Algorithms for Sensor-Based Robotics: Sampling-Based Motion Planning

Computer Science 336

http://www.cs.jhu.edu/~hager/Teaching/cs336

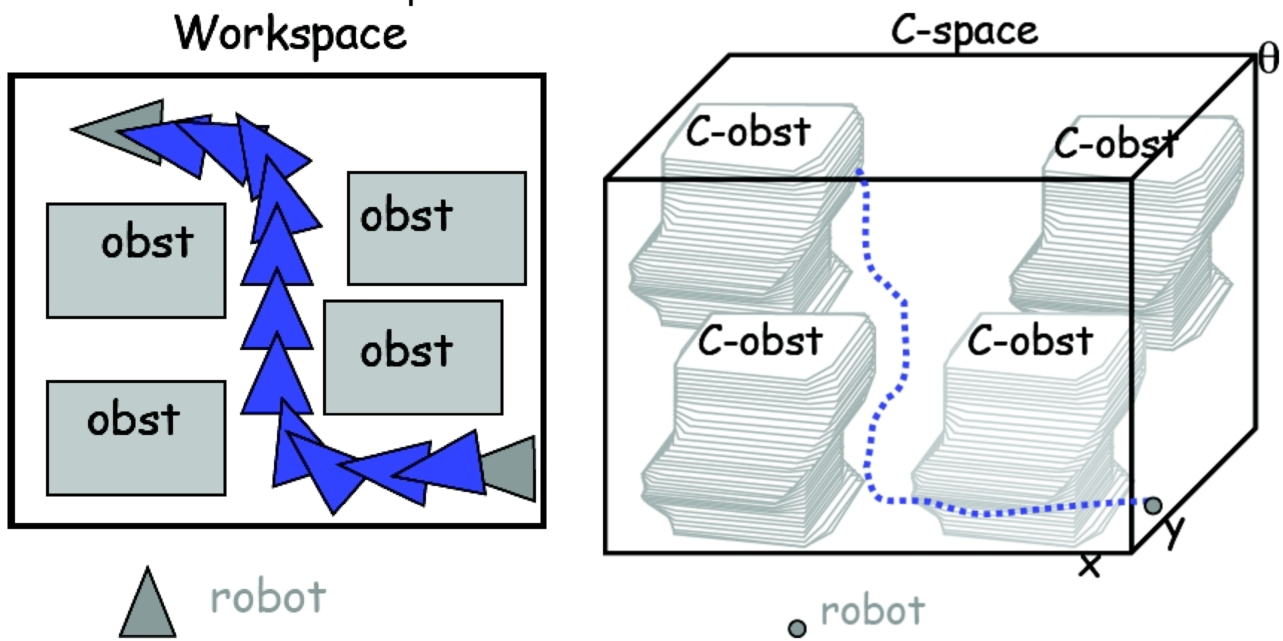
Professor Hager

http://www.cs.jhu.edu/~hager

Recall Earlier Methods

From Workspace to Configuration Space

- simple workspace obstacle transformed into complex configuration-space obstacle
- robot transformed into point in configuration space
- path transformed from swept volume to 1d curve



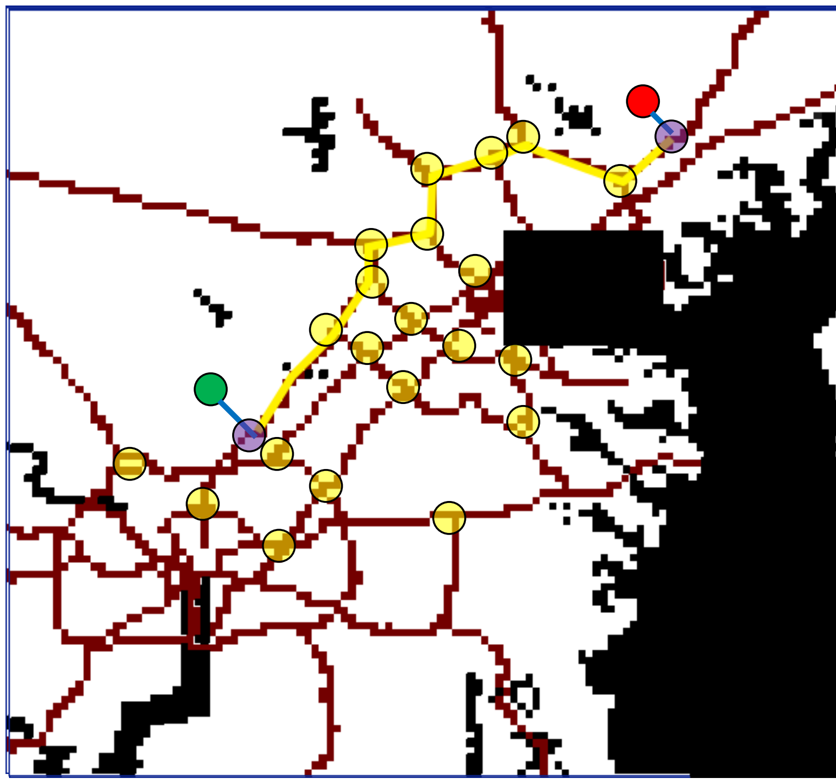
[fig from Jyh-Ming Lien]

Explicit Construction of Configuration Space/Roadmaps

- PSPACE-complete
- Exponential dependency on dimension
- No practical algorithms

The Basic Idea

- Capture the connectivity of Q_{free} by a graph or network of paths.



). Hager who loosely
ato

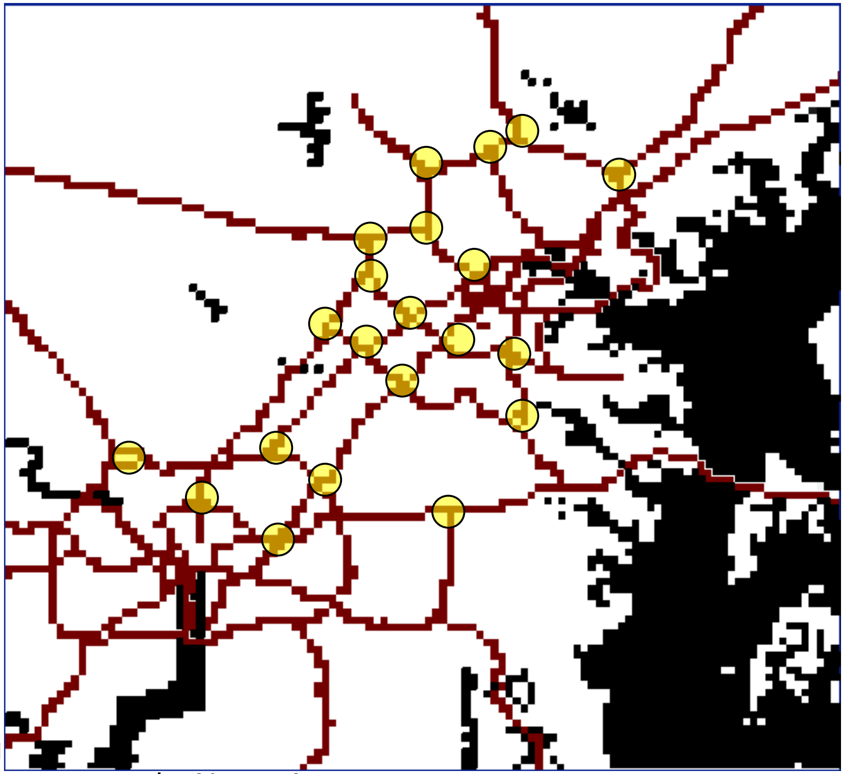
RoadMap Definition

- A roadmap, RM, is a set of ***trajectories*** (i.e. $f(t, \boldsymbol{q}_A, \boldsymbol{q}_B)$) such that for all $\boldsymbol{q}_{\text{start}} \in Q_{\text{free}}$ and $\boldsymbol{q}_{\text{goal}} \in Q_{\text{free}}$ can be connected by a path:

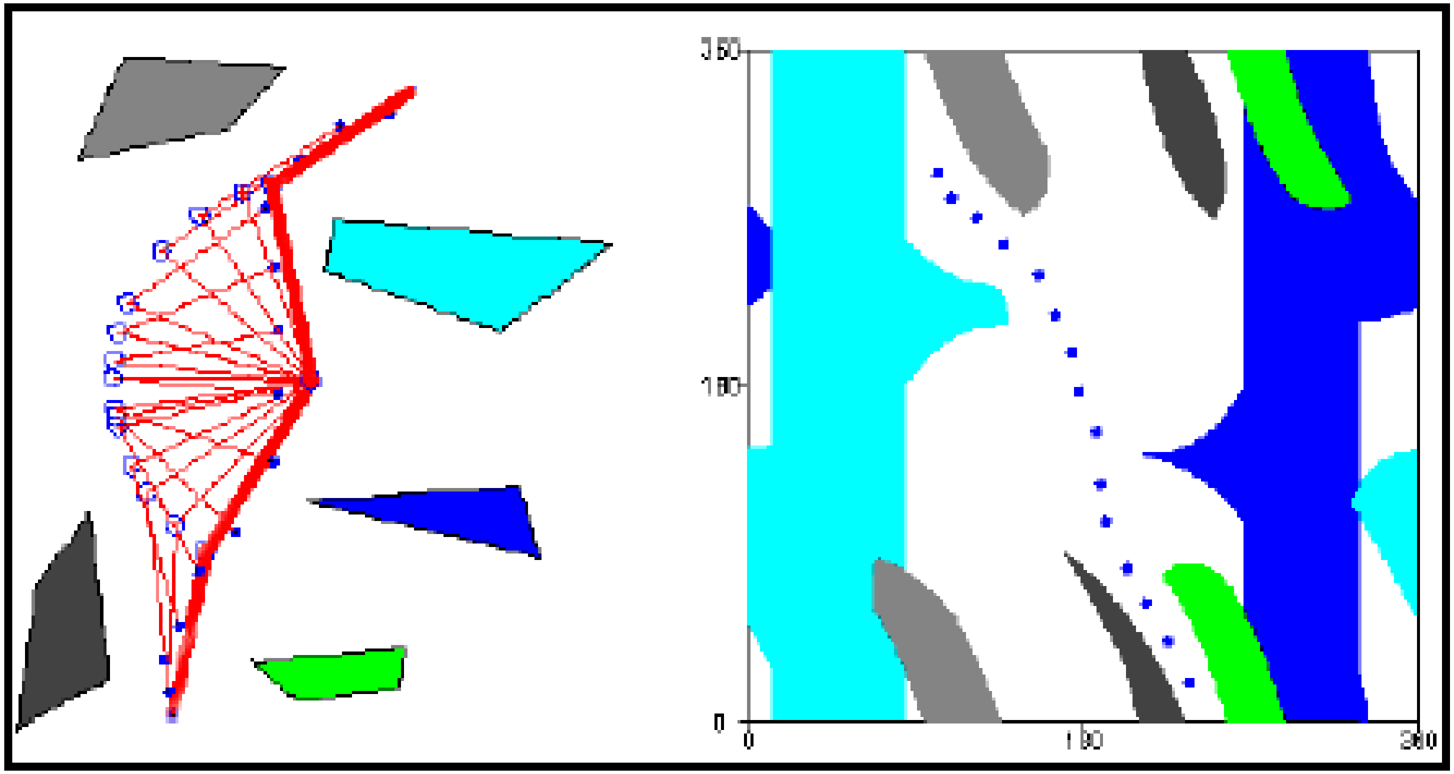
- The three ingredients of a roadmap
  - **Accessibility:** There is a path from $\boldsymbol{q}_{\text{start}} \in Q_{\text{free}}$ to some $\boldsymbol{q}' \in$ RM
  - **Departability:** There is a path from some $\boldsymbol{q}'' \in$ RM to $q_{\text{goal}} \in Q_{\text{free}}$
  - **Connectivity:** there exists a path in RM between q' and q''

RoadMap Path Planning

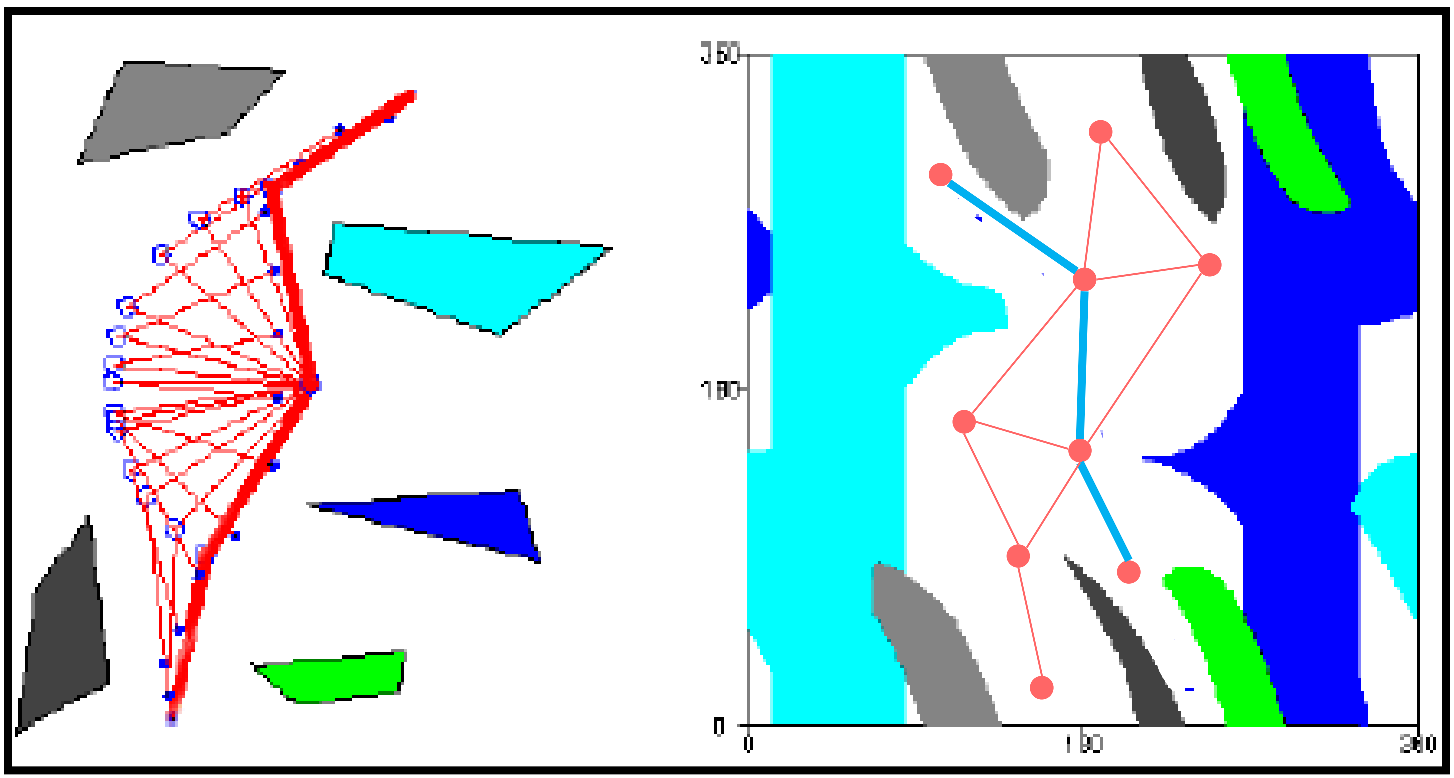
1. Build the roadmap
   a) nodes are points in Q_{free} or its boundary
   b) two nodes are connected by an edge if there is a free path between them (i.e. $f\,(\,t, \boldsymbol{q}_A, \boldsymbol{q}_B\,)$)
2. Connect $\boldsymbol{q}_{\text{start}}$ and $\boldsymbol{q}_{\text{goal}}$ points to the road map at point q' and q'', respectively
3. Find a path on the roadmap between q' and q''. The result is a path in Q_{free} from start to goal



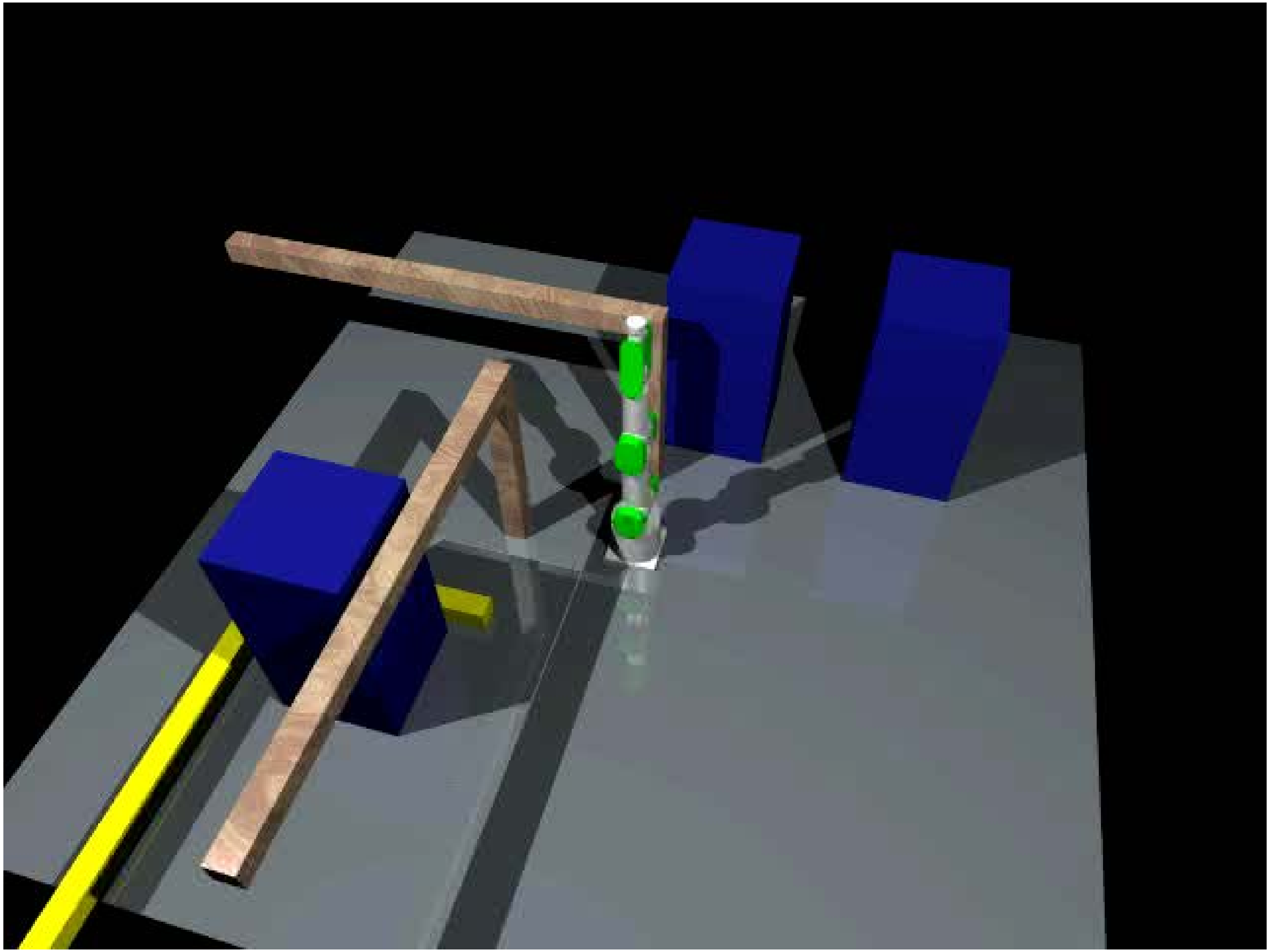
Roadmap



Roadmap



A Hard Problem



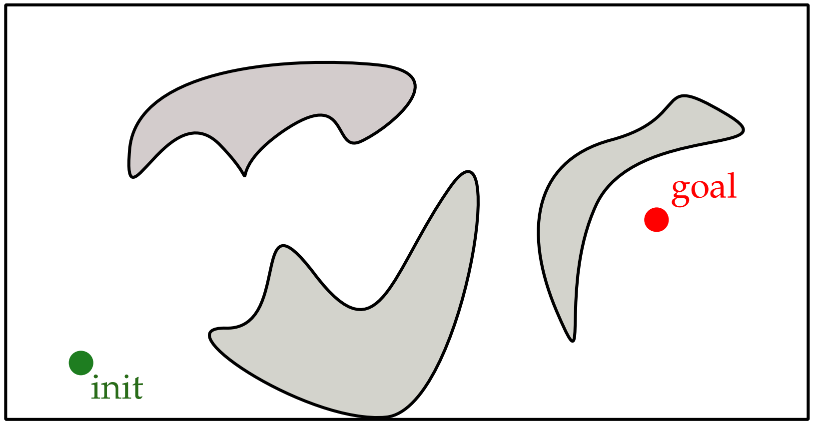
The Way Forward

- The hard part of path planning is to somehow represent the free configuration space
  - Recall the basic PSPACE-hard results of Reif and Canny.
  - Recall the limitations of potential fields and even navigation functions.
- Probabilistic RoadMap Planning (PRM) by Kavraki, 1996
  - samples to find free configurations
  - connects the configurations (creates a graph)
  - is designed to be a multi-query planner
- Expansive-Spaces Tree planner (EST) and Rapidly-exploring Random Tree planner (RRT)
  - are appropriate for single query problems

Remember the Basic Problem

- Robotic system: Single point
- Task: Compute collision-free path from initial to goal position

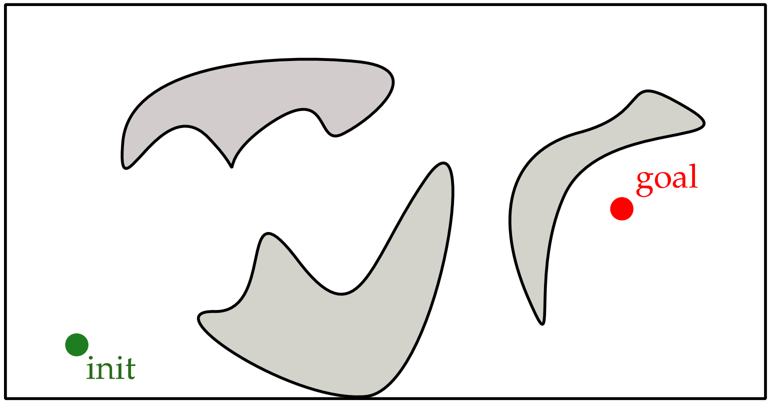
How would you solve it?

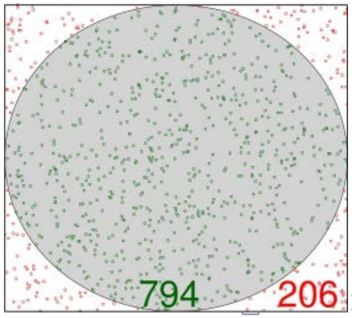


How about Rolling the Dice?

- Robotic system: Single point
- Task: Compute collision-free path from initial to goal position

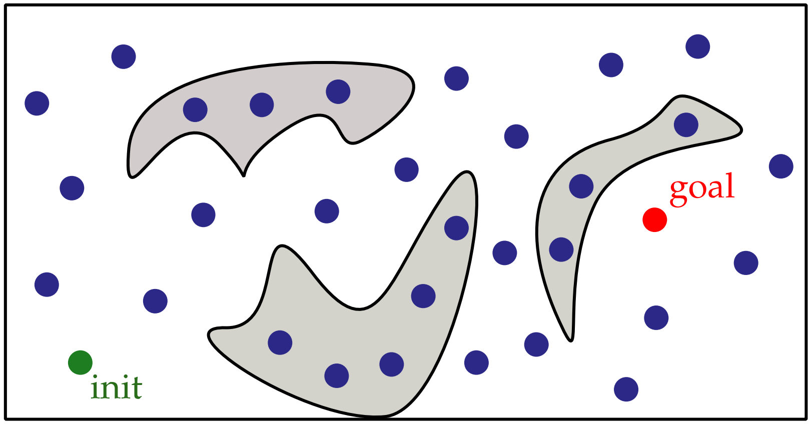
How would you solve it?





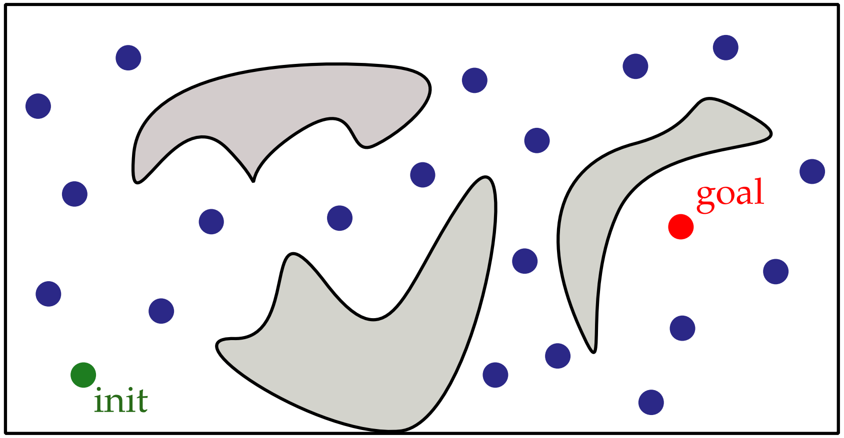
Sample

- Robotic system: Single point
- Task: Compute collision-free path from initial to goal position



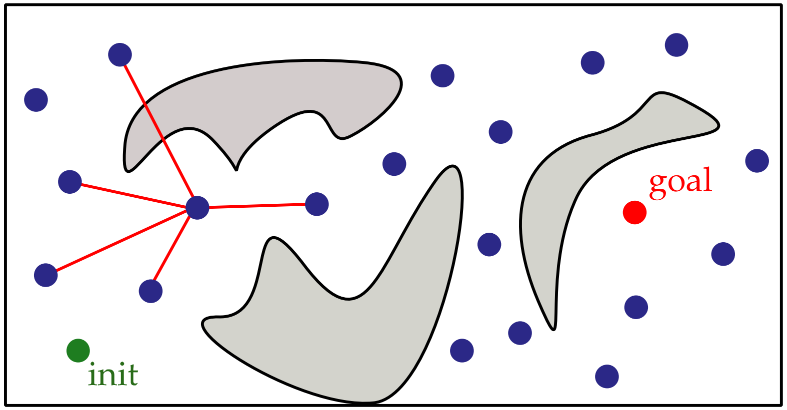
Discard

- Robotic system: Single point
- Task: Compute collision-free path from initial to goal position



Connect

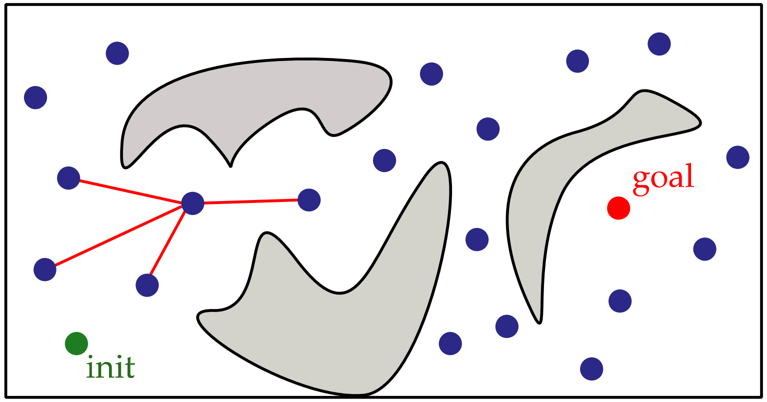
- Robotic system: Single point
- Task: Compute collision-free path from initial to goal position



amples via straight-line segments

Discard #2

- Robotic system: Single point
- Task: Compute collision-free path from initial to goal position

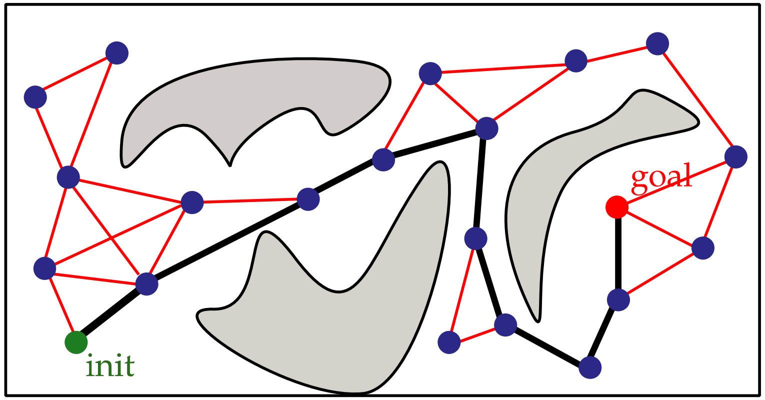


mples via straight-line segments

- Discard straight-line segments that are in collision

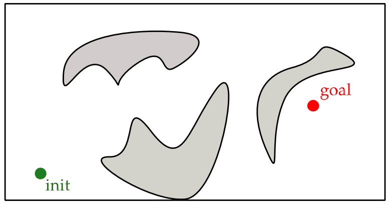
Finally, a Roadmap

- Robotic system: Single point
- Task: Compute collision-free path from initial to goal position



The Complete Algorithm

[Kavraki, Švestka, Latombe, Overmars 1996]



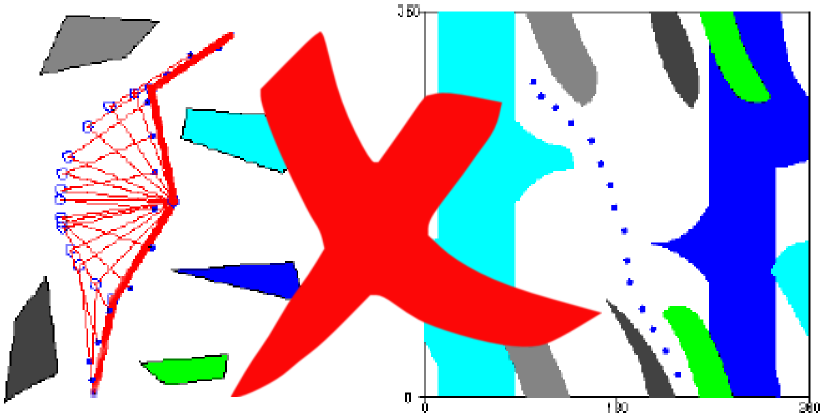
0. Initialization

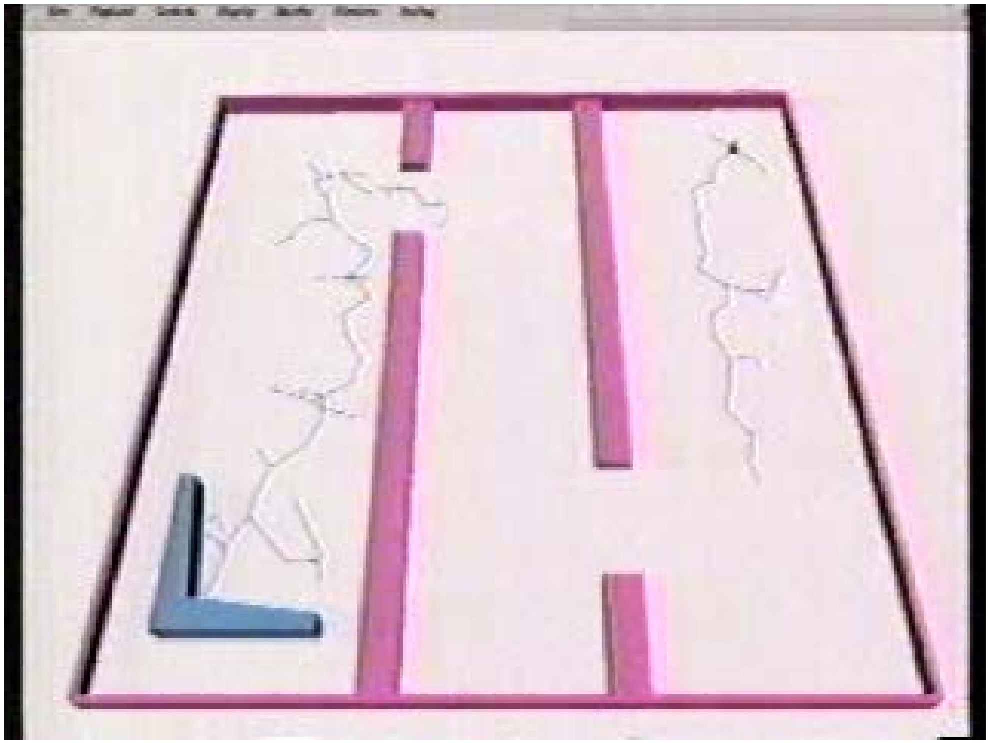
add q_{init} and q_{goal} to roadmap vertex set V

Why Are Probabilistic Methods Effective?

- Mainly, they do not try to construct the free configuration space
  - depend on local sampling of configurations (depends on efficient collision detection)
  - make use of relatively "dumb" planners to connect nodes

- They are often probabilistically complete
  - the probability of correct solution can be as high as desired with enough time





Some Minor Issues

- Ideally, the resulting graph will be connected
  - if it is not it might mean the space is disconnected
  - it might mean we didn't try hard enough (what is a hard case?)
- We haven't specified what the distance or path planner is
- To use for queries we must
  - connect the start and goal configurations to the roadmap (usually just treat like nodes and perform same algorithm)
  - perform a graph search on the resulting graph
  - if desired, smooth the resulting path a bit
- Interesting to note that the queries can be used to add more nodes to the graph

Sampling and Neighbors

- Uniform sampling of configurations
  - need to take care that rotations are “fairly” sampled

- Selecting closest neighbors: kd-tree
  - Given: a set S of n points in d-dimensional space
  - Recursively
    - choose a plane P that splits S about evenly (usually in a coordinate dimension)
    - store P at node
    - apply to children S_l and S_r
  - Requires O(dn) storage, built in O(dn log n) time
  - Query takes $O(n^{1-1/d} + m)$ time where m is # of neighbors
    - asymptotically linear in n and m with large d

- Selecting closest neighbors: cell-based method
  - when each point is generated, hash to a cell location

Distance Functions

- Distance between two configurations should reflect the likelihood that the planner will fail to find a path
  - close points, likely to succeed
  - far away, less likely

- Ideally, this is probably related to the area swept out by the robot
  - very hard to compute exactly
  - usually heuristic distance is used

- Typical approaches
  - Euclidean distance on some embedding of c-space
  - Alternative is to create a weighted combination of translation and rotational “distances”
  - Efficiency varies greatly depending on embedding

Local Planner

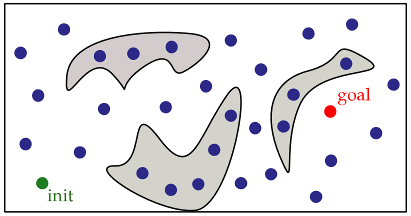
- The local planner should be reasonably fast and can be simple to implement

- A simple way is to do subdivision on straight line path
  – Decompose motion into a straight lines in configuration space
  – Split the line in half; check for collision
  – If none, recurse on halves until distance is small

PRM Applied to a Point Robot

$q = (x, y) \leftarrow \textsc{Sample}()$

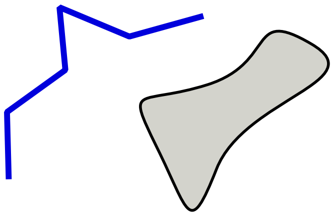
- $x \leftarrow \textsc{rand}(\mathsf{min}_x, \mathsf{max}_x)$
- $y \leftarrow \textsc{rand}(\mathsf{min}_y, \mathsf{max}_y)$



PRM for a Serial Chain

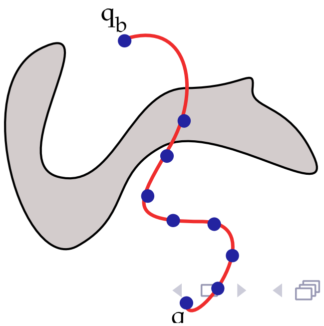
$q = (\theta_1, \theta_2, \ldots, \theta_n) \leftarrow \textsc{Sample}()$

- $\theta_i \leftarrow \textsc{rand}(-\pi, \pi), \forall i \in [1, n]$



ATHCOLLISIONFREE(path)

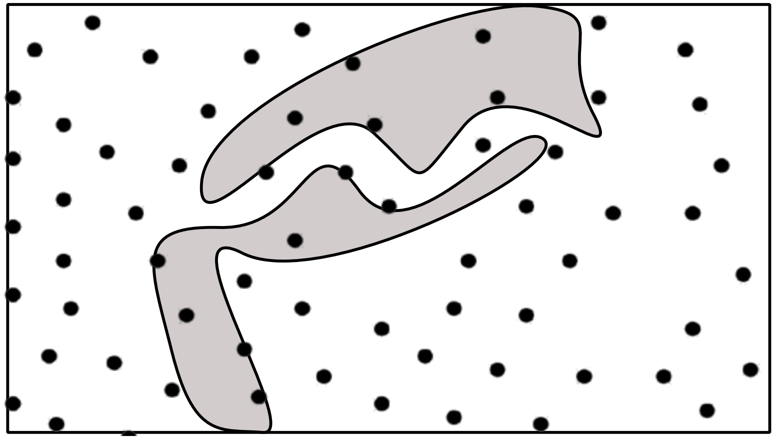
- Incremental approach
- Subdivision approach



[everest] [skeleton] [knot] [manip]

Sampling Strategies

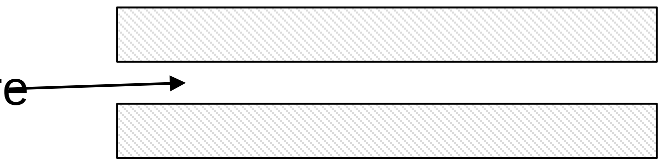
- The narrow corridor problem
  - probability of finding a path related to joint visibility area under uniform sampling



- Other approaches:
  - Bridge planner
    - sample randomly
    - check pairs in collision to see if midpoint (or random distance) is not
    - Somehow use generalized Voronoi diagrams
  - Visibility-based sampling
    - only keep configuration that
      - cannot be connected to an existing component, or
      - connect 2 existing components

Sampling Strategies

- Recall the narrow corridor problem
  - probability of finding a path related to joint visibility area under uniform sampling
  - Few samples will be available in here



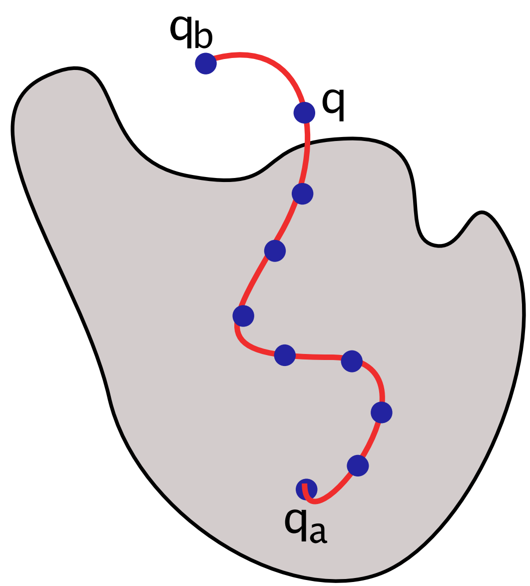
- Other approaches:
  - sampling near obstacles
    - Obstacle-Based PRM (OBPRM)
    - find samples in obstacles (uniform)
    - pick a random direction v
    - search for a free configuration in direction v
  - Gaussian OBPRM
    - generate a random sample
    - generate a Gaussian sample around the sample
    - only keep the sample if the Gaussian sample is in collision
  - Dilated obstacles
    - allow some interpenetration to enhance the likelihood of finding paths, then "fix up" later.

Obstacle Sampling

Objective: Increase Sampling Inside/Near Narrow Passages
Approach: Move samples in collision outside obstacle boundary

GenerateCollisionFreeConfig [Amato, Bayazit, Dale, Jones, Vallejo: WAFR 1998]

```
q_a ← generate config uniformly at random
if IsConfigCollisionFree(q_a) = true then
    return q_a
else
    q_b ← generate config uniformly at random
    path ← GeneratePath(q_a, q_b)
    for t = δ to |path| by δ do
        if IsConfigCollisionFree(path(t)) then
            return path(t)
return null
```



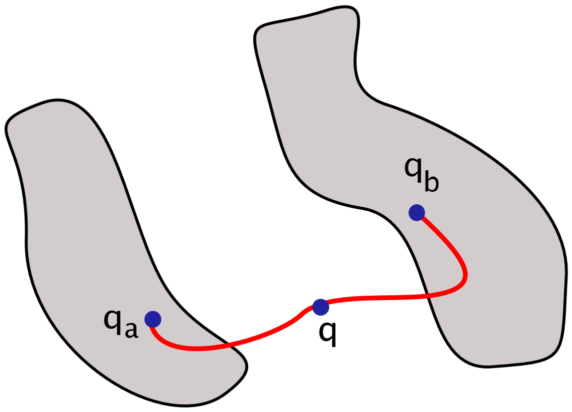
Bridge Sampling

Objective: Increase Sampling Inside/Near Narrow Passages
Approach: Create "bridge" between samples in collision

[Hsu, Jiang, Reif, Sun: ICRA 2003]

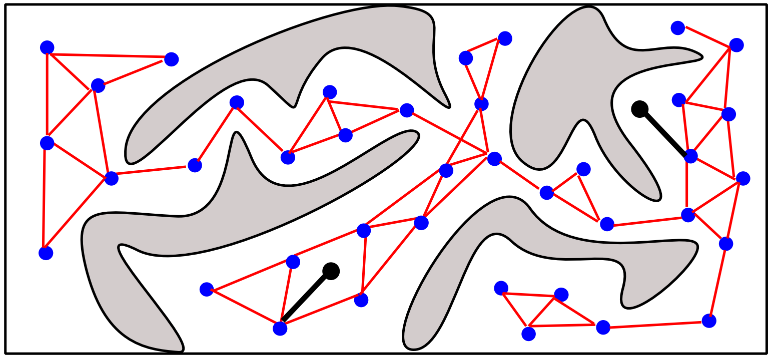
GenerateCollisionFreeConfig

```
qa ← generate config uniformly at random
qb ← generate config uniformly at random
oka ← IsConfigCollisionFree(qa)
okb ← IsConfigCollisionFree(qb)
if oka = false and okb = false then
   path ← GeneratePath(qa, qb)
   q ← path(0.5|path|)
   if IsConfigCollisionFree(q) then
      return q
return null
```



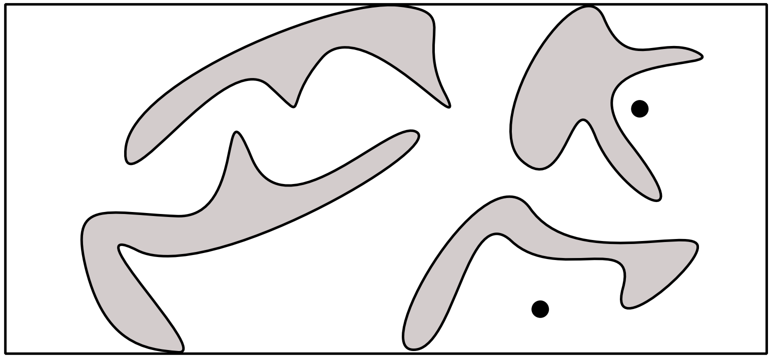
What about Single Queries?

- PRM-based planners aim to construct a roadmap that captures the whole connectivity of the configuration space



s to solve *multiple* queries

- Maybe a bit too much when the objective is to solve a *single* query

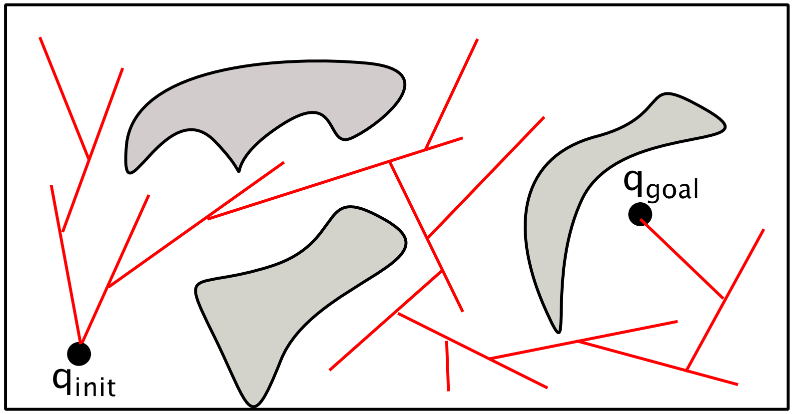


Single Query Planners

- Use ESTs (expansive space trees) and RRTs (rapidly exploring random trees.
  - also work for nonholonomic and kinodynamic planning
  - each of these is initial condition dependent (requires integration) and thus it is hard to build a graph offline
- Generally build two trees from start and goal configurations
  - each tree grows toward the other
  - once linkage is made defines a path from start to goal
- Key is to define sampling strategies that focus on
  - unexplored areas of the free space
  - move toward start or goal
- For completeness, necessary to show that the algorithm will eventually cover the entire space

Tree-Based Motion Planning

Grow a tree in the free configuration space from q_{init} *toward* q_{goal}



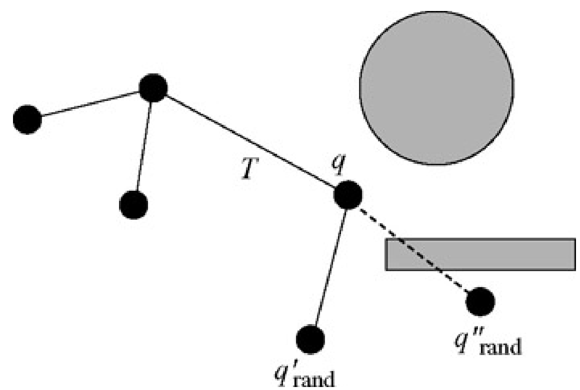
uration?

Expansive Space Trees (EST)

Push the tree frontier in the free configuration space

[Hsu, Rock, Motwani, Latombe: 1999]

- EST relies on a probability distribution to guide tree growth
- EST associates a weight $w(q)$ with each tree configuration q
- $w(q)$ is a running estimate on importance of selecting q as the tree configuration from which to add a new tree branch
  - $w(q) = \frac{1}{1+\deg(q)}$
  - $w(q) = 1/(1+$ number of neighbors near $q)$
  - combination of different strategies



$\textsc{SelectConfigFromTree}$

- select q in $\mathcal{T}$ with probability $w(q)/\sum_{q' \in \mathcal{T}} w(q')$

$\textsc{AddTreeBranchFromConfig}(\mathcal{T}, q)$

- $q_{\text{near}} \leftarrow$ sample a collision-free configuration near q
- $\text{path} \leftarrow$ generate path from q to q_{near}
- if path is collision-free, then add q_{near} and (q, q_{near}) to $\mathcal{T}$

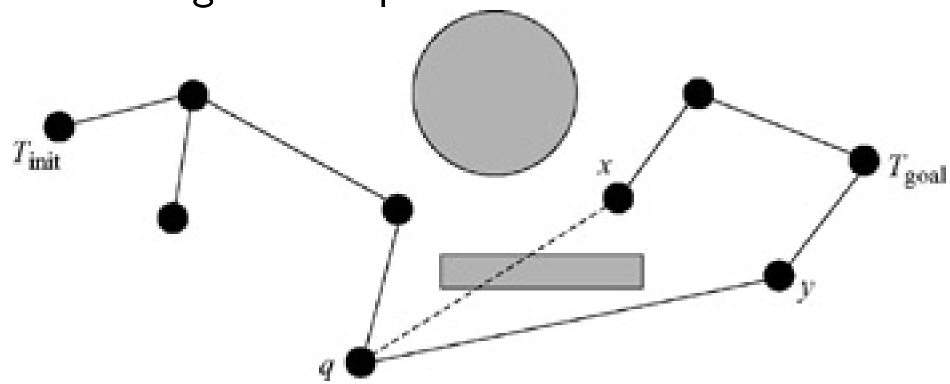
BiDirectional Motion Planning (EST)

Grow two trees, rooted at $q_{\rm init}$ and $q_{\rm goal}$, towards each other

- Bi-directional trees improve computational efficiency compared to a single tree
- Growth slows down significantly later than when using a single tree
- Fewer configurations in each tree, which imposes less of a computational burden
- Each tree explores a different part of the configuration space

$\textsc{BiTree}(q_{\rm init}, q_{\rm goal})$

```
𝒯_init ← create tree rooted at q_init
𝒯_goal ← create tree rooted at q_goal
while solution not found do
    add new branch to 𝒯_init
    add new branch to 𝒯_goal
    attempt to connect neighboring configurations
    from the two trees
    if successful, return path from q_init to q_goal
```



- Different tree planners can be used to grow each of the trees
- E.g., RRT can be used for one tree and EST can be used for the other

RRT Algorithm

- RRT sampling distribution converges to uniform
  - implies probabilistic completeness of the algorithm

1. Choose a sample q_{rand} in free space
2. Find q_{near} (closest configuration to q_{rand} in T)
3. Try a point q_{new} some distance step-size from q_{near} toward q_{new}
   - If q_{new} is collision-free, add edge (q_{near},q_{new}) to the graph

A variation is to repeatedly move toward q_{rand} as far as possible

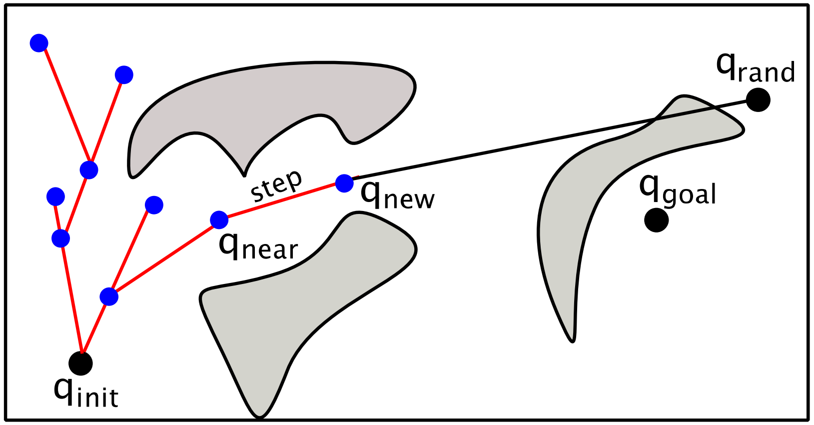
The key is how to sample q_{new}

random guarantees completeness but is inefficient
$q_{new} = q_{goal}$ is efficient but has local minima
solution is to alternate randomly between these two

Rapidly-exploring Random Trees

Pull the tree toward random samples in the configuration space

[LaValle, Kuffner: 1999]



Improvements

Aspects for Improvement

- BasicRRT does not take advantage of q_{goal}
- Tree is pulled towards random directions based on the uniform sampling of
- In particular, tree growth is not directed towards q_{goal}

Suggested Improvements in the Literature

- Introduce goal-bias to tree growth (known as GoalBiasRRT)
  - q_{rand} is selected as q_{goal} with probability p
  - q_{rand} is selected based on uniform sampling of Q with probability $1 - p$
  - Probability p is commonly set to ≈ 0.05

Analysis of PRM

- Goal: show **probabilistic completeness**:
  - Suppose that a,b $\in Q_{free}$ can be connected by a free path. PRM is *probabilistically complete* if, for any (a,b)

$$\lim_{n\to\infty} \Pr[(a,b)\text{FAILURE}] = 0$$

where n is the number of samples used to construct the roadmap

- Basic idea:
  - reduce the path to a set of open balls in free space
  - figure out how many samples it will take to generate a pair of points in those balls
  - connect those points to create a path
- Assuming that a path between a and b exists, the probability of that PRM will fail depends on
  1. The length of the path
  2. The distance of the path to the obstacle
  3. The number of configuration in the roadmap

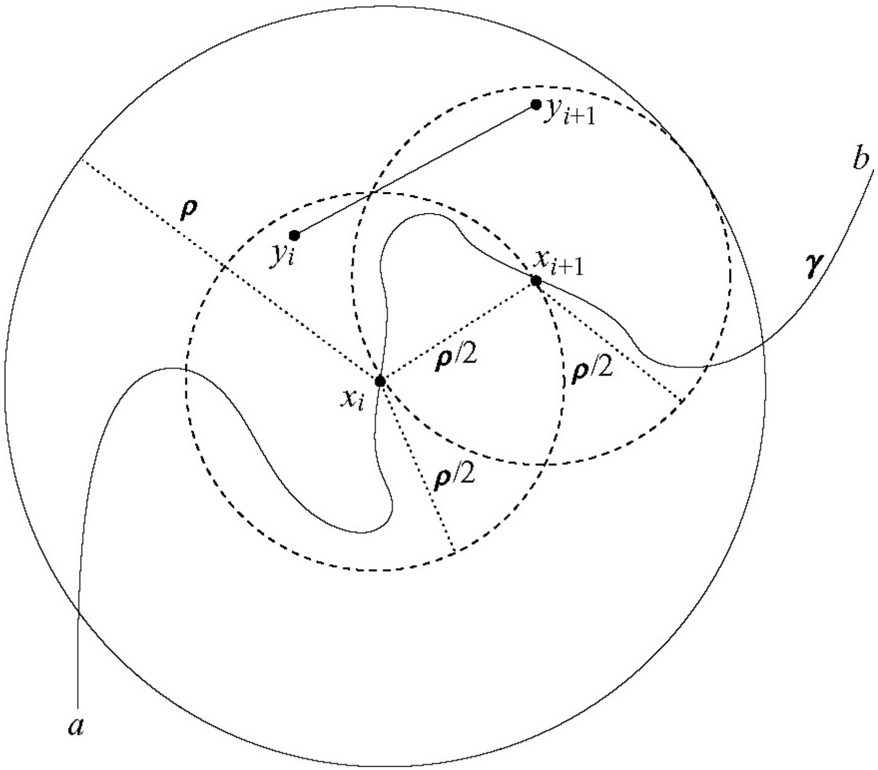
Analysis of PRM

- A path from a to b can be described by a function γ: $[0,1] \rightarrow Q_{free}$ with $\gamma(0) = a$ and $\gamma(1) = b$
- Let $clr(\gamma)$ be the minimum distance between γ and any obstacle.
- Let μ be a volume measure on the space, for set $S \in \mathbb{R}^d, \mu(S)$ is the volume of set S
- $B_\rho(x)$ is a ball centered at x of radius ρ
- For uniform sampling of $A \subset Q_{free}$ the probability that a random point $x \in Q_{\text{free}}$, then $\Pr(x \in A) = \mu(A)/\mu(Q_{free})$

A Simple Idea

Basic Idea

- Reduce path to a set of open balls in Q_{free}
- Calculate probability of generating samples in those balls
- Connect samples in different balls via straight-line paths to compute solution path



Completeness

Components

- Free configuration space Q_{free}: arbitrary open subset of $[0,1]^d$
- Local connector: connects $a, b \in Q_{\mathrm{free}}$ via a straight-line path and succeeds if path lies entirely in Q_{free}
- Collection of roadmap samples from Q_{free}

Let $a, b \in Q_{\mathrm{free}}$ such that there exists a path γ between a and b lying in Q_{free}. Then the probability that PRM correctly answers the query (a, b) after generating n collision-free configurations is given by

$$\Pr[(a,b)\mathrm{SUCCESS}] \geq 1 - \left\lceil \frac{2L}{\rho} \right\rceil e^{-\sigma \rho^d n},$$

where

- L is the length of the path γ
- $\rho = \mathrm{clr}(\gamma)$ is the clearance of path γ from obstacles
- $\sigma = \frac{\mu(B_1(\cdot))}{2^d \mu(Q_{\mathrm{free}})}$
- $\mu(B_1(\cdot))$ is the volume of the unit ball in $\mathbb{R}^d$
- $\mu(Q_{\mathrm{free}})$ is the volume of Q_{free}

The Proof Sketch

- Note that clearance $\rho = \mathrm{clr}(\gamma) > 0$
- Let $m = \left\lceil \frac{2L}{\rho} \right\rceil$. Then, γ can be covered with m balls $B_{\rho/2}(q_i)$ where $a = q_1, \ldots, q_m = b$
- Let $y_i \in B_{\rho/2}(q_i)$ and $y_{i+1} \in B_{\rho/2}(q_{i+1})$.
  Then, the straight-line segment $\overline{y_i y_{i+1}} \in Q_{\mathrm{free}}$, since $y_i, y_{i+1} \in B_{\rho}(q_i)$
- $I_i \overset{def}{=}$ indicator variable that there exists $y \in V$ s.t. $y \in B_{\rho/2}(q_i)$
- $\mathrm{Pr}[(a, b)\mathrm{FAILURE}] \leq \mathrm{Pr}\left[\bigvee_{i=1}^{m} I_i = 0\right] \leq \sum_{i=1}^{m} \mathrm{Pr}[I_i = 0]$
  - Note that $\mathrm{Pr}[I_i = 0] = \left(1 - \frac{\mu(B_{\rho/2}(q_i))}{\mu(Q_{\mathrm{free}})}\right)^n$
    i.e., probability that none of the n `PRM` samples falls in $B_{\rho/2}(q_i)$
  - I_i's are independent because of uniform samling in `PRM`

Therefore, $\mathrm{Pr}[(a, b)\mathrm{FAILURE}] \leq m\left(1 - \frac{\mu(B_{\rho/2}(\cdot))}{\mu(Q_{\mathrm{free}})}\right)^n$

- $\frac{\mu(B_{\rho/2}(\cdot))}{\mu(Q_{\mathrm{free}})} = \frac{\left(\frac{\rho}{2}\right)^d \mu(B_1(\cdot))}{\mu(Q_{\mathrm{free}})} = \sigma\rho^d$

Therefore, $\mathrm{Pr}[(a, b)\mathrm{FAILURE}] \leq m\left(1 - \sigma\rho^d\right)^n \leq m e^{-\sigma\rho^d} = \left\lceil \frac{2L}{\rho} \right\rceil e^{-\sigma\rho^d}\mathrm{n}$ $\square$

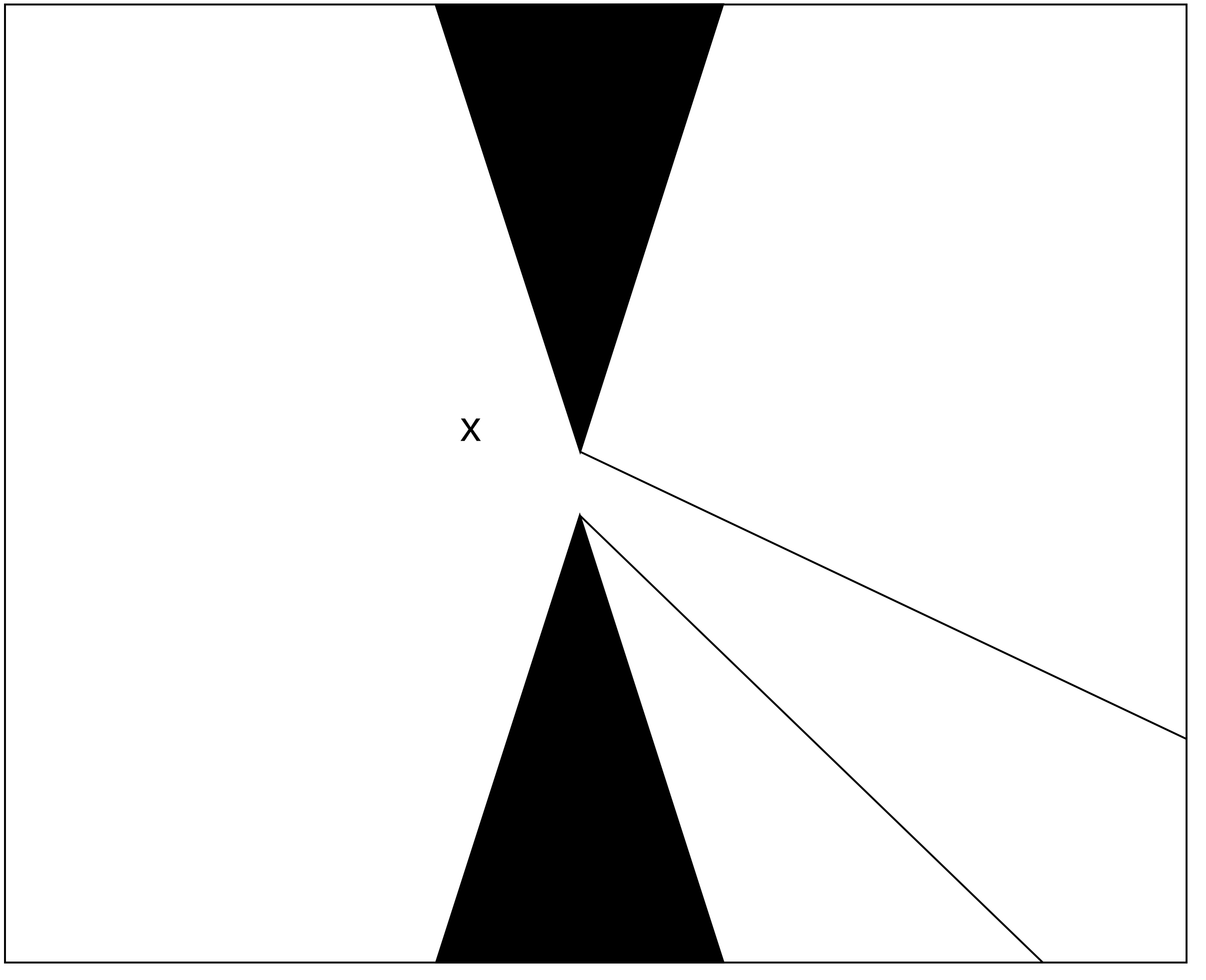
Understanding Complexity

- Relationship between “difficulty” of space and number of samples (avoids dependence on length of path which is unknown)

- Relies on so-called (ε, α, β) expansiveness of space
  - ε measures the fraction of space reachable from any point
    - called ε-good if all points “see” at least ε of the space
    - $\forall x \in Q_{free} \quad \mu(reach(x)) \geq \epsilon\, \mu(Q_{free})$
  - $lookout_\beta$ of a set S is the set of points that can see at least β of Q_{free} - S

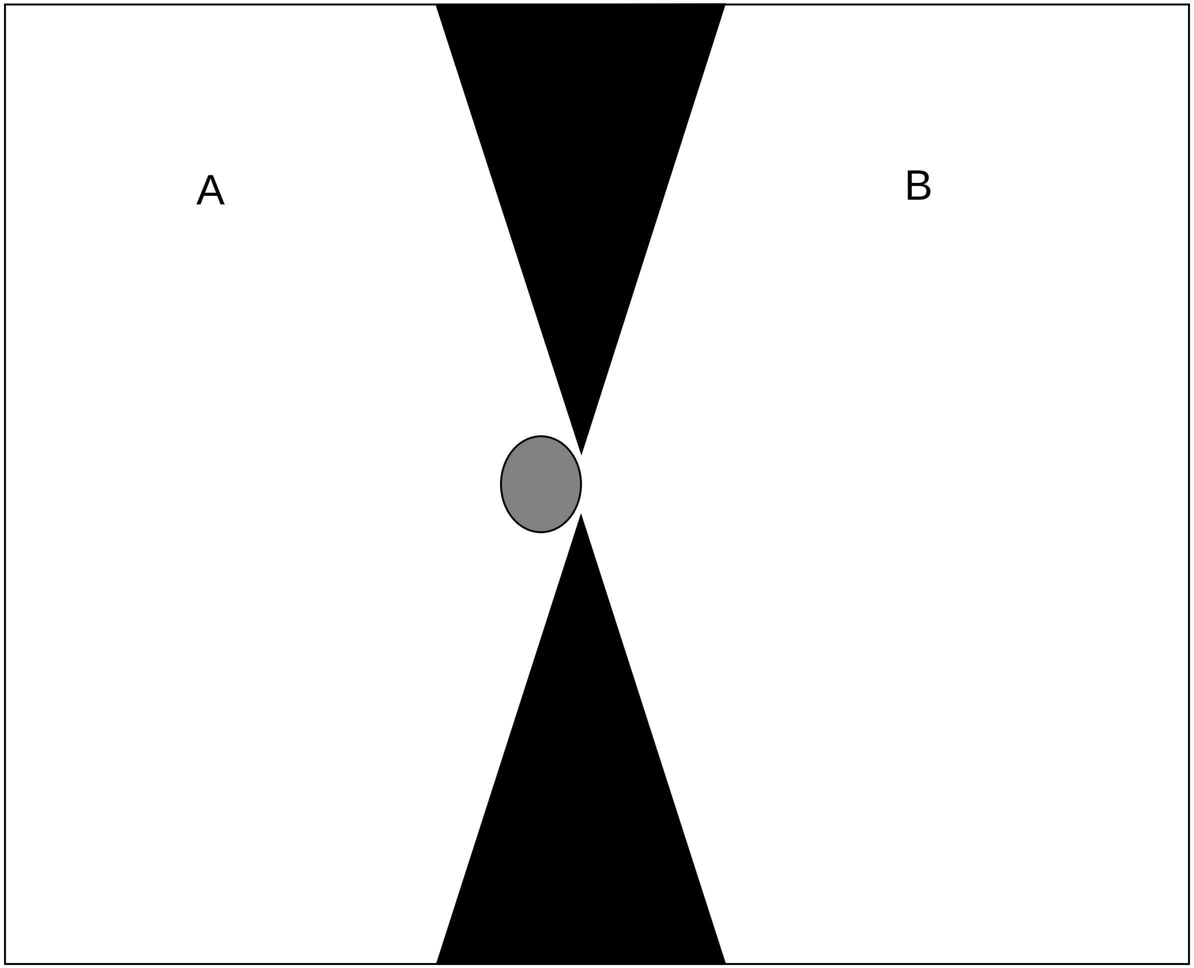
- A space is (ε α β) expansive if
  - it is ε-good
  - for any connected $S \subseteq Q_{free}$ $\mu(lookout_\beta(S)) \geq \alpha\, \mu(S)$
  - intuitively, this capture how easy it is to choose points that link up the space

Expansive Spaces



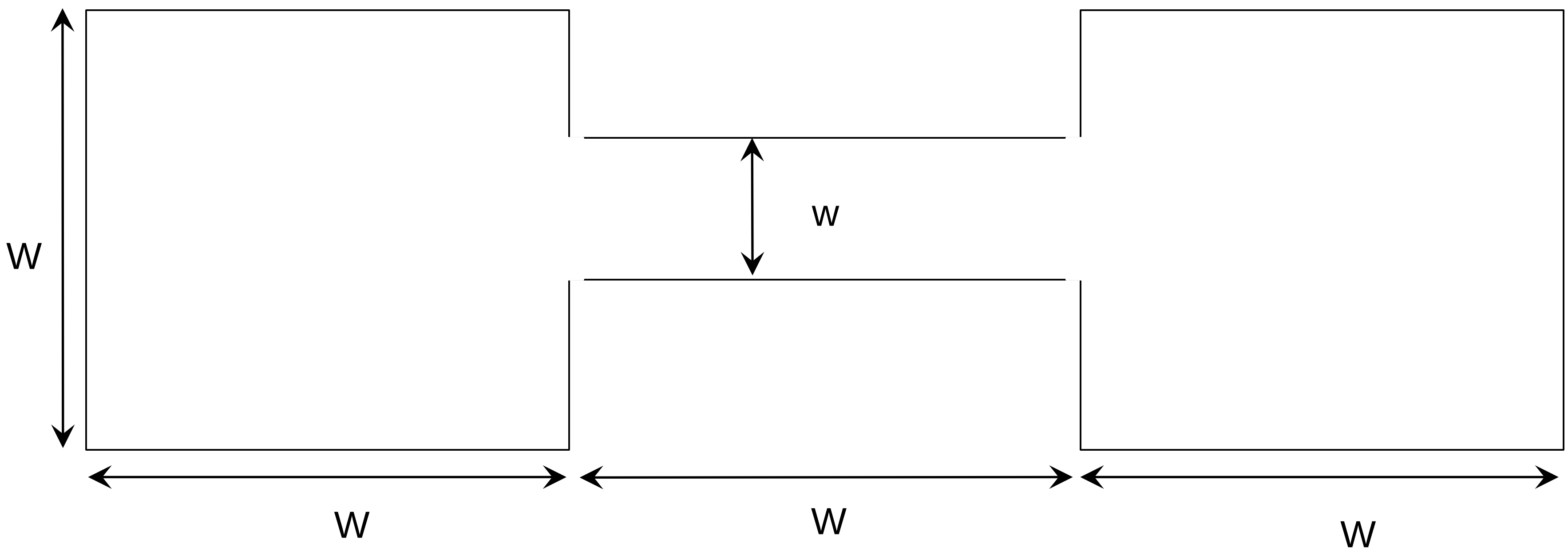
Reachability

Expansive Spaces



Only a small subset of A can see a large fraction of points in B

An Example



Given n random samples, what are the odds the resulting graph is connected?

Summary

- PRS --- a basic sampling-based planner
  - useful for multiple queries
  - can be made efficient using various sampling and evaluation tricks
- EST and RRT
  - single query planners
  - generate trees and merge them
  - again many variations based on applications
- Analysis
  - general idea of completeness
  - particular proof based on sampling
  - other ideas based on structure of environment
- Other Applications