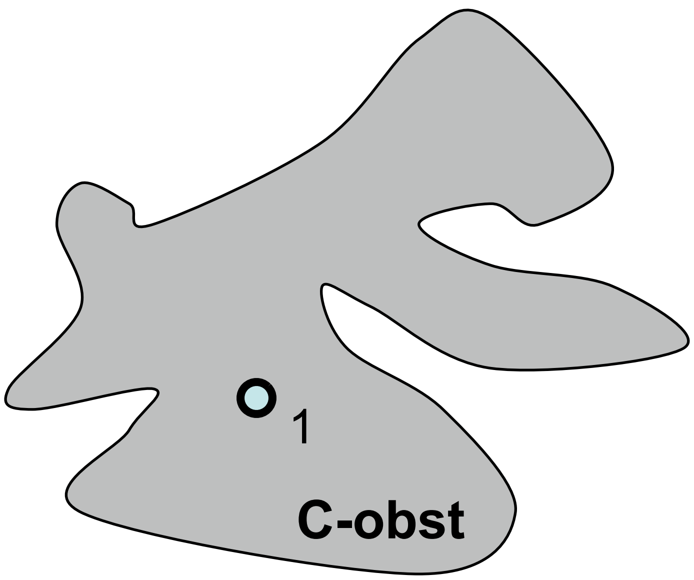


Basic Idea (for workspace obstacle S)

1. Find a point in S's C-obstacle (robot placement colliding with S)
2. Select a random direction in C-space
3. Find a free point in that direction
4. Find boundary point between them using binary search (collision checks)

Note: we can use more sophisticated heuristics to try to cover C-obstacle

Finding Points on C-obstacles

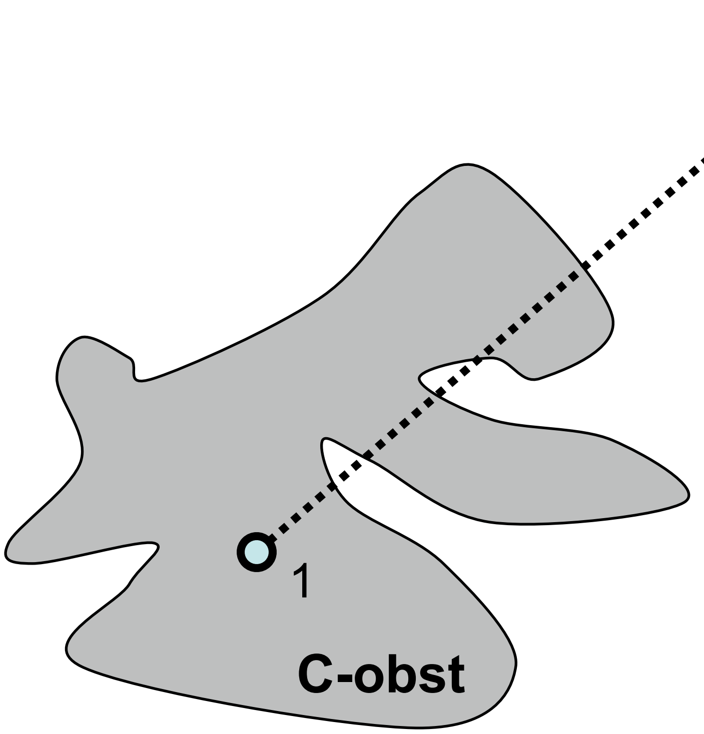


Basic Idea (for workspace obstacle S)

1. Find a point in S's C-obstacle (robot placement colliding with S)
2. Select a random direction in C-space
3. Find a free point in that direction
4. Find boundary point between them using binary search (collision checks)

Note: we can use more sophisticated heuristics to try to cover C-obstacle

Finding Points on C-obstacles

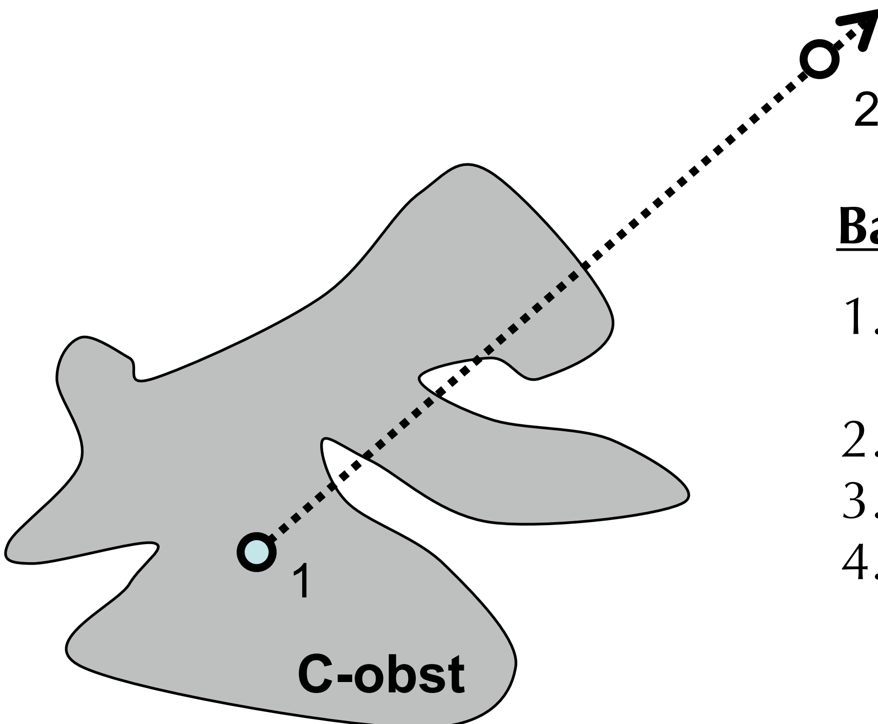


Basic Idea (for workspace obstacle S)

1. Find a point in S's C-obstacle
   (robot placement colliding with S)
2. Select a random direction in C-space
3. Find a free point in that direction
4. Find boundary point between them
   using binary search (collision checks)

Note: we can use more sophisticated heuristics to try to cover C-obstacle

Finding Points on C-obstacles

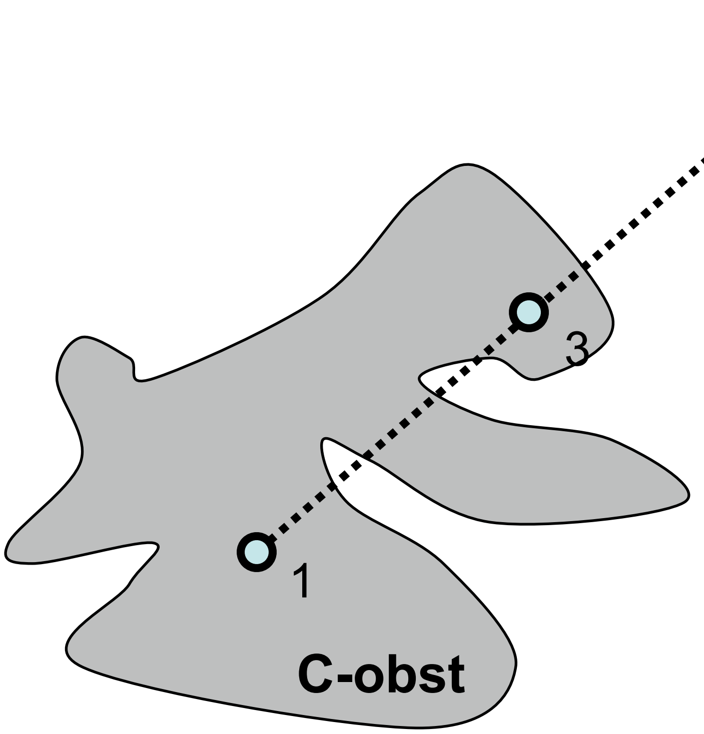


Basic Idea (for workspace obstacle S)

1. Find a point in S's C-obstacle (robot placement colliding with S)
2. Select a random direction in C-space
3. Find a free point in that direction
4. Find boundary point between them using binary search (collision checks)

Note: we can use more sophisticated heuristics to try to cover C-obstacle

Finding Points on C-obstacles

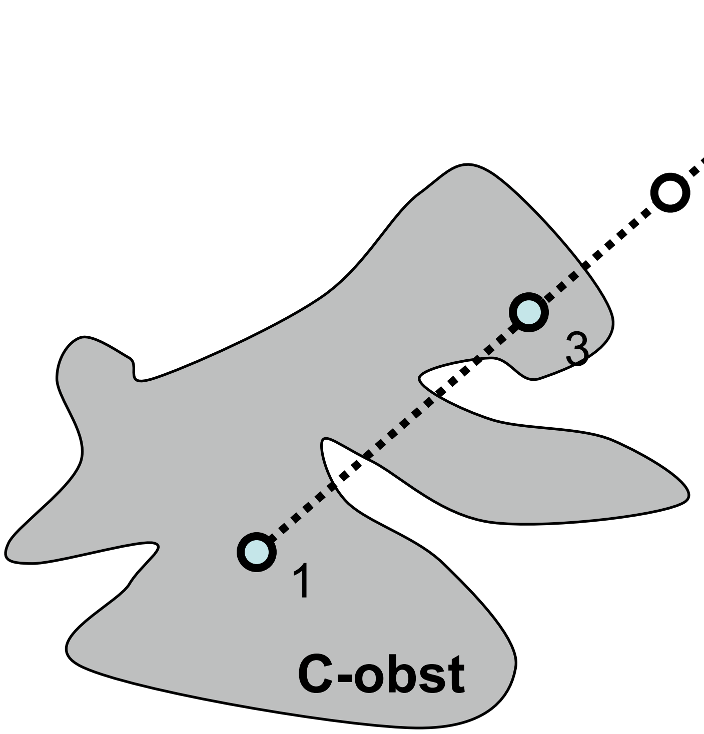


Basic Idea (for workspace obstacle S)

1. Find a point in S's C-obstacle
   (robot placement colliding with S)
2. Select a random direction in C-space
3. Find a free point in that direction
4. Find boundary point between them
   using binary search (collision checks)

Note: we can use more sophisticated heuristics to try to cover C-obstacle

Finding Points on C-obstacles

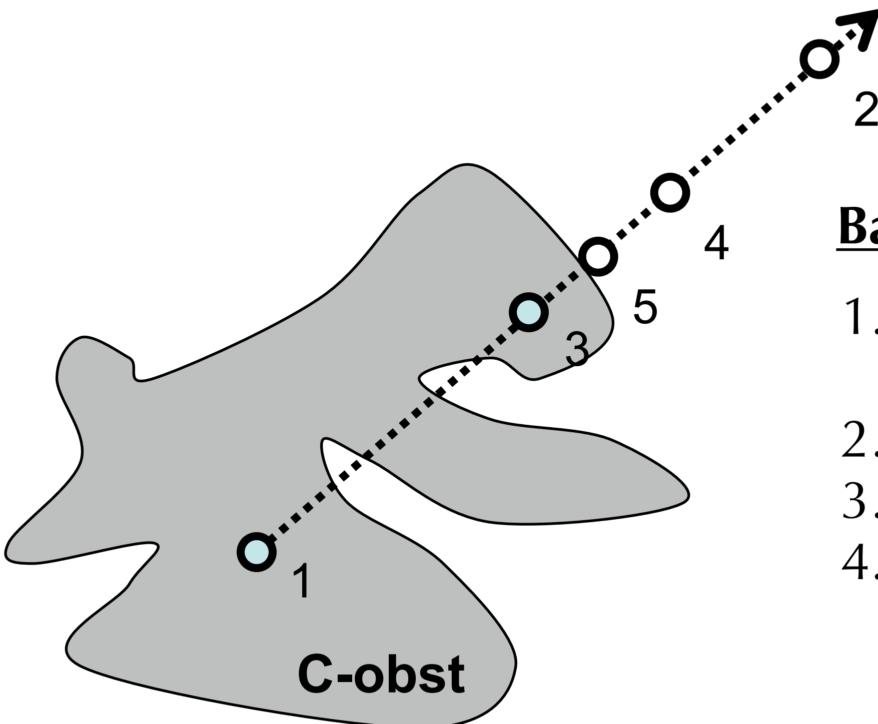


Basic Idea (for workspace obstacle S)

1. Find a point in S's C-obstacle (robot placement colliding with S)
2. Select a random direction in C-space
3. Find a free point in that direction
4. Find boundary point between them using binary search (collision checks)

Note: we can use more sophisticated heuristics to try to cover C-obstacle

Finding Points on C-obstacles

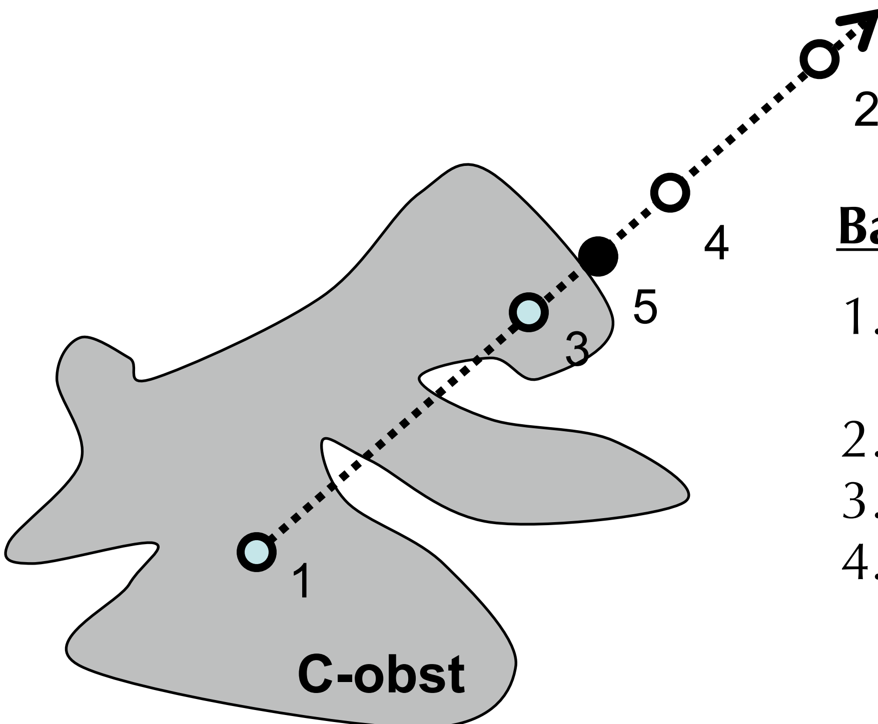


Basic Idea (for workspace obstacle S)

1. Find a point in S's C-obstacle (robot placement colliding with S)
2. Select a random direction in C-space
3. Find a free point in that direction
4. Find boundary point between them using binary search (collision checks)

Note: we can use more sophisticated heuristics to try to cover C-obstacle

Finding Points on C-obstacles

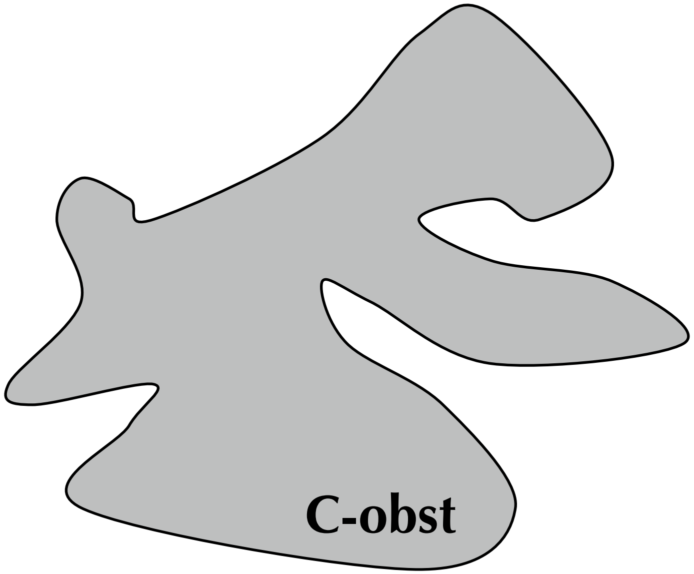


Basic Idea (for workspace obstacle S)

1. Find a point in S's C-obstacle (robot placement colliding with S)
2. Select a random direction in C-space
3. Find a free point in that direction
4. Find boundary point between them using binary search (collision checks)

Note: we can use more sophisticated heuristics to try to cover C-obstacle

Gaussian Sampling PRM

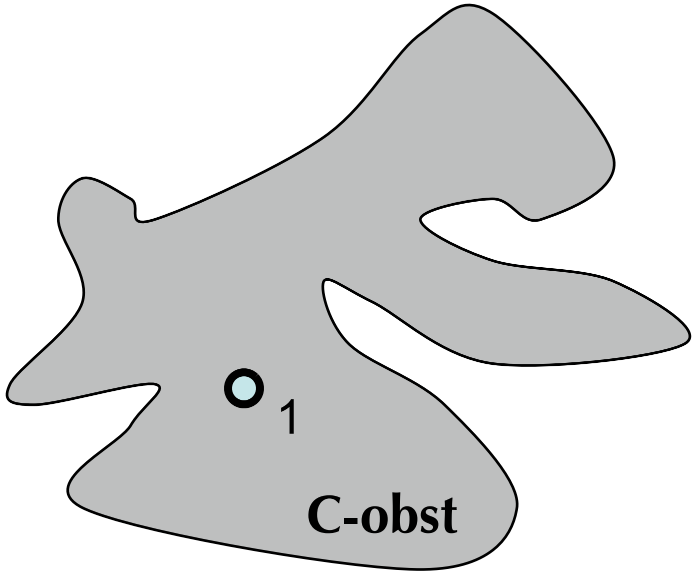


1. Find a point in S's C-obstacle (robot placement colliding with S)

2. Find another point that is within distance *d* to the first point, where d is a random variable in a *Gaussian distribution*

3. Keep the second point if it is collision free

Gaussian Sampling PRM

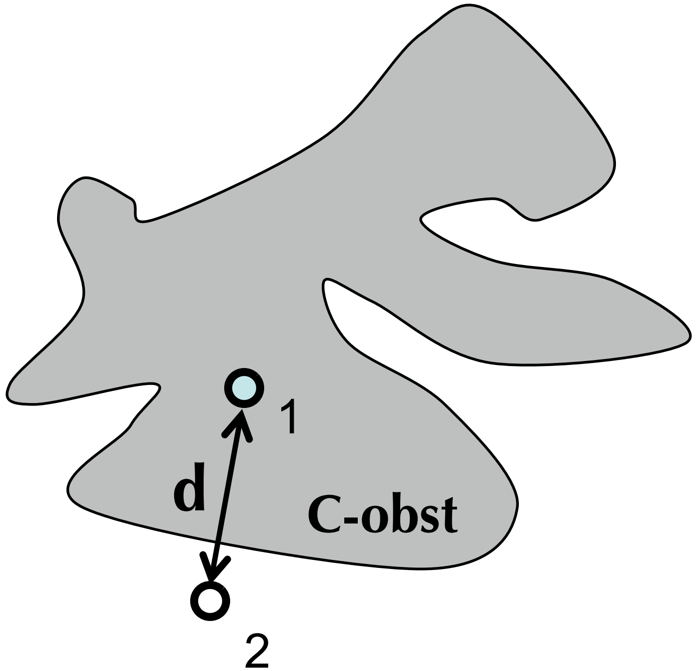


1. Find a point in S's C-obstacle
   (robot placement colliding with S)

2. Find another point that is within distance *d* to the first point, where d is a random variable in a *Gaussian distribution*

3. Keep the second point if it is collision free

Gaussian Sampling PRM

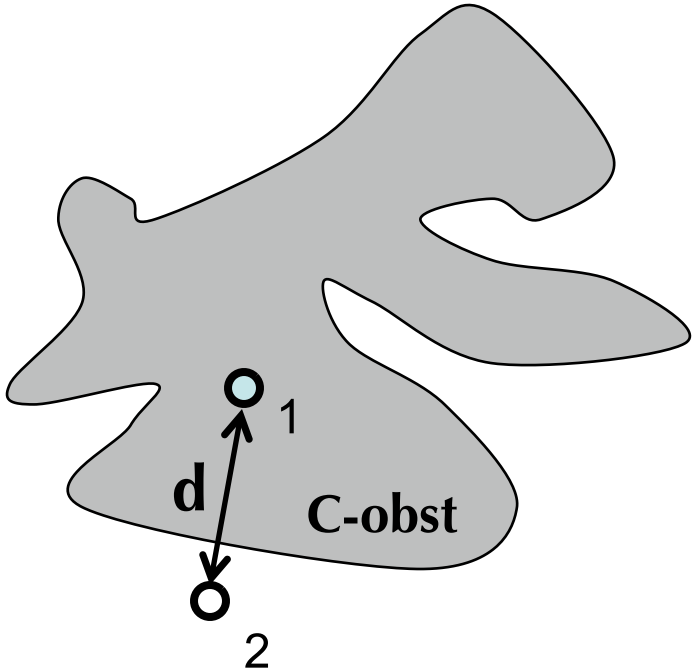


1. Find a point in S's C-obstacle
   (robot placement colliding with S)

2. Find another point that is within distance *d* to the first point, where d is a random variable in a *Gaussian distribution*

3. Keep the second point if it is collision free

Gaussian Sampling PRM



1. Find a point in S's C-obstacle (robot placement colliding with S)

2. Find another point that is within distance *d* to the first point, where d is a random variable in a *Gaussian distribution*

3. Keep the second point if it is collision free

Note

- Two paradigms: (1) OBPRM: Fix the samples (2) Gaussian PRM: Filter the samples
- None of these methods can (be proved to) provide guarantee that the samples in the narrow passage will increase!

Related Work (selected)

- **Probabilistic Roadmap Methods**
  - Uniform Sampling (original) [Kavraki, Latombe, Overmars, Svestka, 92, 94, 96]
  - Obstacle-based PRM (OBPRM) [Amato et al, 98]
  - PRM Roadmaps in Dilated Free space [Hsu et al, 98]
  - Gaussian Sampling PRMs [Boor/Overmars/van der Steppen 99]
  - Bridge test [Hsu et al 03]
  - Visibility Roadmaps [Laumond et al 99]
  - Using Medial Axis [Kavraki et al 99, Lien/Thomas/Wilmarth/Amato/Stiller 99, 03, Lin et al 00]
  - Generating Contact Configurations [Xiao et al 99]
  - Using workspace clues

Probabilistic Methods

- Avoid computing C-obstacles
  - Too difficult to compute
- Sacrifice completeness to gain simplicity and efficiency - probabilistic complete!
- Probabilistic Methods
  - Graph based
  - Tree based - single-shot planners!

Rapidly-Exploring Random Tree (RRT)

- RRTs: Rapidly-exploring Random Trees

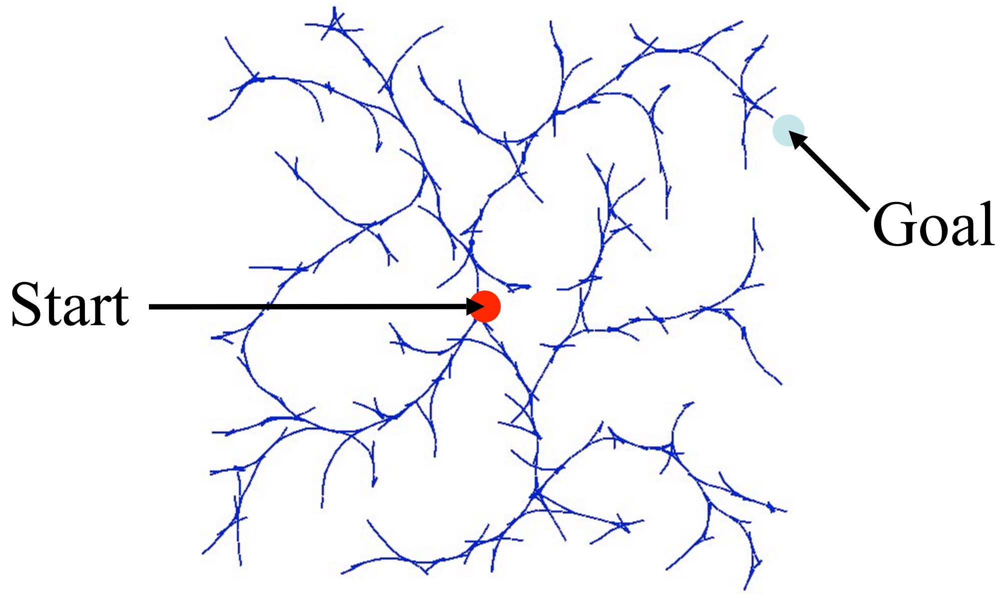
  Rapidly-exploring random trees: Progress and prospects. S. M. LaValle and J. J. Kuffner. In *Proceedings Workshop on the Algorithmic Foundations of Robotics*, 2000.)

  – Incrementally builds the roadmap tree

- Extends to more advanced planning techniques
  – Integrates the control inputs to ensure that the kinodynamic constraints are satisfied

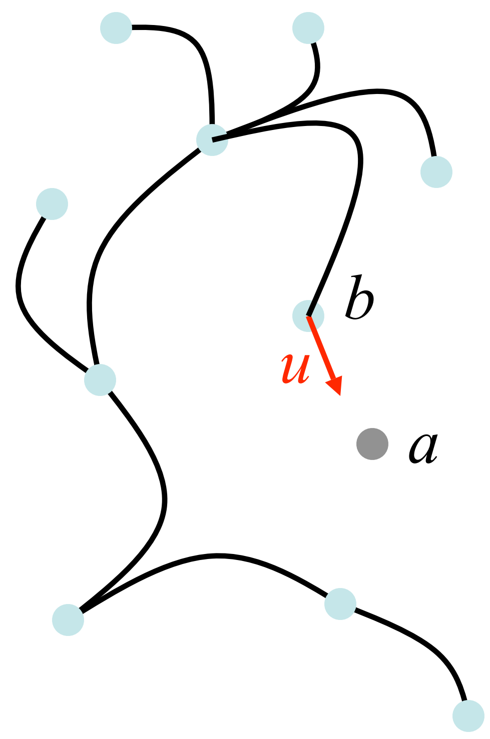
How it Works

- Build a rapidly-exploring random tree in state space (X), starting at s_{start}
- Stop when tree gets sufficiently close to s_{goal}



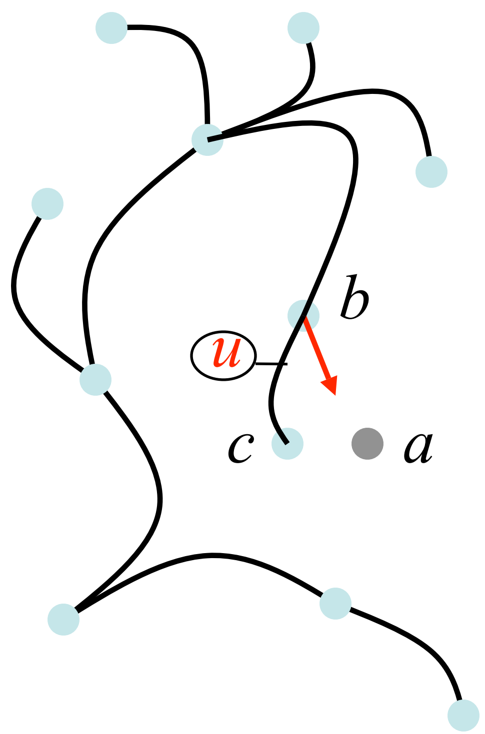
Building an RRT

- To extend an RRT:
  - Pick a random point *a* in *X*
  - Find *b*, the node of the tree closest to *a*
  - Find control inputs *u* to steer the robot from *b* to *a*



Building an RRT

- To extend an RRT (cont.)
  - Apply control inputs *u* for time δ, so robot reaches *c*
  - If no collisions occur in getting from *a* to *c*, add *c* to RRT and record *u* with new edge



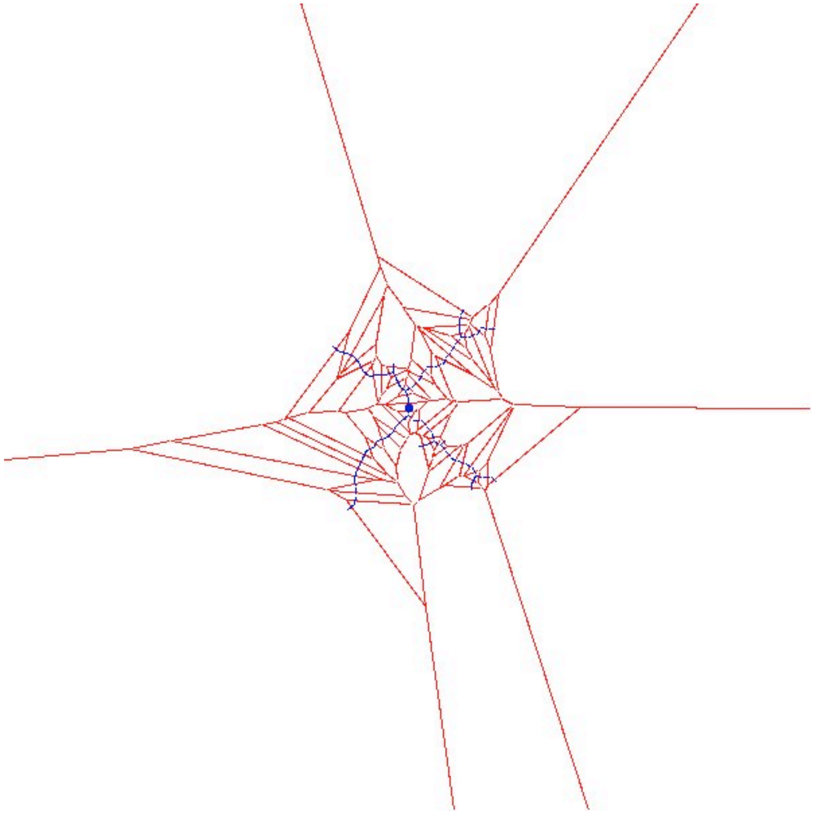
Executing the Path

Once the RRT reaches s_{goal}

- Backtrack along tree to identify edges that lead from s_{start} to s_{goal}
- Drive robot using control inputs stored along edges in the tree

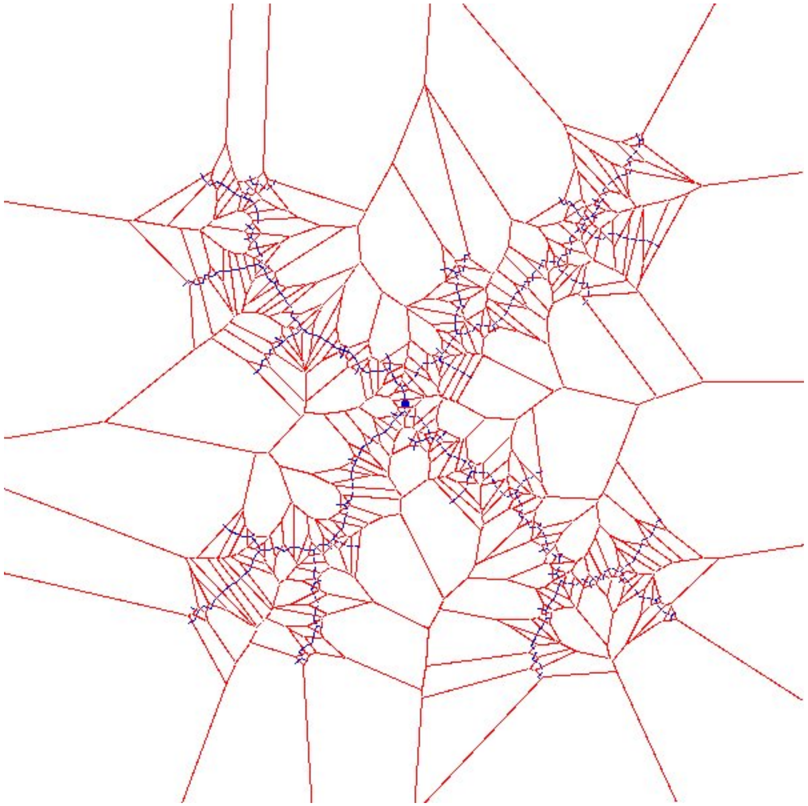
Principle Advantage

- RRT quickly explores the state space:
  - Nodes most likely to be expanded are those with largest Voronoi regions

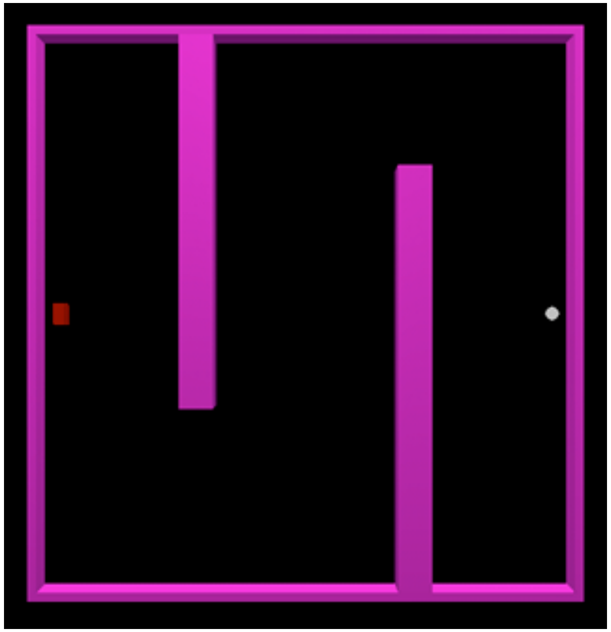


Principle Advantage

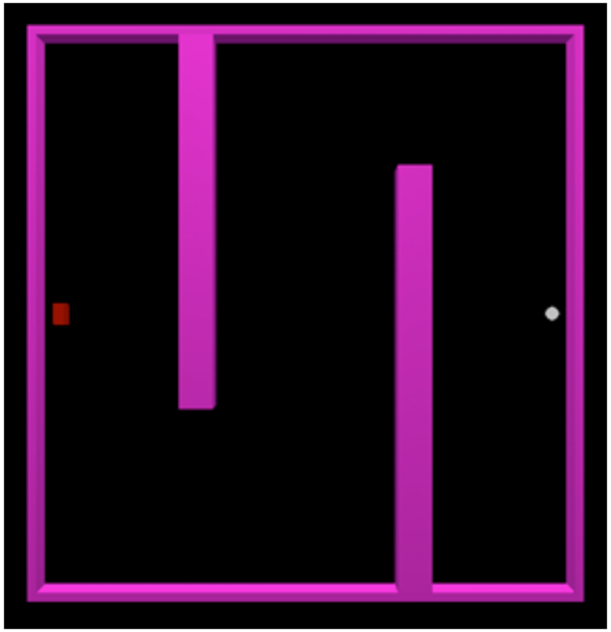
- RRT quickly explores the state space:
  - Nodes most likely to be expanded are those with largest Voronoi regions

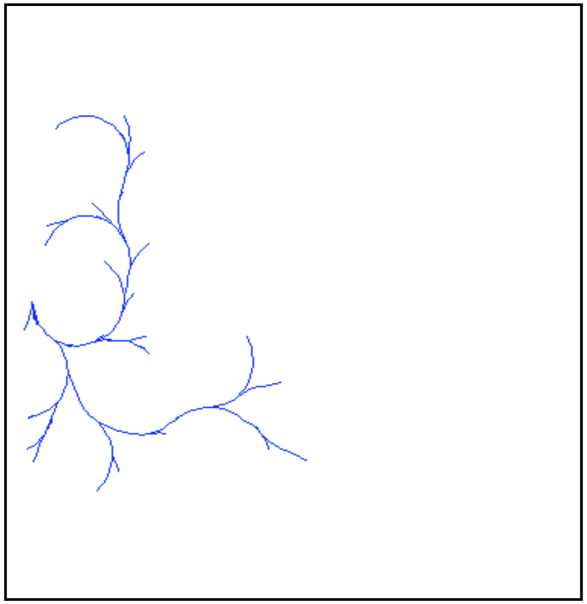


Problem of Simple RRT Planner

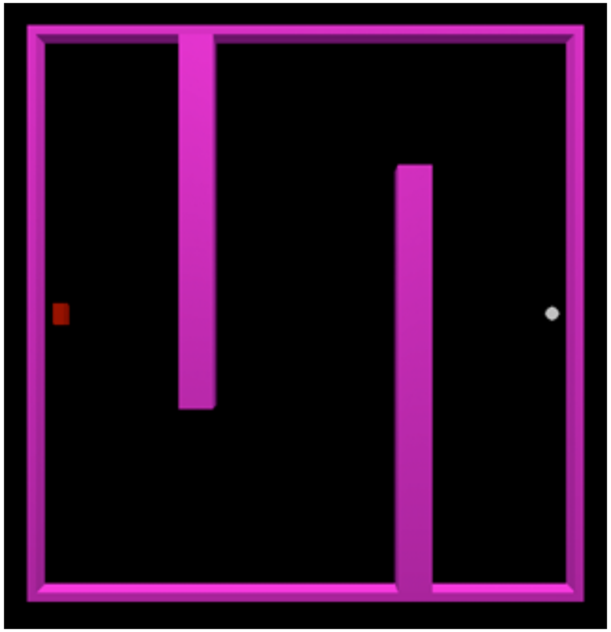


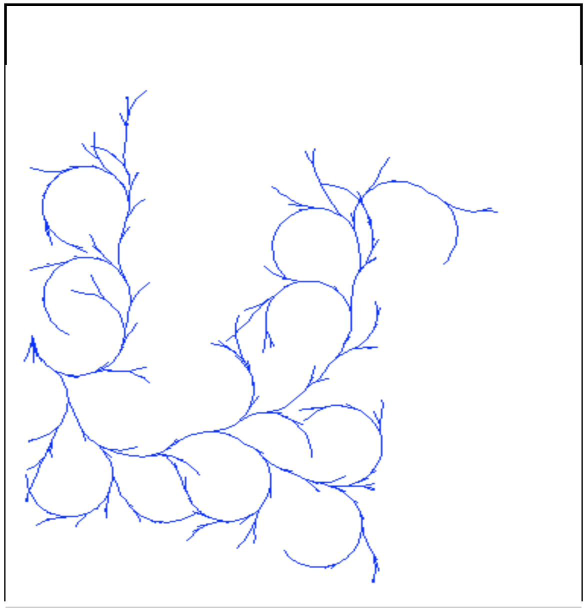
Problem of Simple RRT Planner



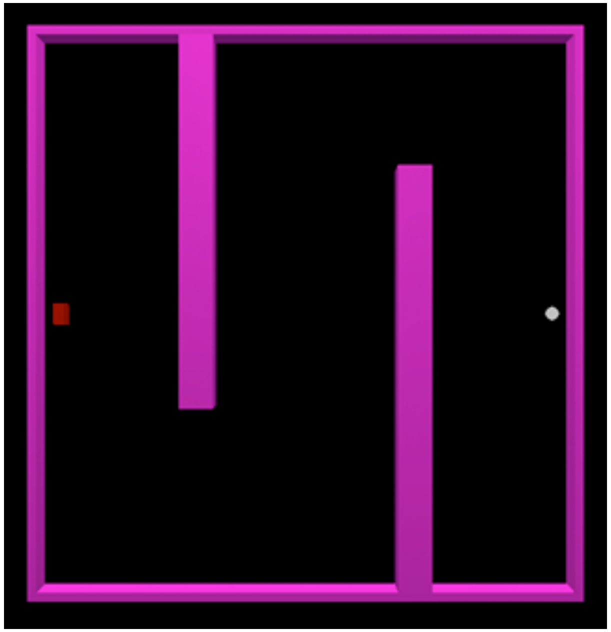


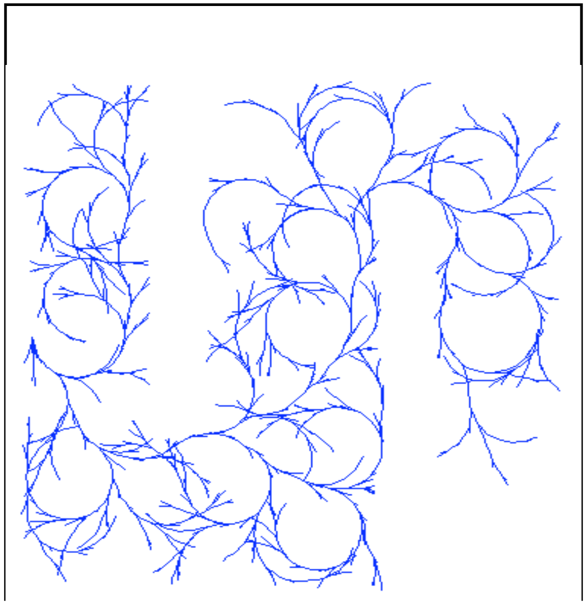
Problem of Simple RRT Planner



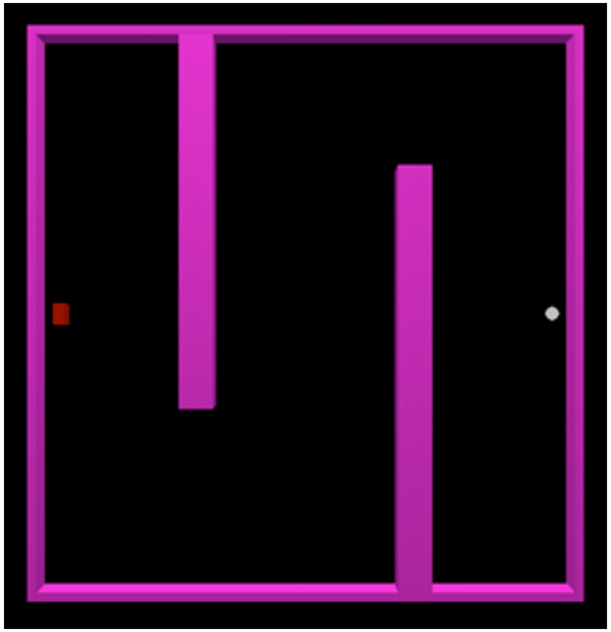


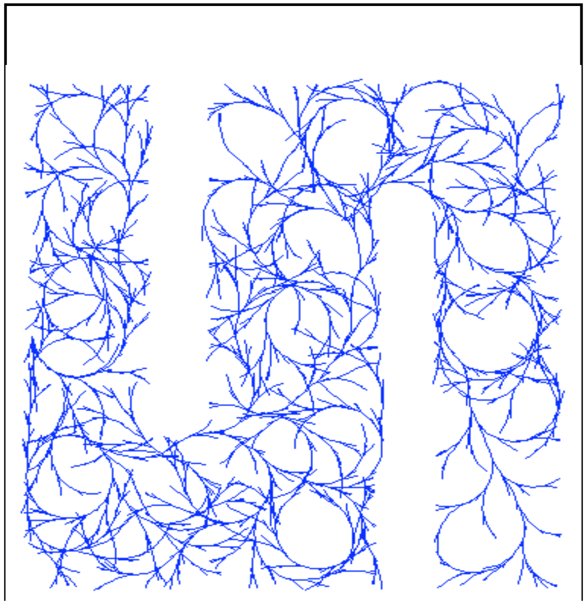
Problem of Simple RRT Planner



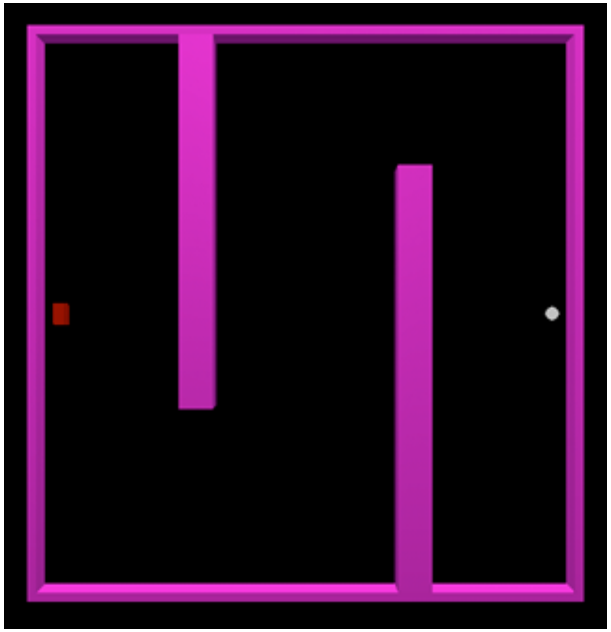


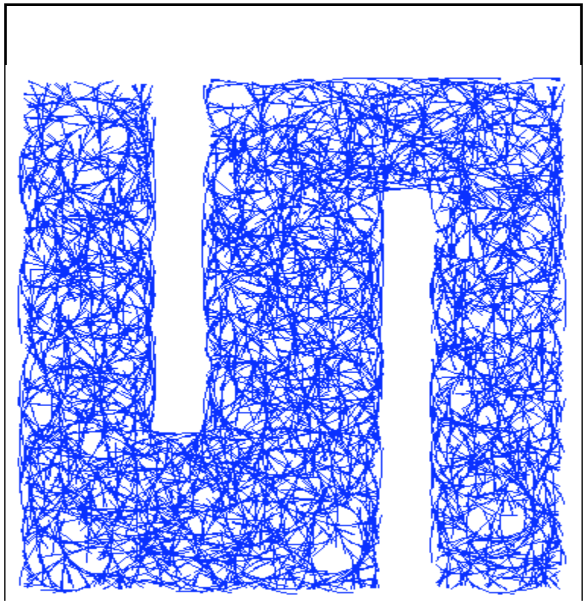
Problem of Simple RRT Planner



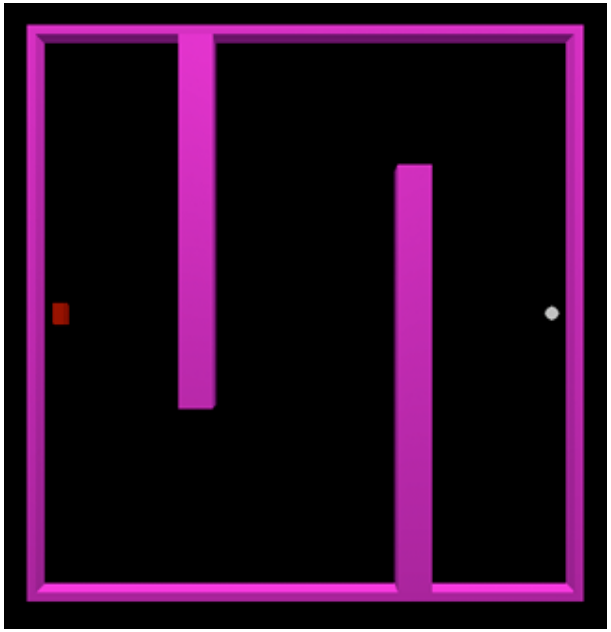


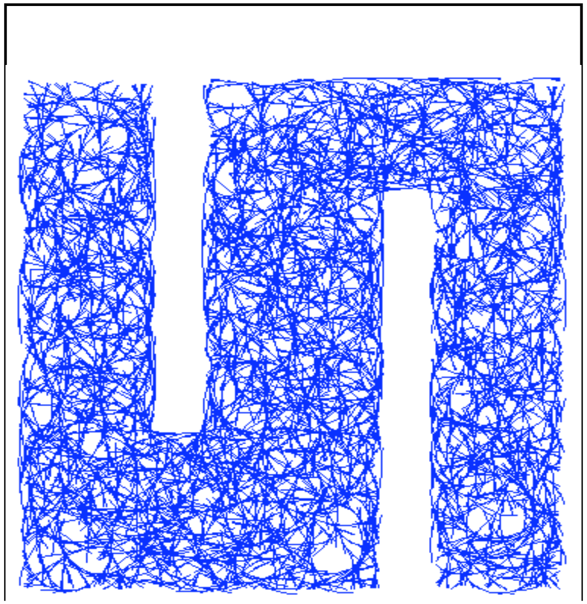
Problem of Simple RRT Planner





Problem of Simple RRT Planner

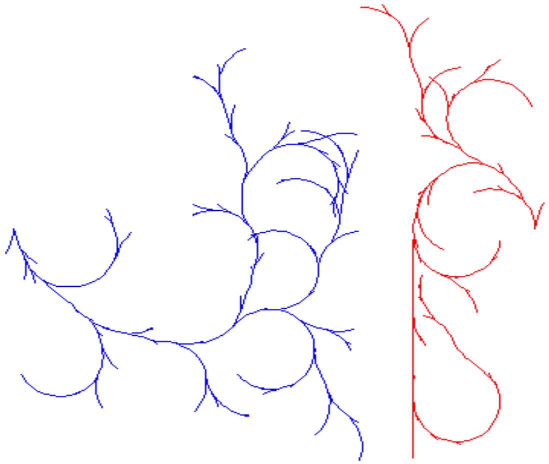




- Problem: ordinary RRT explores *X* uniformly
  - → slow convergence
- Solution: bias distribution towards the goal

Bidirectional Planners

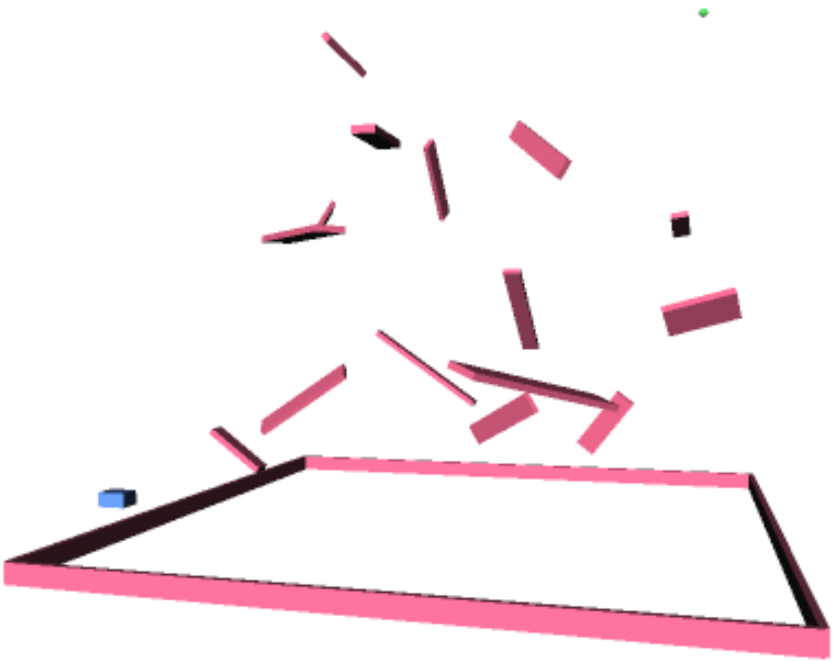
- Build two RRTs, from start and goal state

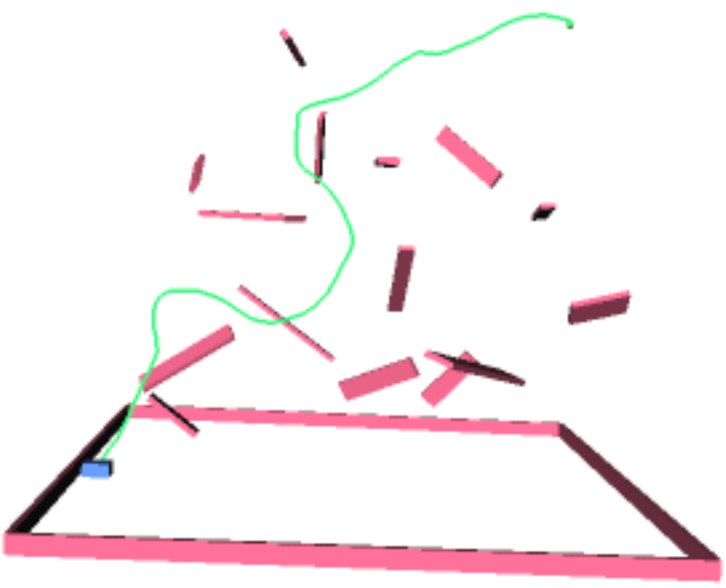


- Complication: need to connect two RRTs
  - local planner will not work (dynamic constraints)
  - **bias** the distribution, so that the trees meet

Bidirectional RRT Example

Bidirectional RRT Example

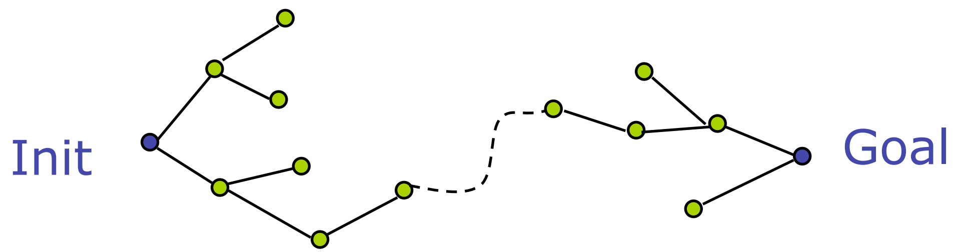




Path Planning in Expansive Configuration Spaces, D. Hsu, J.C. Latombe, & R. Motwani, 1999.

Expansion Space Tree (EST)

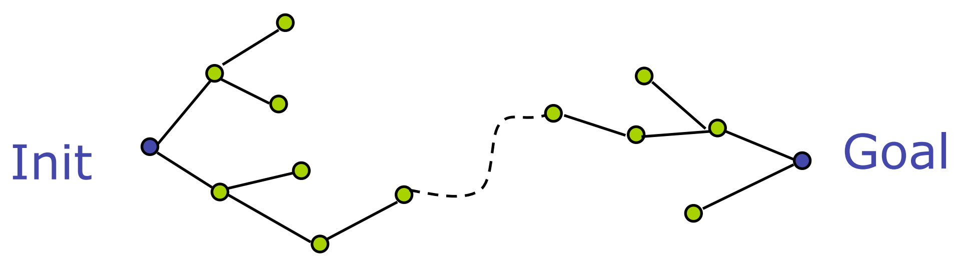
1. Grow two trees from Init position and Goal configurations.
2. Randomly sample nodes around existing nodes.
3. Connect a node in the tree rooted at Init to a node in the tree rooted at the Goal.



Path Planning in Expansive Configuration Spaces, D. Hsu, J.C. Latombe, & R. Motwani, 1999.

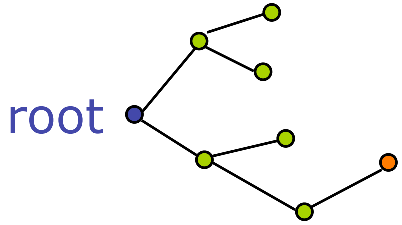
Expansion Space Tree (EST)

1. Grow two trees from Init position and Goal configurations.
2. Randomly sample nodes around existing nodes.
3. Connect a node in the tree rooted at Init to a node in the tree rooted at the Goal.



Expansion + Connection

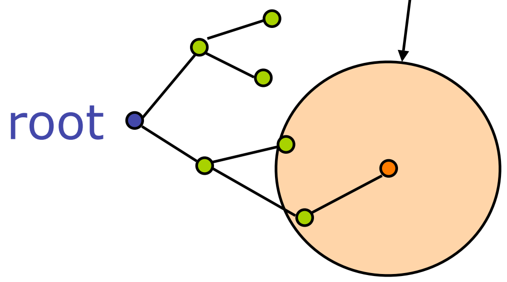
Expansion



Expansion

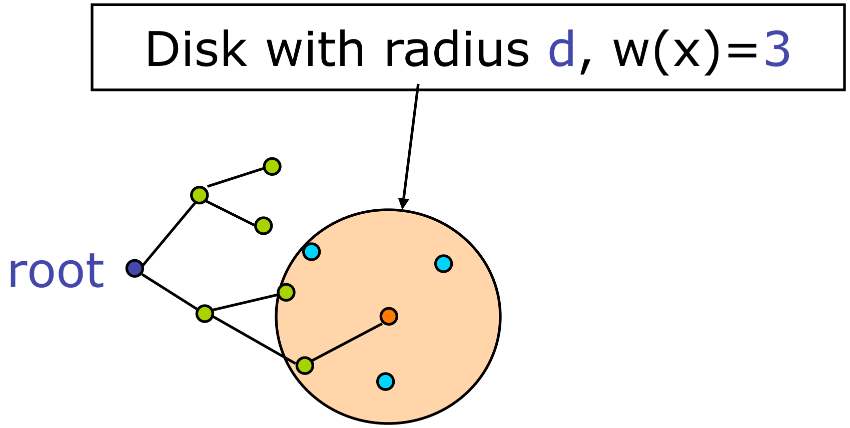
1. Pick a node x with probability 1/w(x).

Disk with radius d, w(x)=3



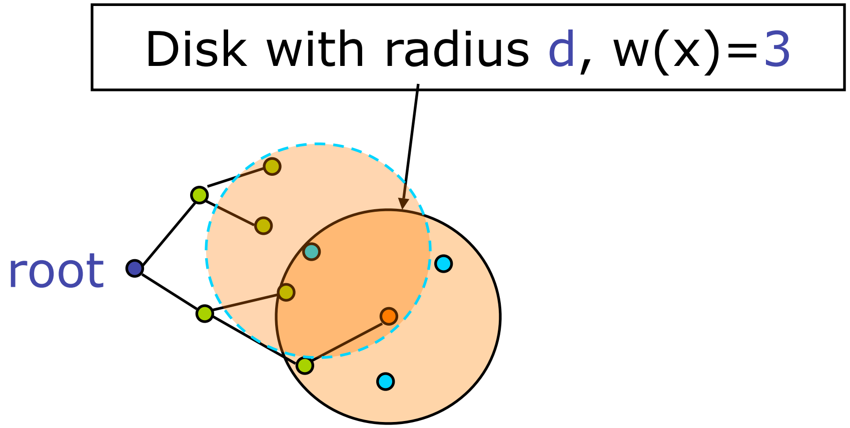
Expansion

1. Pick a node x with probability 1/w(x).
2. Randomly sample k points around x.



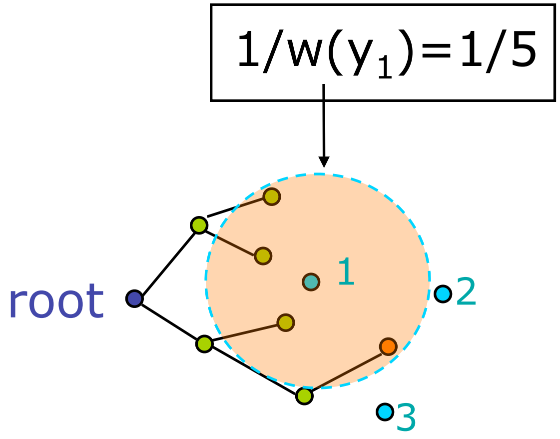
Expansion

1. Pick a node x with probability 1/w(x).
2. Randomly sample k points around x.
3. For each sample y, calculate w(y), which gives probability 1/w(y).



Expansion

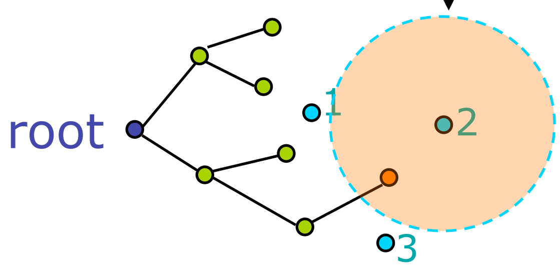
1. Pick a node x with probability 1/w(x).
2. Randomly sample k points around x.
3. For each sample y, calculate w(y), which gives probability 1/w(y).



Expansion

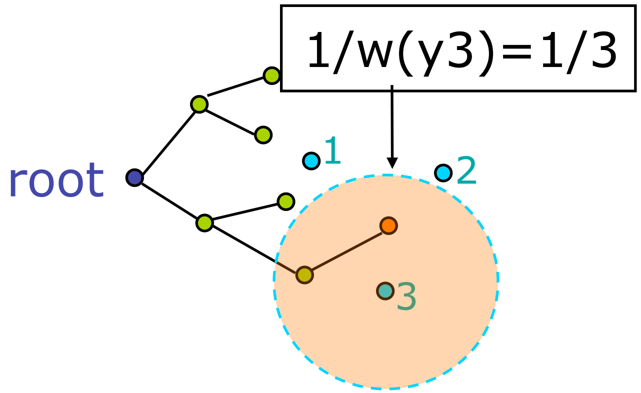
1. Pick a node x with probability 1/w(x).
2. Randomly sample k points around x.
3. For each sample y, calculate w(y), which gives probability 1/w(y).

$1/w(y_2)=1/2$



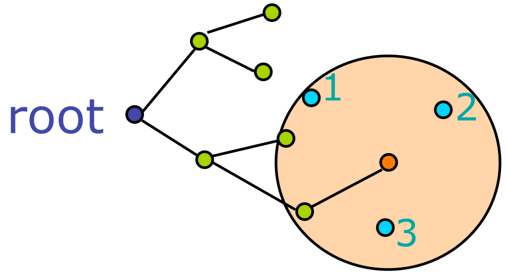
Expansion

1. Pick a node x with probability 1/w(x).
2. Randomly sample k points around x.
3. For each sample y, calculate w(y), which gives probability 1/w(y).



Expansion

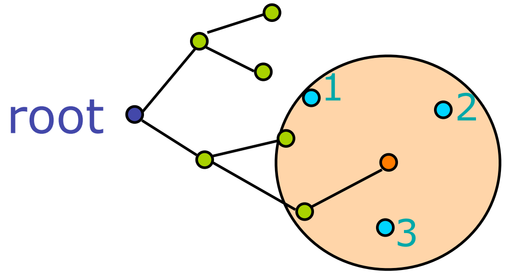
1. Pick a node x with probability 1/w(x).
2. Randomly sample k points around x.
3. For each sample y, calculate w(y), which gives probability 1/w(y). If y



Expansion

1. Pick a node x with probability 1/w(x).
2. Randomly sample k points around x.
3. For each sample y, calculate w(y), which gives probability 1/w(y). If y

   (a) has higher probability; (b) collision free; (c) can sees x

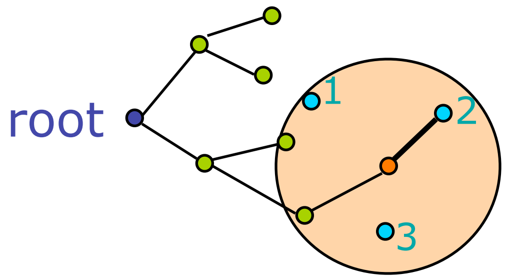


Expansion

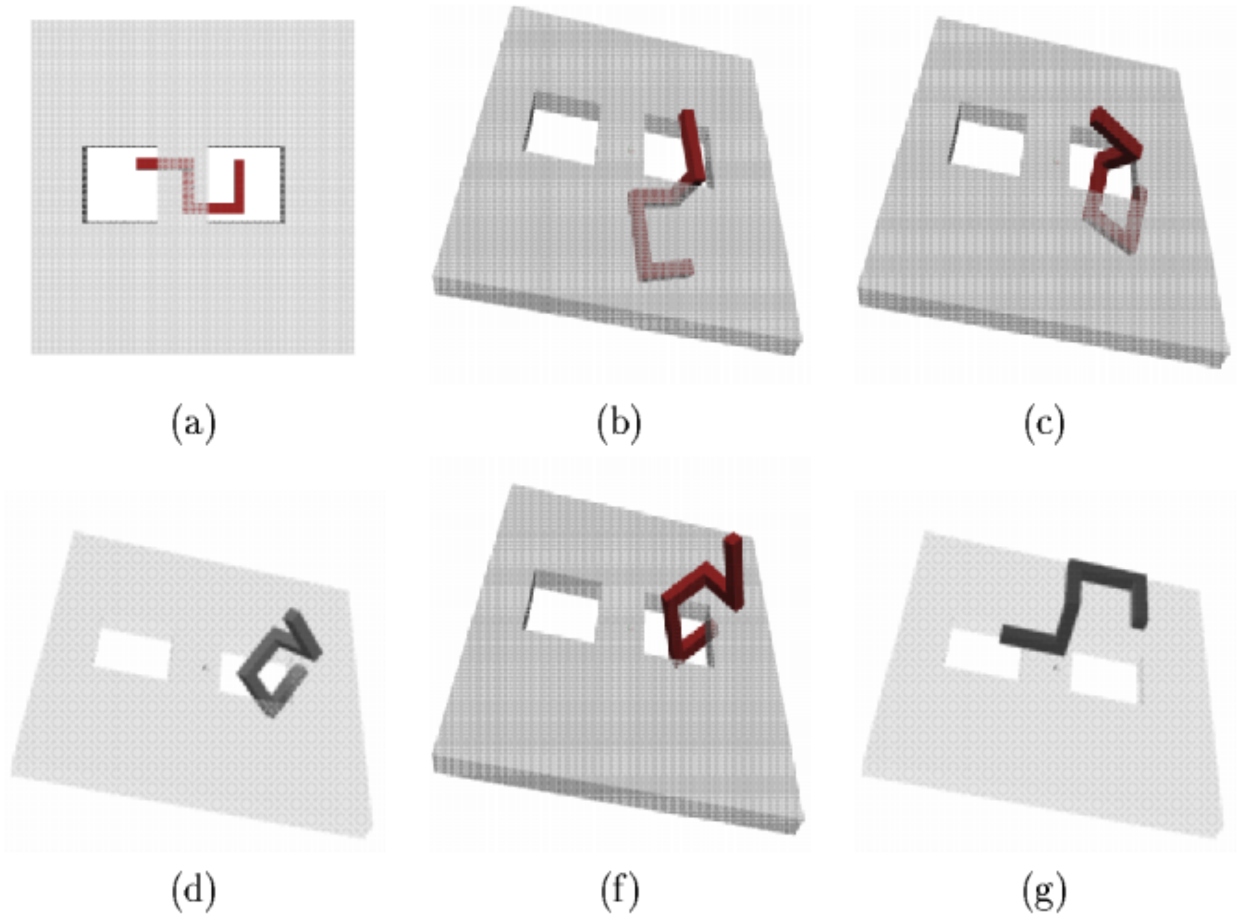
1. Pick a node x with probability 1/w(x).
2. Randomly sample k points around x.
3. For each sample y, calculate w(y), which gives probability 1/w(y). If y

   (a) has higher probability; (b) collision free; (c) can sees x

   then add y into the tree.



Computed example



More Problems with Probabilistic Methods

- ??

Conclusion

- Motion planning is difficult (intractable)

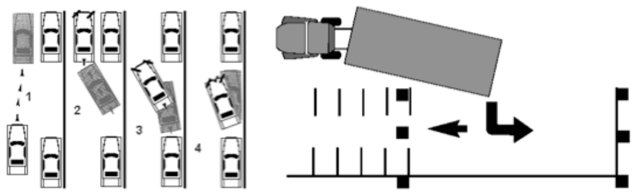
- Roadmap methods
  - Probabilistic Motion Planners

What is not covered?

- C-space
  - Minkowski sum computation (end of the semester)
- Deterministic Roadmap methods
  - Visibility graph, cell, decomposition,…
  - Algorithms of visibility graph, trapezoidation
  - Schwartz and Sharir's critical curve method
  - Canny's Silhouette methods
  - Voronoi diagram computation
- Probabilistic Roadmap methods
  - Analysis of PRM, RRT, EST, OBPRM, MAPRM…

What is not covered?

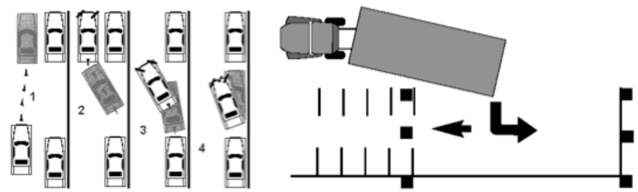
- Other types of motion planning
  - With constraints
    - Close-chain constraint
    - Nonholonomic constraint
    - Differential constraints
  - Manipulate planning
  - Assembly planning
  - Planning with uncertainty
  - Planning for multiple robots, dynamic env
  - Planning for highly articulated objects
  - Planning for deformable objects
  - …



Little Seiko

What is not covered?

- Other types of motion planning
  - With constraints
    - Close-chain constraint
    - Nonholonomic constraint
    - Differential constraints
  - Manipulate planning
  - Assembly planning
  - Planning with uncertainty
  - Planning for multiple robots, dynamic env
  - Planning for highly articulated objects
  - Planning for deformable objects
  - …



These will require another semester…

Little Seiko

Homework Assignment

- TBD

Programming Assignment

- Programming assignment #2 will be given out soon (by the end of this week)
  - Probabilistic motion planning implementation (in C/C++)
  - Implement at least two planning strategies
    - Papers is posted on the discussion board
  - Design an interesting motion planning problem

Additional Readings

- **Gross motion planning—a survey**, Y. K. Hwang and N. Ahuja, ACM Computing Surveys, 1992 (survey paper)
- **Robot Motion Planning**. J.C. Latombe. Kluwer Academic Publishers, Boston, MA, 1991.
- **Motion Planning: A Journey of Robots**, **Molecules, Digital Actors, and Other Artifacts**. Jean-Claude Latombe, IJRR, 1999 (survey paper)
- **Planning Algorithms**, Steven LaValle, 2006, Cambridge University Pres, (Free download at http://planning.cs.uiuc.edu/)