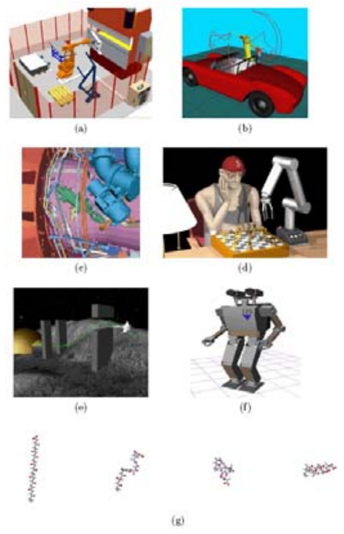
Robotic Motion Planning: Sample-Based Motion Planning

Robotics Institute 16-735

http://voronoi.sbp.ri.cmu.edu/~motion

Howie Choset

http://voronoi.sbp.ri.cmu.edu/~choset



Path-Planning in High Dimensions

- IDEAL:

  Build a ***complete*** motion planner

- PROBLEM:

  Path Planning is PSPACE-hard [Reif 79, Hopcroft et al. 84 & 86]

  Complexity is exponential in the dimension of the robot's C-space [Canny 86]

Building Configuration Space

⇨ Heuristic algorithms trade off completeness for practical efficiency. Weaker performance guarantee.

Ways to Simplify Problem

- Project search to lower-dimensional space
- Limit the number of possibilities
  (add constraints, reduce "volume"
  of free space)
- Sacrifice optimality, completeness

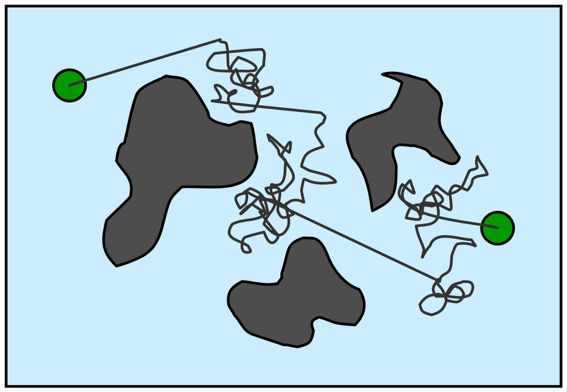
The Rise of Monte Carlo Techniques

- KEY IDEA:
  Rather than exhaustively explore ALL possibilities, randomly explore a smaller subset of possibilities while keeping track of progress
- Facilities "probing" deeper in a search tree much earlier than any exhaustive algorithm can
- What's the catch?
  Typically we must sacrifice both *completeness* and *optimality*
  Classic tradeoff between solution quality and runtime performace

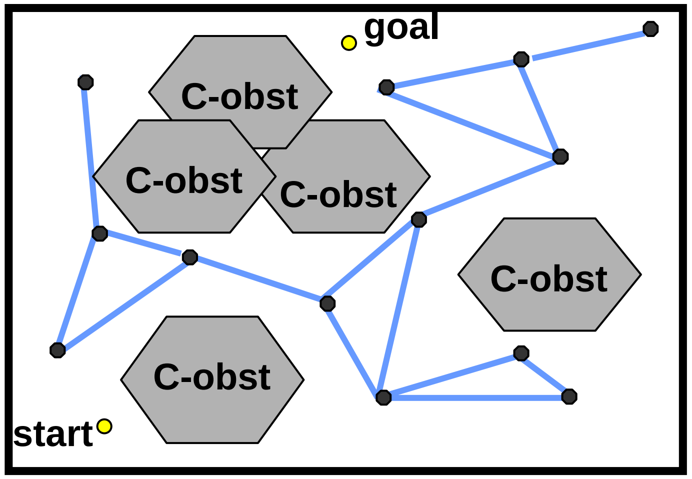
Sampling Based Planning:

Search for collision-free path only by sampling points.

EXAMPLE: Potential-Field



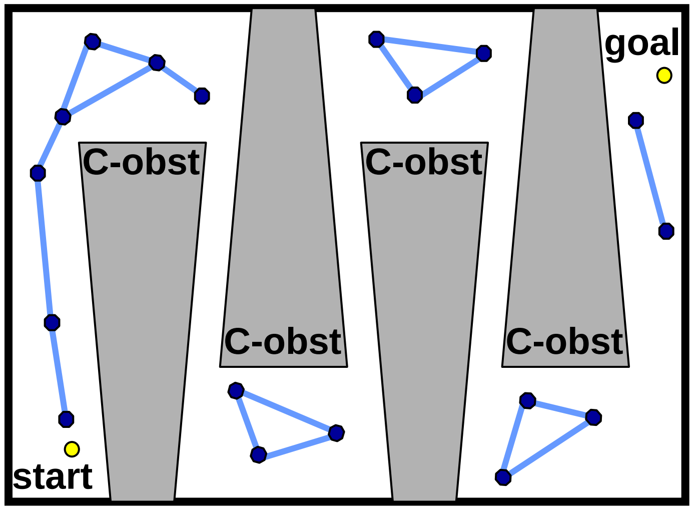
Good news, but bad news too



Sample-based: The Good News

1. *probabilistically complete*
2. Do not construct the C-space
3. apply easily to high-dimensional C-space
4. support fast queries w/ enough preprocessing

Many success stories where PRMs solve previously unsolved problems



Sample-Based: The Bad News

1. don't work as well for some problems:
– unlikely to sample nodes in *narrow passages*
– hard to sample/connect nodes on constraint surfaces
2. No optimality or completeness

Everyone is doing it.

- **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]
  - PRM for Closed Chain Systems [Lavalle/Yakey/Kavraki 99, Han/Amato 00]
  - PRM for Flexible/Deformable Objects [Kavraki et al 98, Bayazit/Lien/Amato 01]
  - 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]
  - Single Shot [Vallejo/Remmler/Amato 01]
  - Bio-Applications: Protein Folding [Song/Thomas/Amato 01,02,03, Apaydin et al 01,02]
  - Lazy Evaluation Methods: [Nielsen/Kavraki 00 Bohlin/Kavraki 00, Song/Miller/Amato 01, 03]
  - Seth Hutchinson workspace-based approach, 2001

- **Related Methods**
  - Ariadnes Clew Algorithm [Ahuactzin et al, 92]
  - RRT (Rapidly Exploring Random Trees) [Lavalle/Kuffner 99]

Overview

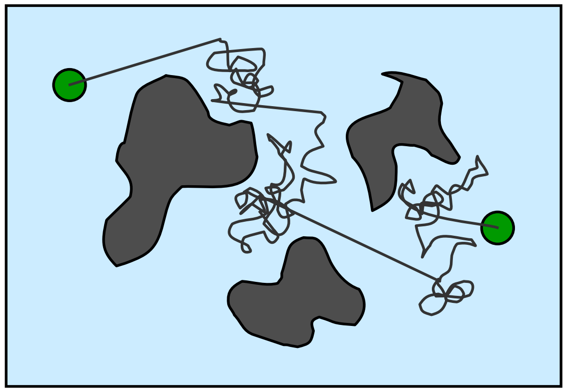
- Probabilistic RoadMap Planning (PRM) by Kavraki
  - samples to find free configurations
  - connects the configurations (creates a graph)
  - is designed to be a multi-query planner
- Expansive-Spaces Tree planner (EST) and Rapidly-exploring Random Tree planner (RRT)
  - are appropriate for single query problems
- Probabilistic Roadmap of Tree (PRT) combines both ideas

High-Dimensional Planning as of 1999

Single-Query:

Barraquand, Latombe ’89; Mazer, Talbi, Ahuactzin, Bessiere ’92; Hsu, Latombe, Motwani ’97; Vallejo, Jones, Amato ’99;

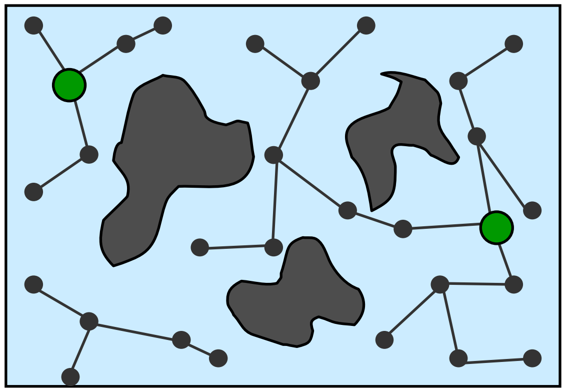
EXAMPLE: Potential-Field



Multiple-Query:

Kavraki, Svestka, Latombe, Overmars ’95; Amato, Wu ’96;   Simeon, Laumound, Nissoux ’99;   Boor, Overmars, van der Stappen ’99;

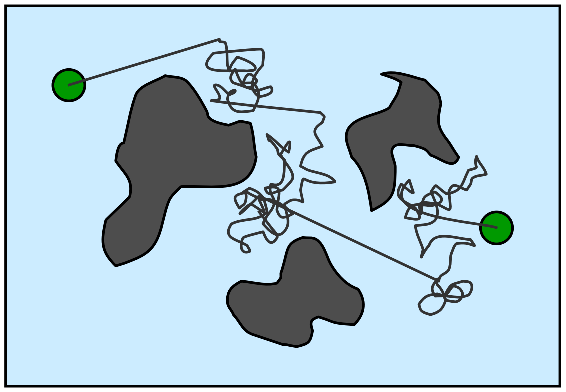
EXAMPLE: PRM



Randomized Potential Functions (Barranquand and Latome)

May take a long time in local minima

EXAMPLE: Potential-Field



Probabalistic Roadmaps (Kavraki, Latombe and lots more)

- Learning Phase
  - Construction Step
  - Expansion Step

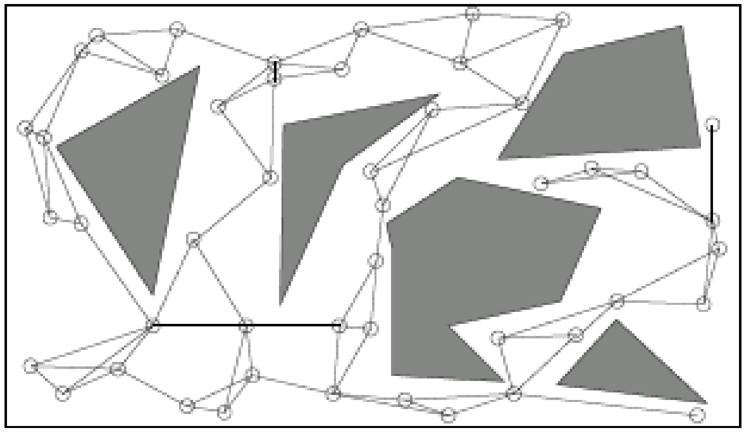
- Query Phase

The Learning Phase

- Construct a probabilistic roadmap by generating random free configurations of the robot and connecting them using a simple, but very fast motion planner, also know as a *local planner*
- Store as a graph whose nodes are the configurations and whose edges are the paths computed by the *local planner*

Learning Phase (Construction Step)

- Initially, the graph **G = (V, E)** is empty
- Then, repeatedly, a *random free configuration* is generated and added to V
- For every new node c, select a number of nodes from V and try to connect c to each of them using the *local planner*.
- If a path is found between c and the selected node v, the edge (c,v) is added to E. The path itself is not memorized (usually).



How do we determine a random free configuration?

- We want the nodes of V to be a rather **uniform** sampling of Q_{free}
  - Draw each of its coordinates from the interval of values of the corresponding degrees of freedom. (Use the uniform probability distribution over the interval)
  - Check for collision both with robot itself and with obstacles
  - If collision free, add to V, otherwise discard
  - What about rotations? Sample Euler angles gives samples near poles, what about quartenions?
- This is HUGE TOPIC, which we will get to later

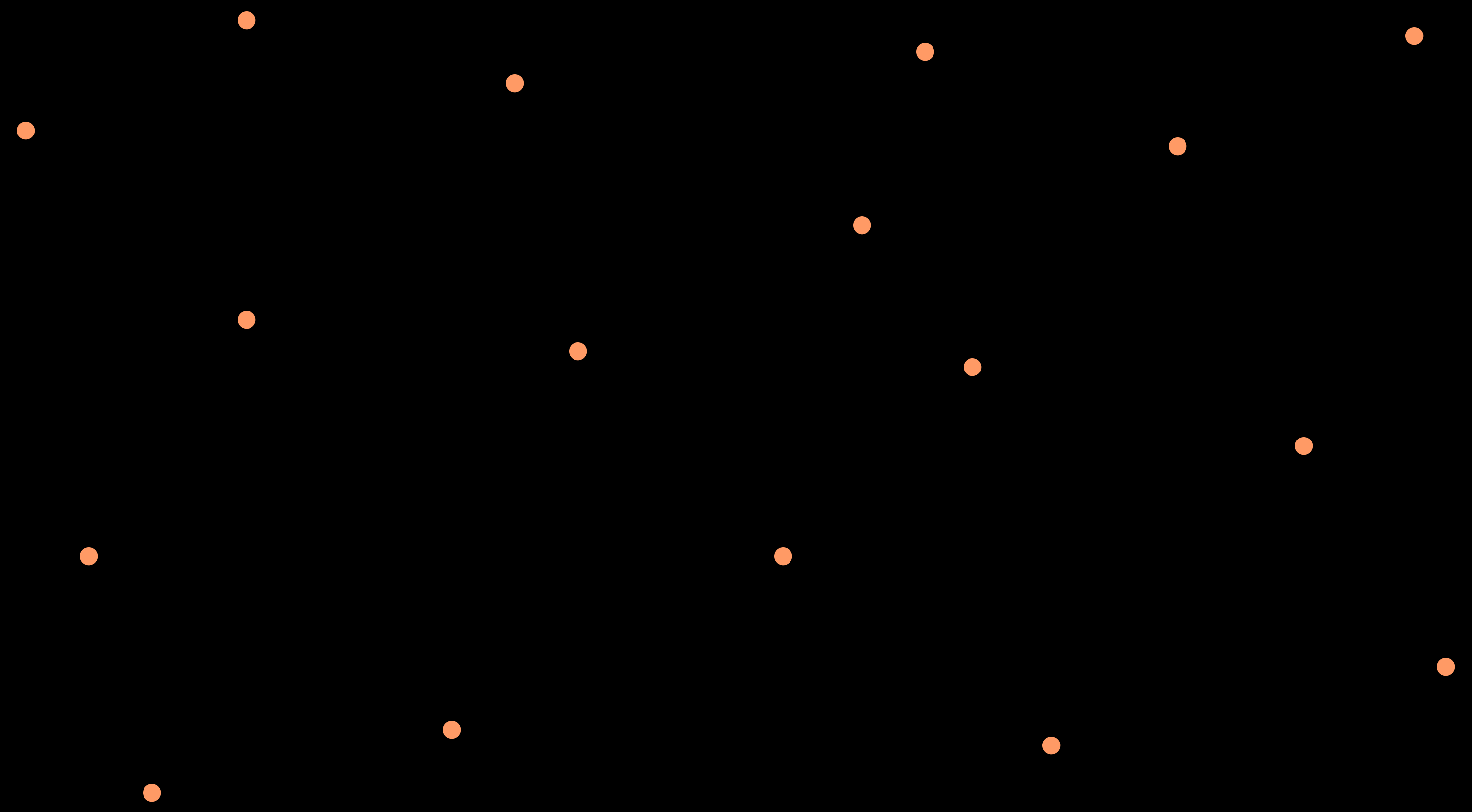
What’s the local path planner?

- Can pick different ones
  - Nondeterministic – have to store local paths in roadmap
  - Powerful - slower but could take fewer nodes but takes more time
  - Fast - less powerful, needs more nodes

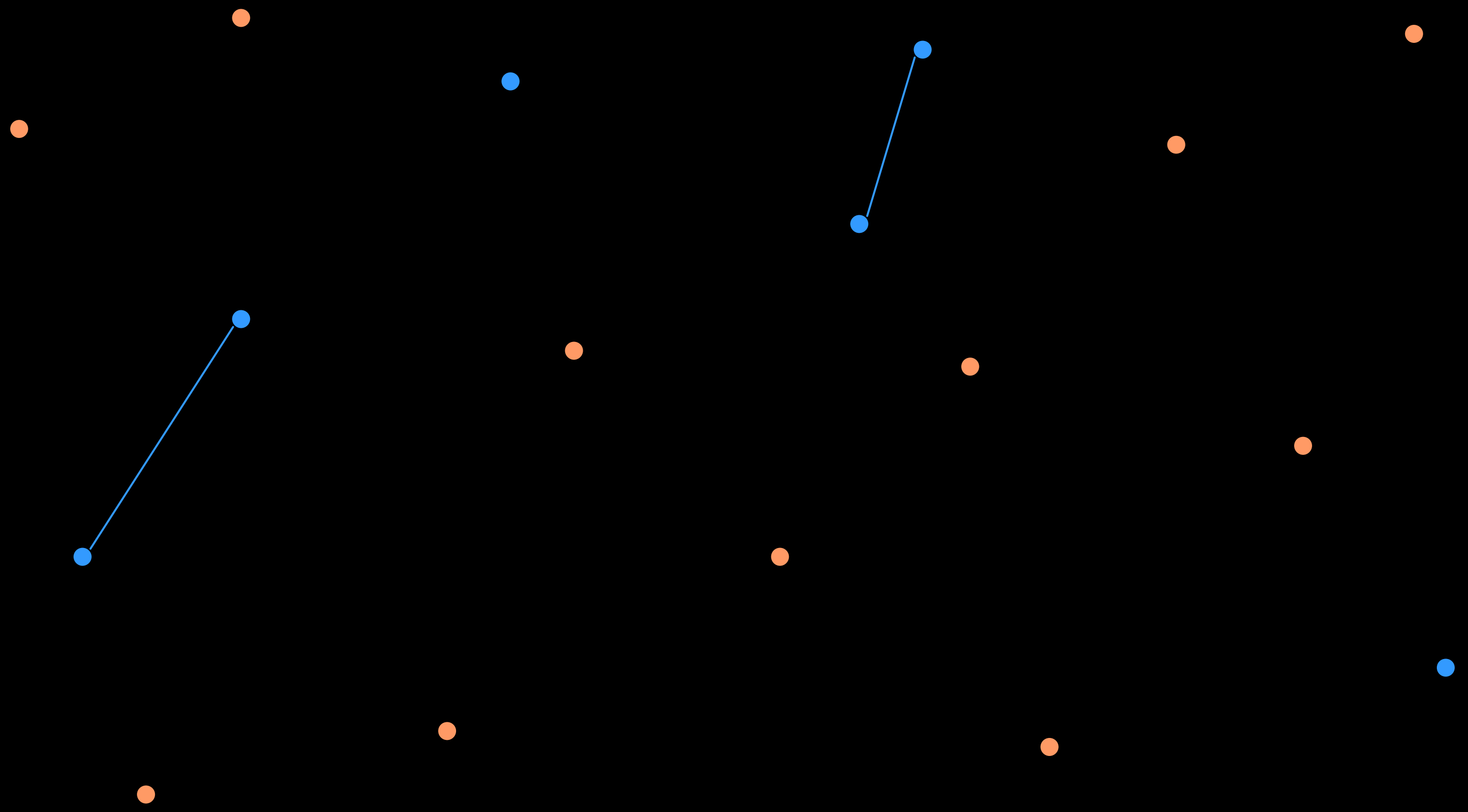
Go with the fast one

- Need to make sure start and goal configurations can connect to graph, which requires a somewhat dense roadmap
- Can reuse local planner at query time to connect start and goal configurations
- Don’t need to memorize local paths

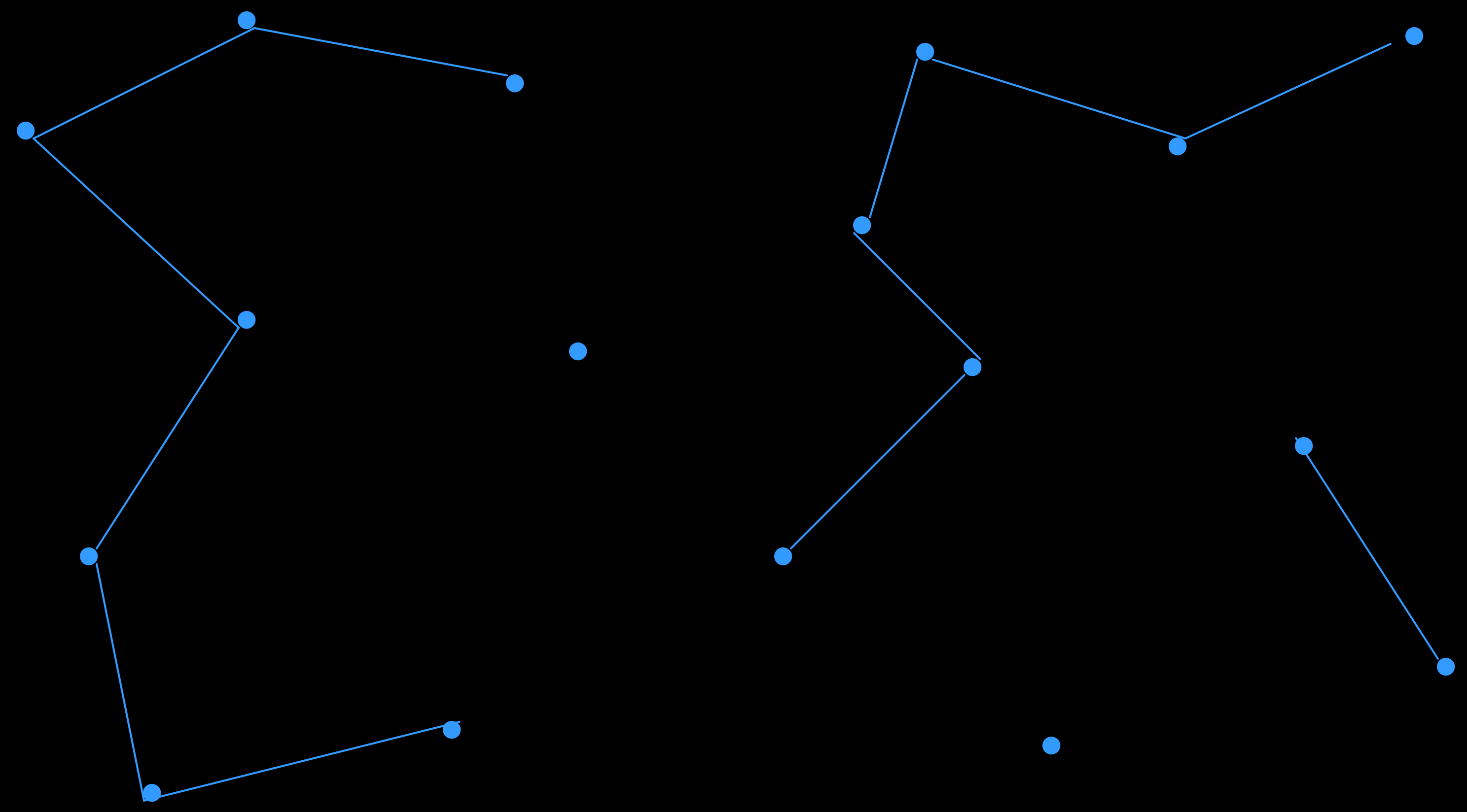
Create random configurations



Update Neighboring Nodes’ Edges



End of Construction Step



Basic PRM, reviewed

- Goal: construct a graph $G=(V,E)$ where $e=(q_1,q_2)$ is an edge only if there is a collision-free path from q_1 to q_2

- Notation:
  - let Δ be a (deterministic) local planner that is correct (but may not be complete)
    $D: Q \times Q \rightarrow [0,\infty]$ --- a distance function on Q

```
1. While $|V| < n$ do
     – sample until a collision-free configuration q is found; add q to V

2. for all $q \in V$
      for all $q' \in N_q$ (k closest neighbors of q)
         if $\Delta(q,q')$ = True
            $E = E \cup \{(q,q')\}$
         end
      end
   end
```

Distance Functions

- Really, D 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

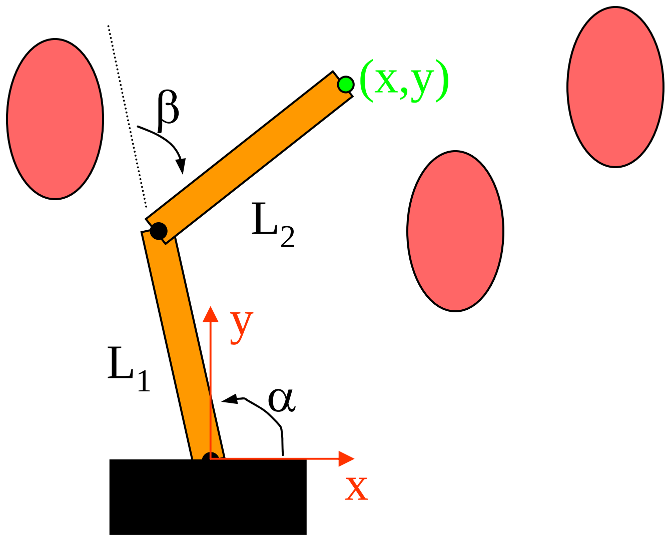
Evaluating the effect of a potential field on the robot would involve computing an integral over the area/volume defined by the robot, and this can be quite complex (both mathematically and computationally). An alternative approach is to select a subset of points on the robot, called control points, and to define a workspace potential for each of these points.

- Typical approaches
  - Euclidean distance on some embedding of c-space $dist(q', q'') = \| \mathrm{emb}(q') - \mathrm{emb}(q'') \|$
    - Embedding is often based on control points (recall end of potential field chapter)
  - Alternative is to create a weighted combination of translation and rotational "distances" $dist(q', q'') = w_t \|X' - X''\| + w_r f(R', R'')$
  - Workspace volume

An Example

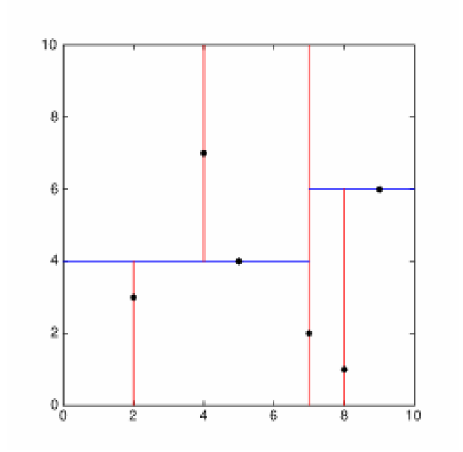
- Suppose that we have elliptical obstacles and a polygonal revolute robot.
  - Check intersection by
    - checking endpoints and line-ellipse intersection for each segment
    - do this for each link
  - Recall kinematics K: $T^n \rightarrow \Re(2)$
  - Approximate distance by vector norm on angles w/some minor hacks

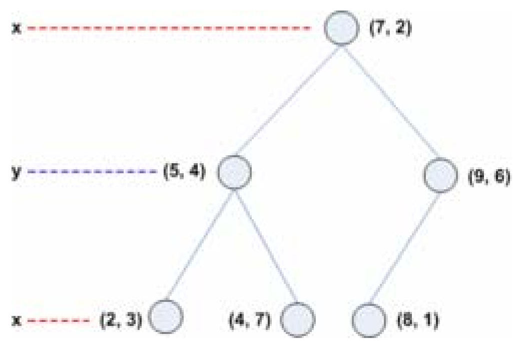
$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} L_1 c_\alpha \\ L_1 s_\alpha \end{pmatrix} + \begin{pmatrix} L_2 c_{\alpha+\beta} \\ L_2 s_{\alpha+\beta} \end{pmatrix}$$



Selecting Closest Neighbors

Why k?



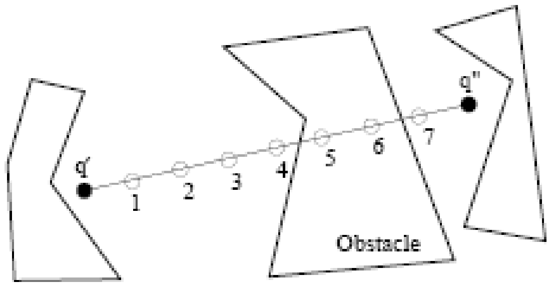


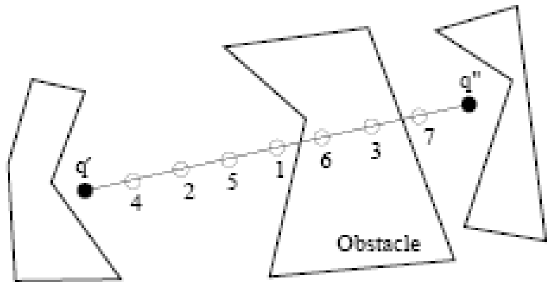
- kd-tree
  - Given: a set S of n points in d-dimensional space
  - Recursively
    - choose a plane P that splits S about evenly (usually in a coordinate dimension)
    - store P at node
    - apply to children S_l and S_r
  - Requires O(dn) storage, built in O(dn log n) time
  - Query takes $O(n^{1-1/d}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

- cell-based method
  - when each point is generated, hash to a cell location

Local Planner

- Again, chose the quick and dirty one
- Don’t necessarily store paths





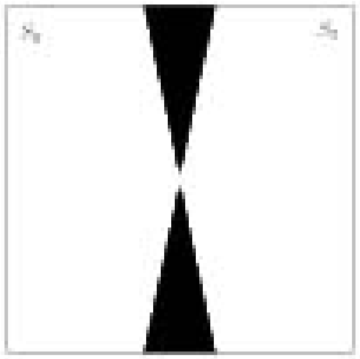
How to chose step_size?

<u>Next homework:</u> assume multi-line segment robot and polygonal obstacles

<u>Current homework</u>: due in a week

<u>Update progress report</u>: next week (maybe wed)

Expansion



- Sometimes G consists of several large and small components which do not effectively capture the connectivity of Q_{free}
- The graph can be disconnected at some narrow region
- Assign a positive weight w(c) to each node c in V

w(c) is a heuristic measure of the “difficulty” of the region around c. So w(c) is large when c is considered to be in a difficult region. We normalize w so that all weights together add up to one. The higher the weight, the higher the chances the node will get selected for expansion.

How to choose w(c) ?

- Can pick different heuristics
  - Count number of nodes of V lying within some predefined distance of c.
  - Check distance D from c to nearest connected component not containing c.
  - Use information collected by the local planner.

    (If the planner often fails to connect a node to others, then this indicates the node is in a difficult area).

How to choose w(c) ?

One Example:

At the end of the construction step, for each node c, compute the failure ratio $r_f(c)$ defined by:

$$r_f(c) = \frac{f(c)}{n(c)+1}$$

where n(c) is the total number of times the local planner tried to connect c to another node and f(c) is the number of times it failed.

How to choose w(c) ?

– At the beginning of the expansion step,
  for every node c in V, compute w(c) proportional to the failure ratio.

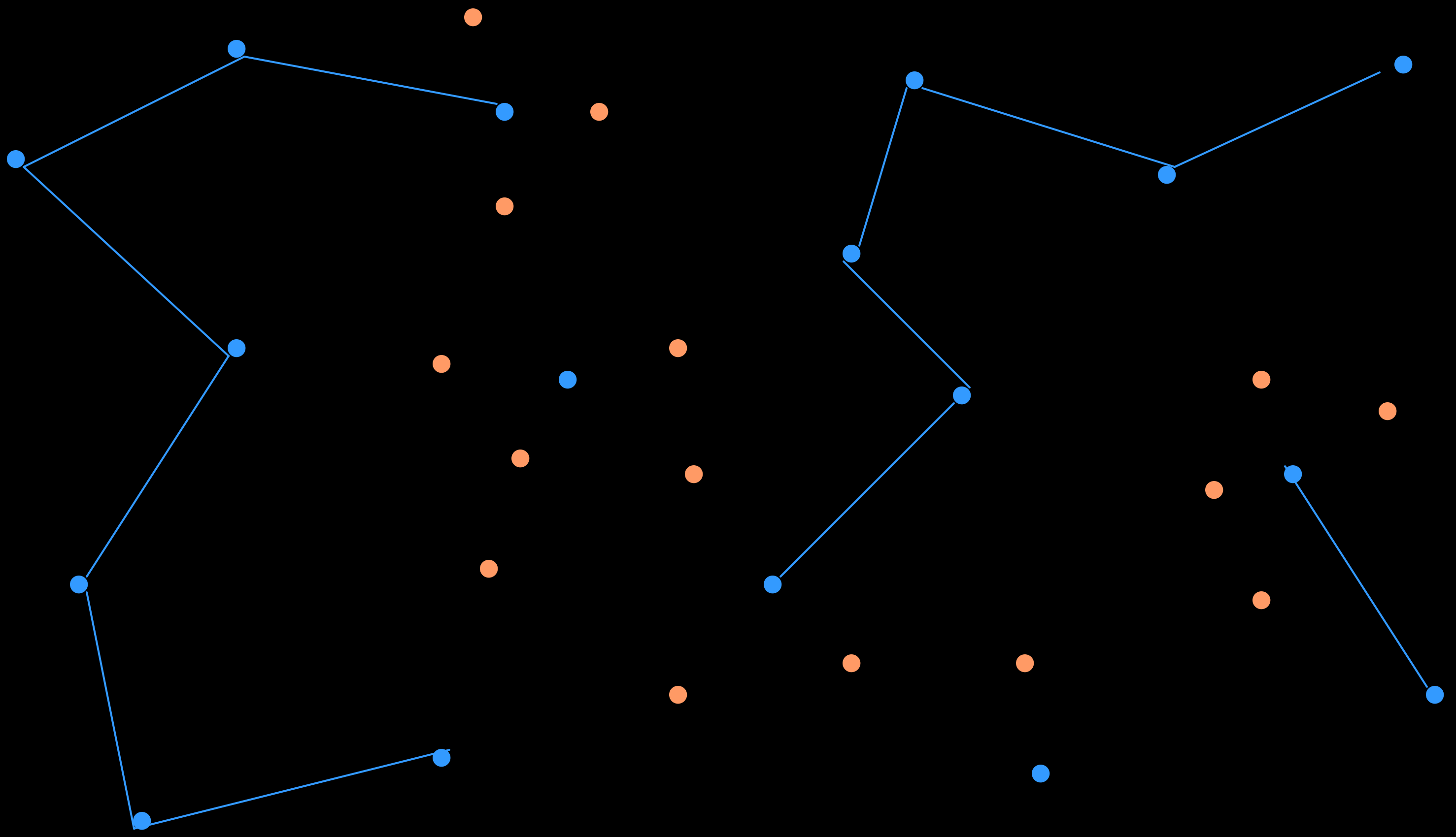
$$w(c) = \frac{r_f(c)}{\sum_{a \in V} r_f(a)}$$

Now that we have weights…

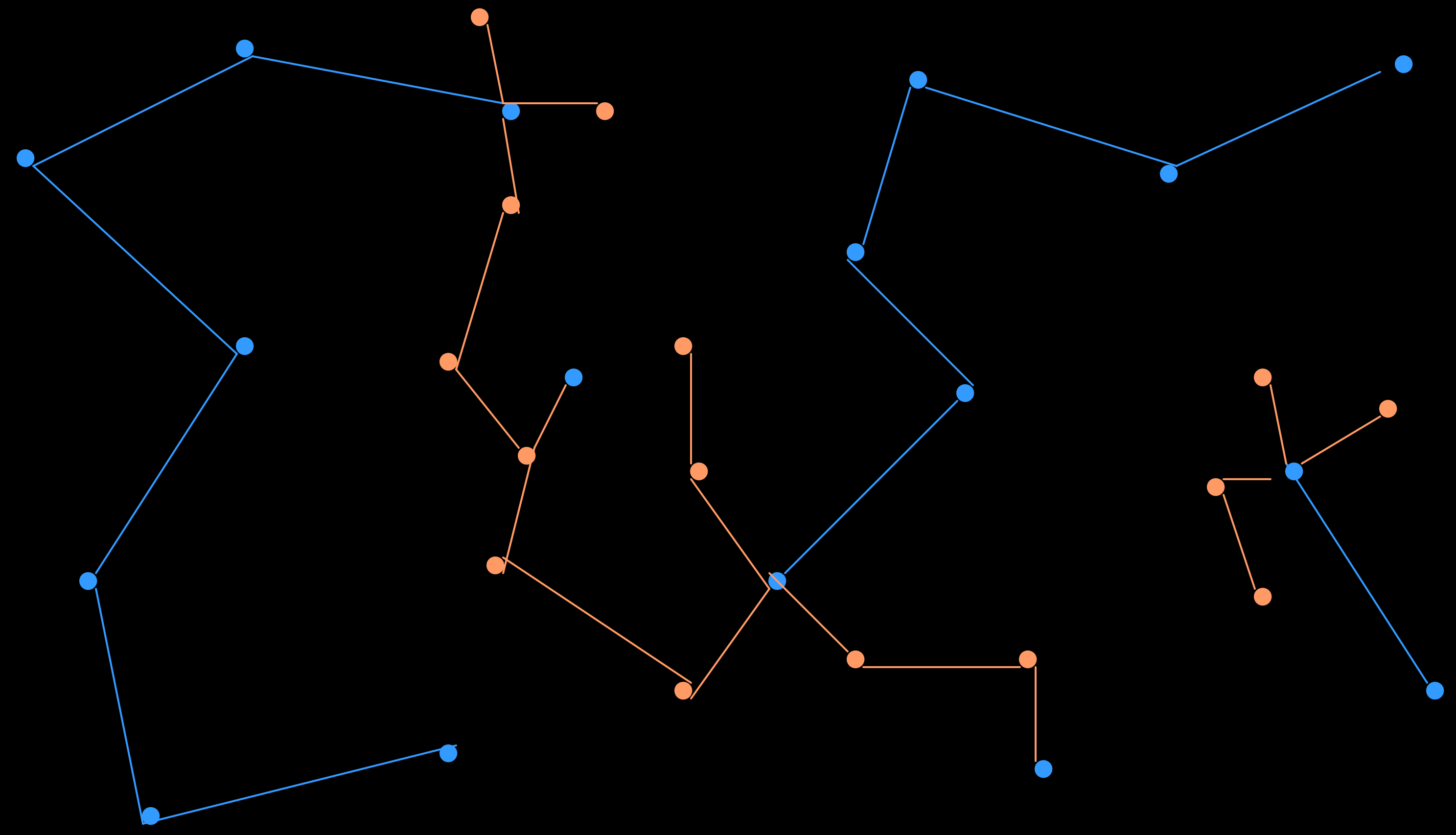
- To expand a node c, we compute a short random-bounce walk starting from c.

  This means
  - Repeatedly pick at random a direction of motion in C-space and move in this direction until an obstacle is hit.
  - When a collision occurs, choose a new random direction.
  - The final configuration n and the edge (c,n) are inserted into R and the path is memorized.
  - Try to connect n to the other connected components like in the construction step.
  - Weights are only computed once at the beginning and not modified as nodes are added to G.

Expansion Step

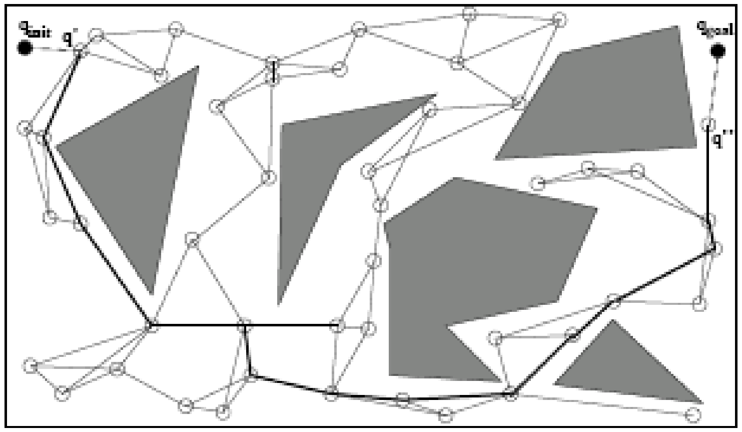


End of Expansion Step



The Query Phase

- Find a path from the start and goal configurations to two nodes of the roadmap
- Search the graph to find a sequence of edges connecting those nodes in the roadmap
- Concatenating the successive segments gives a feasible path for the robot



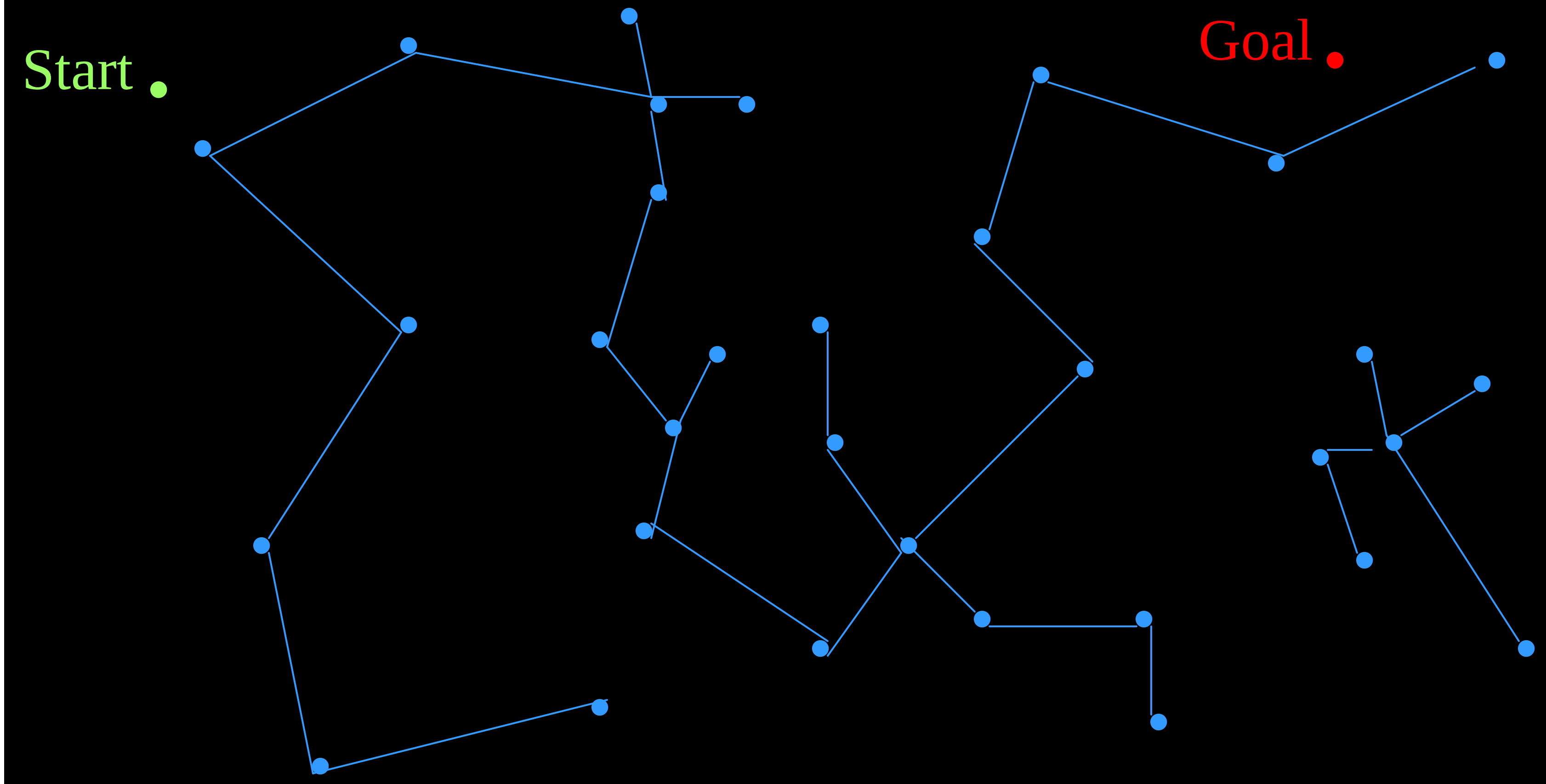
The Query Phase (contd.)

- Let start configuration be s
- Let goal configuration be g

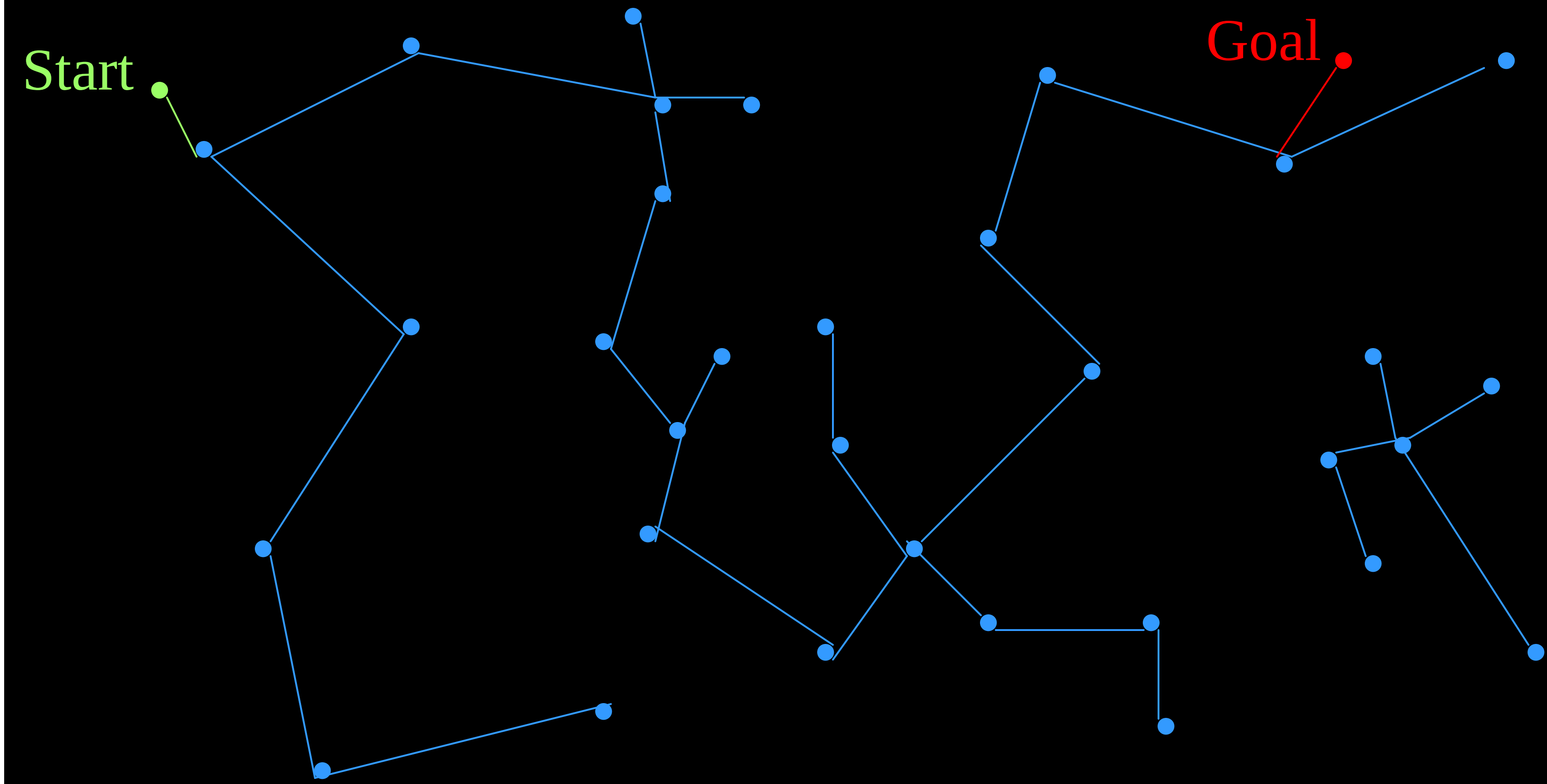
- Try to connect s and g to Roadmap R at two nodes ŝ and ĝ, with feasible paths P_s and P_g. If this fails, the query fails.
  - Consider nodes in G in order of increasing distance from s (according to D) and try to connect s to each of them with the local planner, until one succeeds.
  - Random-bounce walks:

- Compute a path P in R connecting ŝ to ĝ.
- Concatenate P_s ,P and reversed P_g to get the final path.

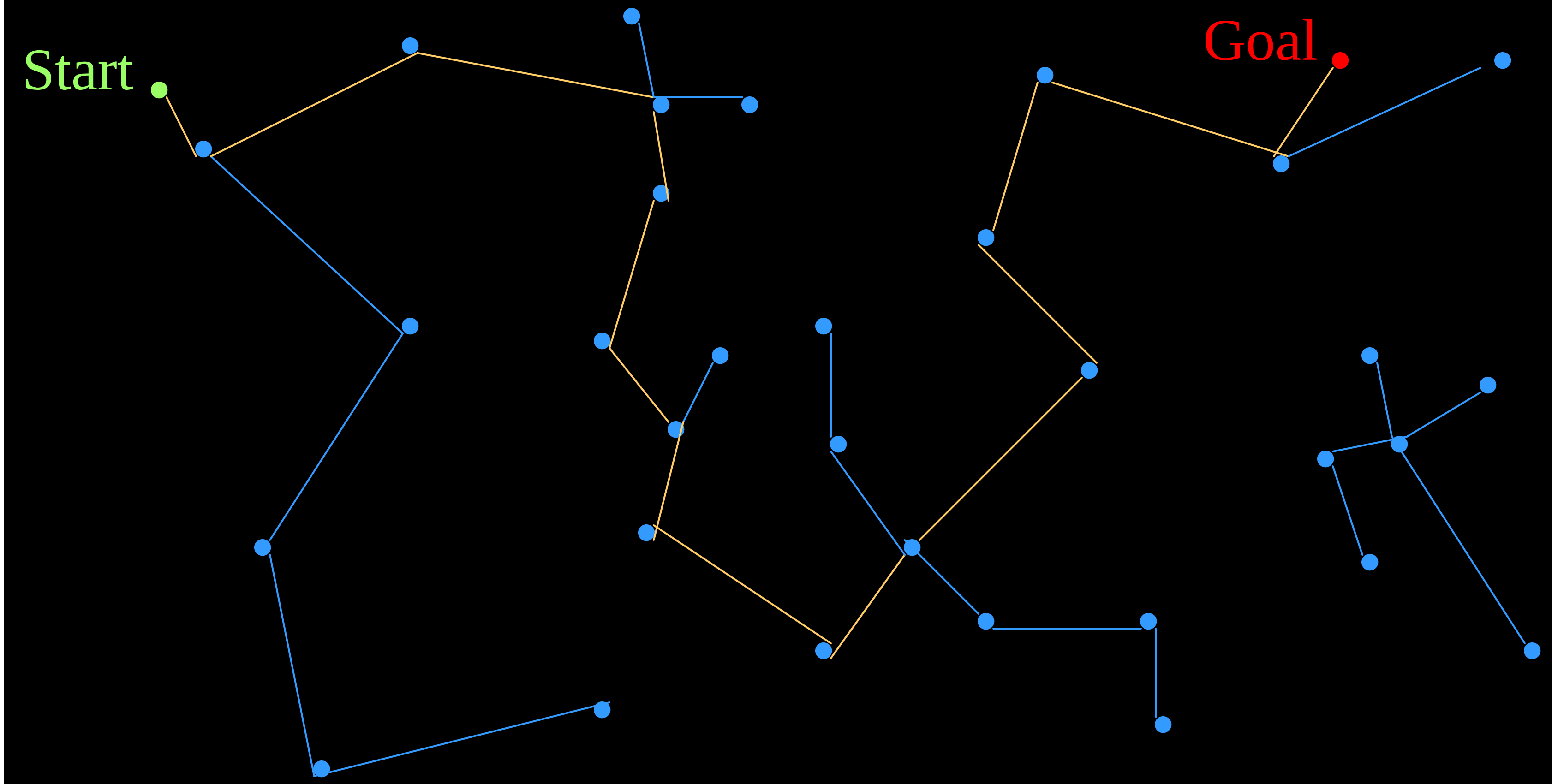
Select start and goal



Connect Start and Goal to Roadmap



Find the Path from Start to Goal

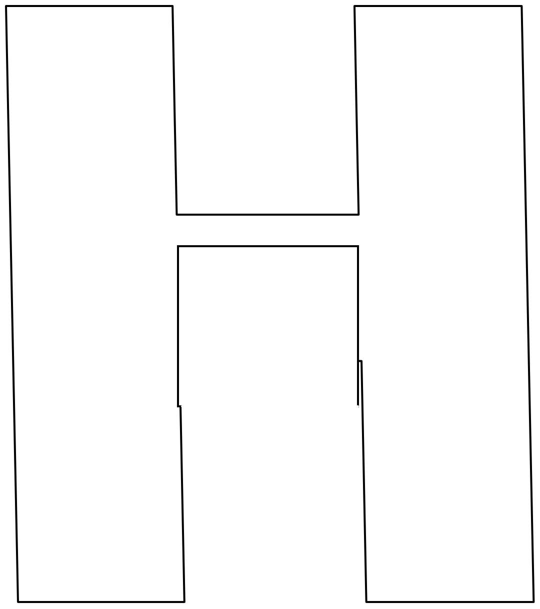


What if we fail?

- Maybe the roadmap was not adequate.
- Could spend more time in the Learning Phase
- Could do another Learning Phase and reuse R constructed in the first Learning Phase. In fact, Learning and Query Phases don't have to be executed sequentially.

Sampling Strategies

Uniform is good because it is easy to implement but is bad because of



- Learning Phase
  - Construction Step
    - Uniform sampling
    - New sampling
  - Expansion Step
    - Uniform around neighbor (local repair)
    - New sampling
- Query Phase

Different Strategies

- Near obstacles
- Narrow passages
- Visibility-based
- Manipulatibility-based
- Quasirandom
- Grid-based

Sample Near Obstacles

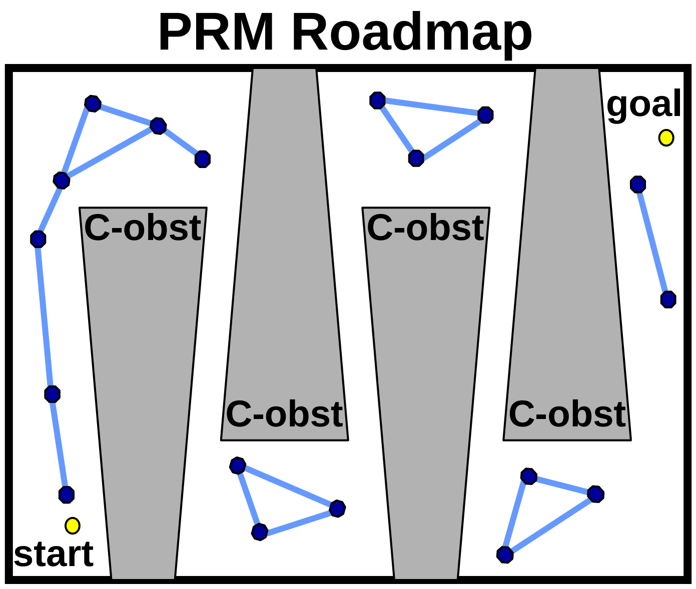
- OBPRM
  - q_{in} found in collision
  - Generate random direction v
  - Find q_{out} in direction v that is free
  - Binary search from q_{in} to obstacle boundary to generate node
- Gaussian sampler
  - Find a q_1
  - Find another q_2 picked from a Guassian distribution centered at q_1
  - If they are both in colision or free, discard. Otherwise, keep the free
- Dilate the space (pushed back via a clever resampling)

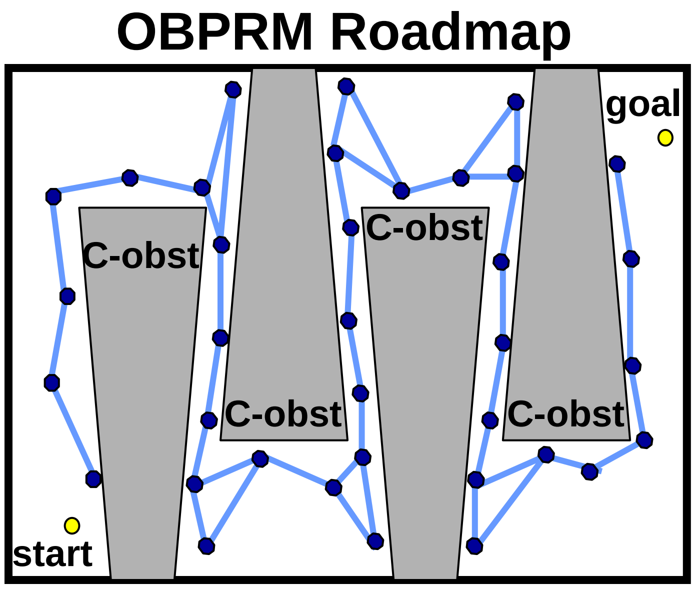
OBPRM: An Obstacle-Based PRM

[IEEE ICRA'96, IEEE ICRA'98, WAFR'98]

To Navigate Narrow Passages we must sample in them

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

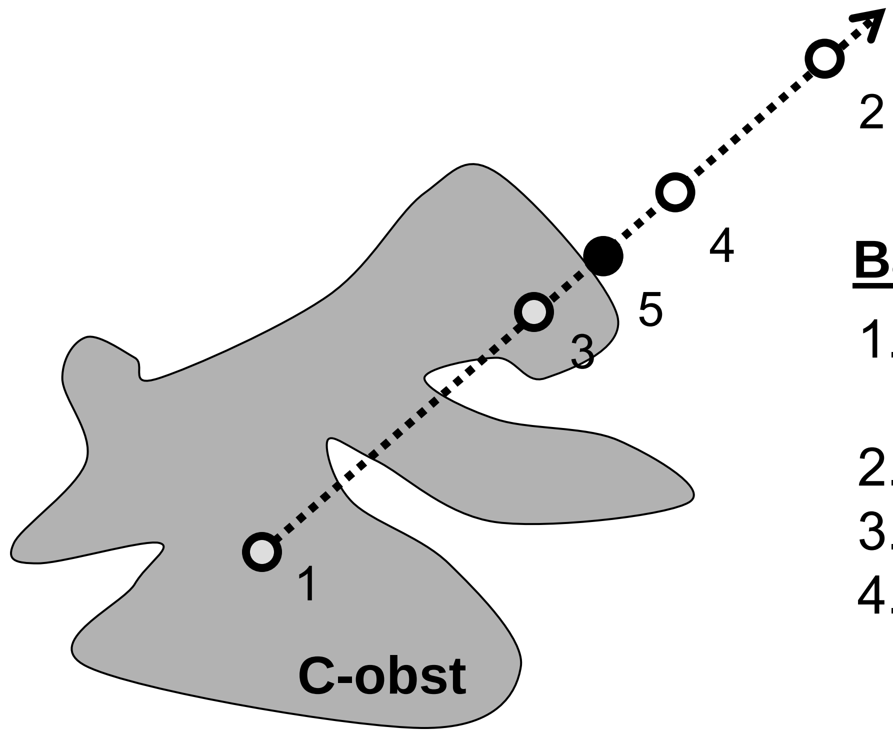




Idea: Can we sample nodes near C-obstacle surfaces?

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

OBPRM: Finding Points on C-obstacles

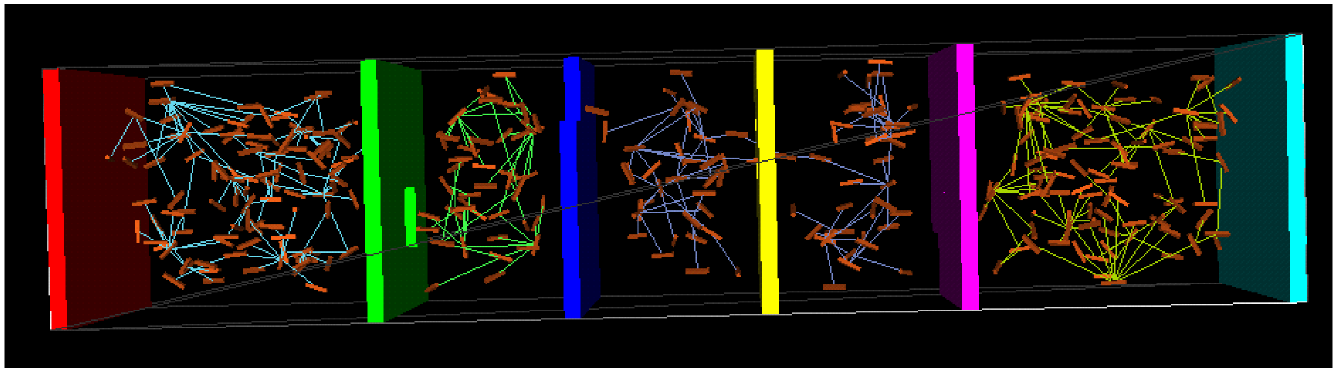


<u>Basic Idea (for workspace obstacle S)</u>

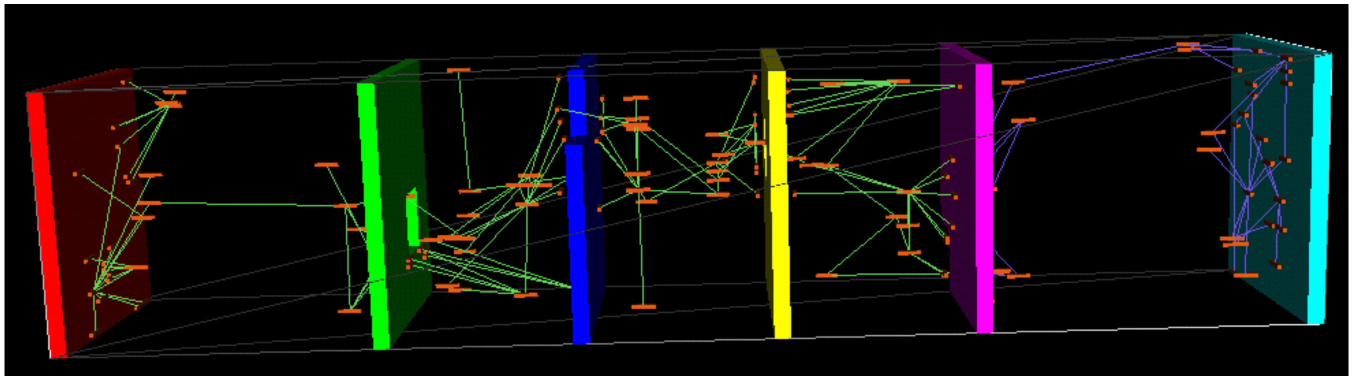
1. Find a point in S's C-obstacle (robot placement colliding with S)
2. Select a random direction in C-space
3. Find a free point in that direction
4. Find boundary point between them using binary search (collision checks)

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

PRM vs OBPRM Roadmaps

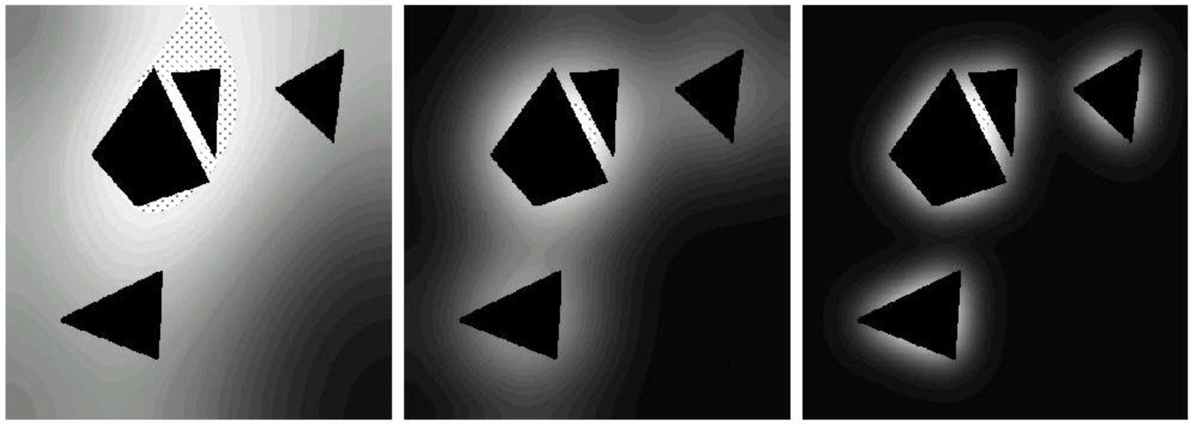


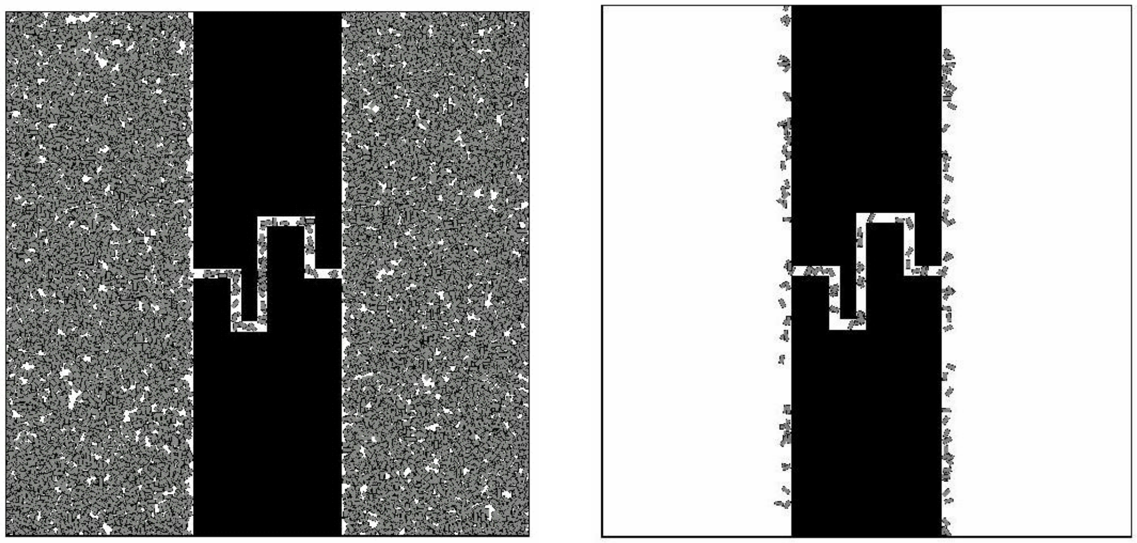
PRM
- 328 nodes
- 4 major CCs



OBPRM
- 161 nodes
- 2 major CCs

Guassians





Sampling inside the Narrow Passageways

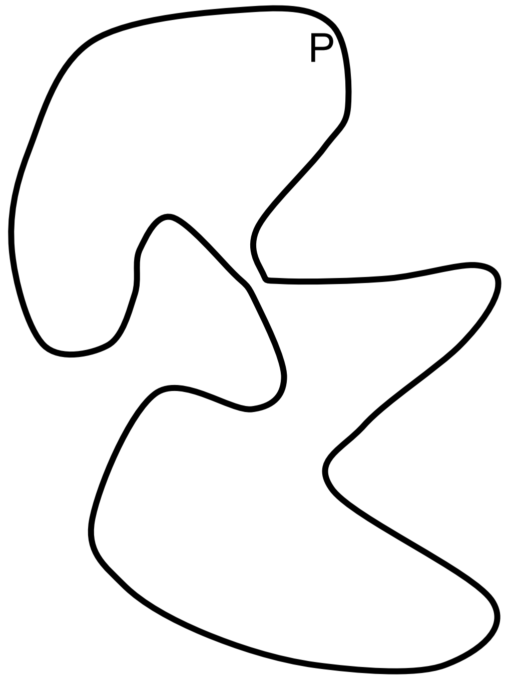
- Bridge Planner
  - q1 and q2 are randomly sampled
  - If they are both in collision, their midpoint is considered
- Dilate
- GVD of Cspace
  - Somehow retract samples onto it without construction
- GVD of Workspace
  - Use knot points or handle points

Dilate

Example:

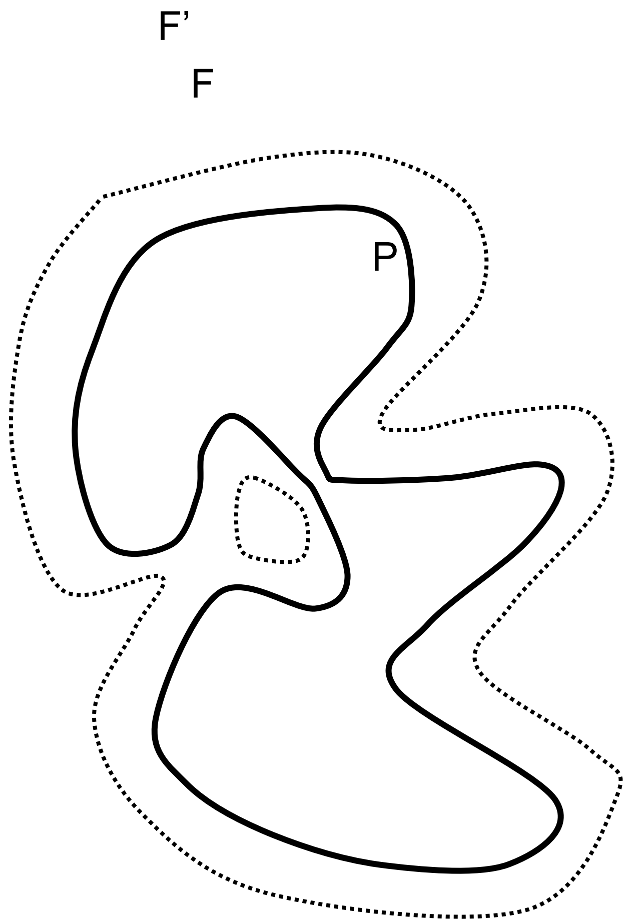
F

F -> Free Space
P -> Narrow Passage



Dilate

Example:

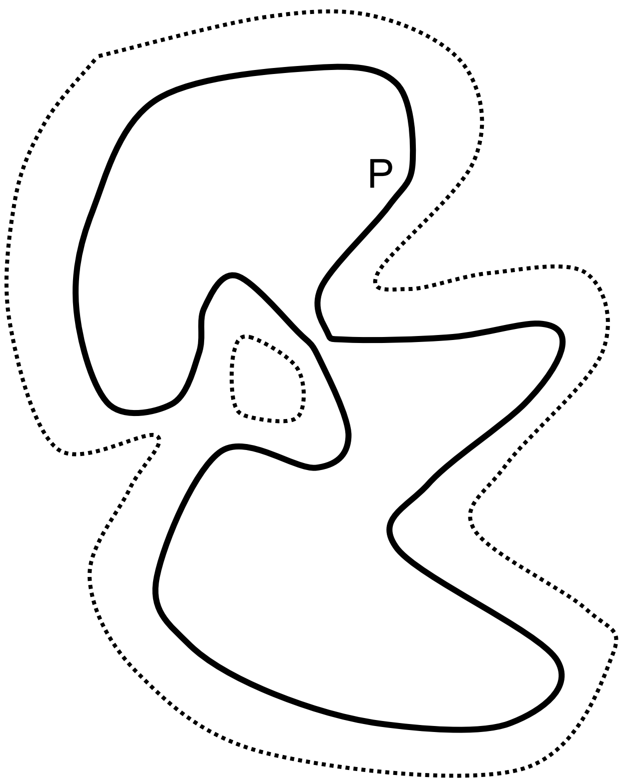


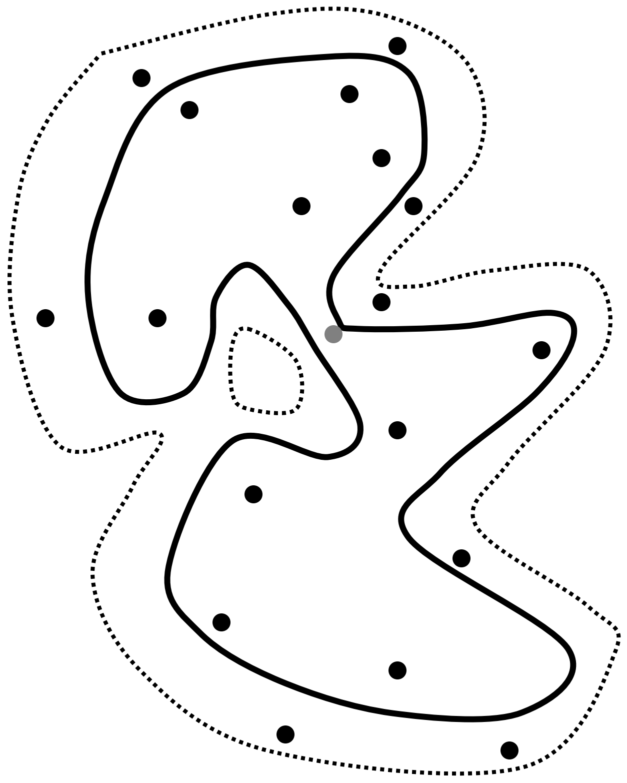
Dilate

Example:

F'

F

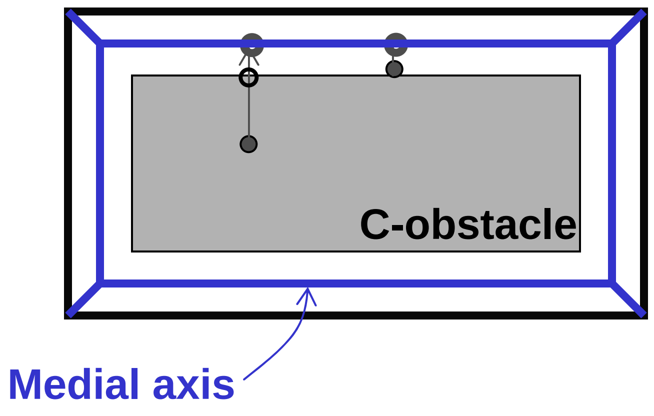




MAPRM: Medial-Axis PRM

Steve Wilmarth, Jyh-Ming Lien, Shawna Thomas [IEEE ICRA'99, ACM SoCG'99, IEEE ICRA'03]

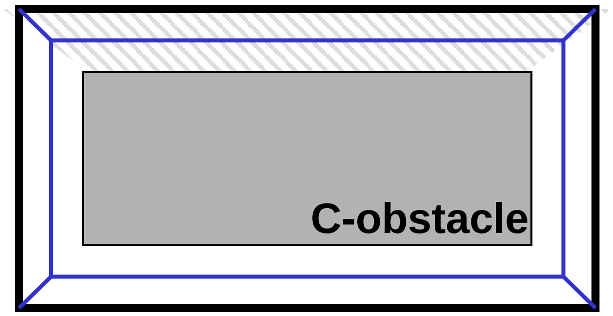
- **Key Observation**: We can efficiently retract almost any configuration, free or not, onto the medial axis of the free space without computing the medial axis explicitly.



MAPRM: Significance

Theorem: *Sampling and retracting can increase the #nodes in narrow corridors in a way that is* <u>*independent of the corridor's volume*</u> *(depends on volume that bounds corridor).*

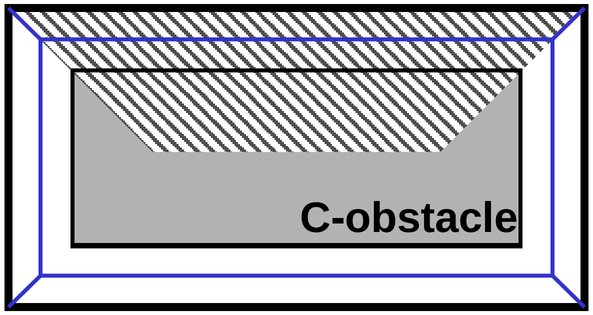
PRM (uniform sampling)



Probability PRM node in corridor

$$\frac{\mu(S)}{\mu(C)}$$

MAPRM



Probability MAPRM node in corridor

$$\frac{\mu(S)+\mu(b_B^{-1}(\partial S))}{\mu(C)}$$

where $b_B : B \setminus \mathrm{MA}(B) \rightarrow \partial B$ maps colliding nodes to closest boundary point

Manipulability Sampling

$$w(q) = \sqrt{\det(J(q)J^T(q))}$$

Quasirandom

- Discrepancy

(Uniformity)

Partition

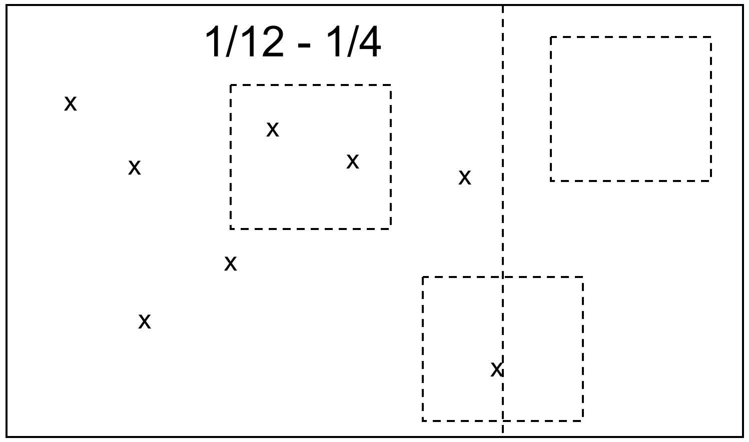
$$D(P, \Re) = \sup_{R \in \Re} \left| \frac{\mu(R)}{\mu(X)} - \frac{|P \cap R|}{N} \right|$$

Samples

Set of partitions

Range Space

Number of points



1/12 - 0

1/4 - 0

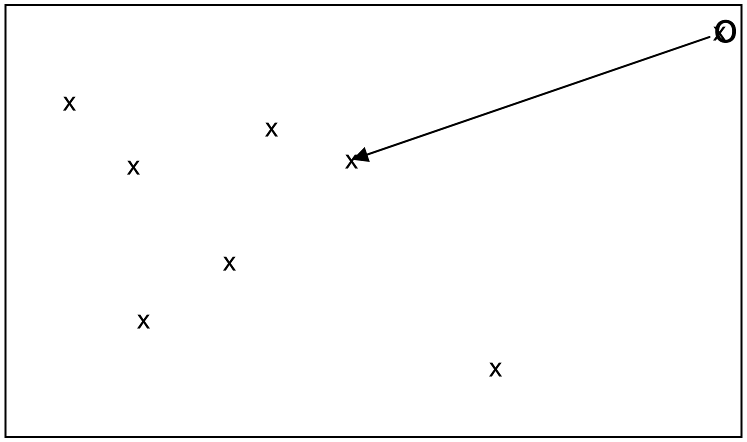
1/12-1/8

–intuitively, the "nonuniformity" of samples

Quasi-Random Sampling

- Consider N points P in a space X and let R $\in \Re$ be some rectangular subset (i.e generalized interval) of X.
- Define dispersion $\delta(P, \Re) = \sup_x \min_p d(x,p)$

–intuitively, the largest empty ball or “unoccupied” space



Sukharev sampling ρ is Linf norm

$$\delta(P,\rho) \geq \frac{1}{2\lfloor N^{\frac{1}{d}} \rfloor}$$

$$\delta* \geq \frac{1}{2\lfloor N^{\frac{1}{d}} \rfloor} \rightarrow N \geq \left(\frac{1}{2\delta*}\right)^d$$

Van der Corput Sequence

- Minimize dispersion and discrepancy

Unit interval, binary number $a_i \in \{0,1\}$

$$n = \sum_i a_i 2^i = a_0 + a_1 2 + a_2 2^2 \cdots$$

$$\Phi(n) = \sum_i a_i 2^{-(i+1)} = a_0 2^{-1} + a_1 2^{-2} \cdots$$

n	n (binary)	$\Phi(n)$ (binary)	$\Phi(n)$
0	0	0.0	0
1	1	0.1	1/2
2	10	0.01	1/4
3	11	0.11	3/4
4	100	0.001	1/8
5	101	0.101	5/8
6	110	0.011	3/8
7	111	0.111	7/8
8	1000	0.0001	1/16
9	1001	0.1001	9/16
10	1010	0.0101	5/16
11	1011	0.1101	13/16
12	1100	0.0011	3/16
13	1101	0.1011	11/16
14	1110	0.0111	7/16
15	1111	0.1111	15/16

Haltan Sequence (higher dims)

$$n = \sum_i a_{ij} b_j^i, \quad a_{ij} \in \{0, 1 \cdots b_j\}$$

$$\Phi_{b_j}(n) = \sum a_{ij} b_j^{-(i+1)}.$$

$$p_n \quad = \quad (\Phi_{b_1}(n), \Phi_{b_2}(n), \cdots \Phi_{b_d}(n))$$

R is Axis aligned subsets of X

$$D(P, \mathcal{R}) \leq O\left(\frac{\log^d N}{N}\right).$$

n	$\Phi_2(n)$	$\Phi_1(n)$
0	0	0
1	1/3	1/2
2	2/3	1/4
3	1/9	3/4
4	4/9	1/8
5	7/9	5/8
6	2/9	3/8
7	5/9	7/8
8	8/9	1/16
9	1/27	9/16
10	10/27	5/16
11	19/27	13/16
12	4/27	3/16
13	13/27	11/16
14	22/27	7/16
15	7/27	15/16

Theoretical Analysis Overview

- Analyze a simple PRM model and attempt to find factors which affect and control the performance of the PRM.
- Define ε-goodness
- β-Lookout
- $(\varepsilon, \alpha, \beta)$–expansive space.

- Derive more relations linking probability of failure to controlling factors
- Finding Narrow passages with PRM

The Simplified Probabilistic Roadmap Planner (s-PRM)

The parameters of our model are:

- Free Space Q_{free}: An arbitrary open subset of the unit square W=[0,1]^d
- The Robot: A point free to move in Q_{free}
- The Local Connector: It takes the robot from point a to point b along a straight line and succeeds if the straight line segment *ab* is contained in Q_{free}
- The collection of Random Configurations:
  Collection of N independent points uniform in Q_{free}

Analysis of PRM (s-PRM Probability of Failure)

- 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

We assume that they can be connected by a continuous path γ such that

$\gamma : [0 : L] \to Q_{free}$ where $\gamma(0) = a$ and $\gamma(L) = b$

We will try to find upper bounds for the probability of failure to find such a path γ between a and b when we assume

a) minimum distance from obstacles
b) varying (mean) distance from obstacles

Theorem 1

(Upper Bound Involving Minimum Distance)

Let $\gamma : [0 : L] \to Q_{free}$ be a path of (Euclidean) length L.

Then the probability that s-PRM will ***fails*** to connect the points a and b is at most

$$\frac{2L}{R}(1 - \alpha R^2)^N$$

Where $\alpha = \frac{\pi}{4|Q_{free}|}$ is a constant.

Here, we assume *R* to be the minimum distance the obstacles.

Analysis of PRM

- A path from a to b can be described by a function γ: $[0,1] \rightarrow Q_{\text{free}}$.
- Let clr(γ) be the minimum distance between γ and any obstacle.
- Let μ be a volume measure on the space $\mu([0,1]^d) = 1$

$\mu(Q_{\text{free}}) <= 1$

- $B_\delta(x)$ is a ball centered at x of radius δ
- For uniform sampling of $A \subset Q_{\text{free}}$

$$Pr(x \in A) = \frac{\mu(A)}{\mu(Q_{\text{free}})}$$

Analysis of PRM

- Theorem: Let a b be a pair in Q_{free} s.t. there is a path γ lying in Q_{free}. The probability that PRM answers the query correctly after n configurations is

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

where L is the length of the path γ,

$$\rho = \texttt{clr}(\gamma)$$

$$\sigma = \frac{\mu(B_1(\cdot))}{2^d \mu(Q_{\text{free}})}$$

Proof

$\rho = \texttt{clr}(\gamma)$ and note that $\rho > 0$

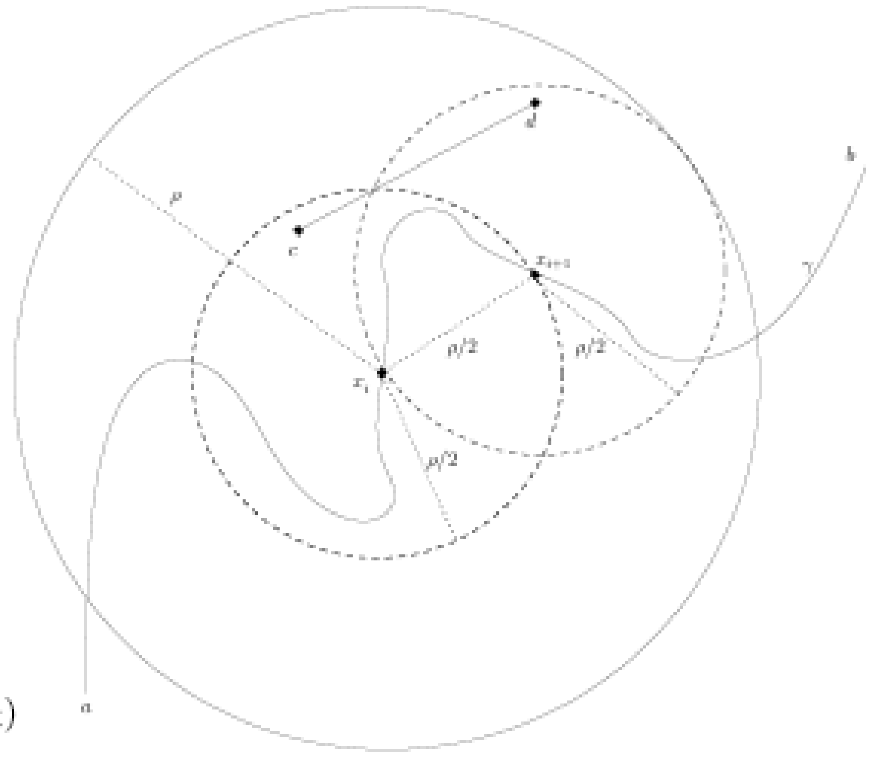
$m = \left\lceil \frac{2L}{\rho} \right\rceil$

there are m points on the path $a = x_1, \ldots, x_m = b$
such that $dist(x_i, x_{i+1}) < \rho/2$

$y_i \in B_{\rho/2}(x_i)$ and $y_{i+1} \in B_{\rho/2}(x_{i+1})$.

Let $y_i \in B_{\rho/2}(x_i)$ and $y_{i+1} \in B_{\rho/2}(x_{i+1})$

$\overline{y_i y_{i+1}}$ must lie inside Q_{free} since both in the ball $B_\rho(x_i)$



Points c and d are inside the $\rho/2$ balls and straight-line $\overline{cd}$ is in Q_{free}

Analysis of PRM

- Suppose we generate n samples uniformly. If there is a subset $y_1 \ldots y_m$ s.t. $y_i \in B_{\rho/2}(x_i)$ then we have a path
- Let I_i represent the event that there is a y in $B_{\rho/2}(x_i)$

$$Pr[(a,b)\,\text{FAILURE}] \le Pr\left(\bigvee_{i=1}^{m} I_i = 0\right) \le \sum_{i=1}^{m} Pr[I_i = 0]$$

- Probability of a point falling into a ball is $\mu(B_{\rho/2}(x_i))/\mu(Q_{\text{free}})$
- Probability of none of the n samples falling itno a ball is

$$Pr[I_i = 0] = \left(1 - \frac{\mu(B_{\rho/2}(x_i))}{\mu(Q_{\text{free}})}\right)^n.$$

-
- Thus, $$Pr[(a,b)\,\text{FAILURE}] \le \left\lceil \frac{2L}{\rho} \right\rceil \left(1 - \frac{\mu(B_{\rho/2}(\cdot))}{\mu(Q_{\text{free}})}\right)^n$$
- But, $$\frac{\mu(B_{\rho/2}(\cdot))}{\mu(Q_{\text{free}})} = \frac{\left(\frac{\rho}{2}\right)^d \mu(B_1(\cdot))}{\mu(Q_{\text{free}})} = \sigma\rho^d$$
- Finally $(1-\beta)^n \le e^{-\beta n}$ for $0 \le \beta \le 1$ $$Pr[(a,b)\,\text{FAILURE}] \le \left\lceil \frac{2L}{\rho} \right\rceil e^{-\sigma\rho^d n}.$$

What the results tell us

The results give us an idea of what factors control failure:

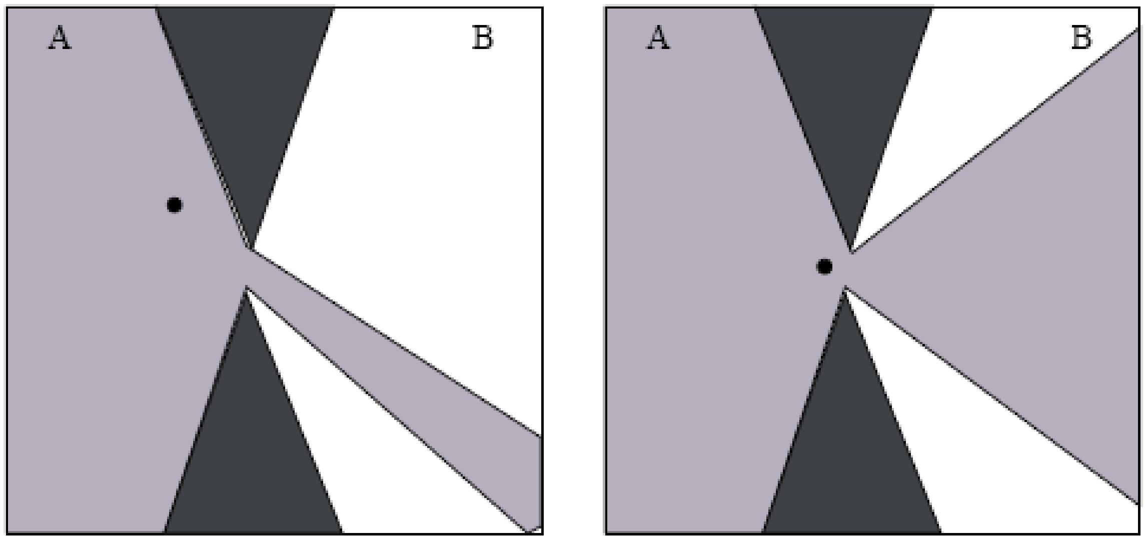
- Dependence on N is exponential
- Dependence on L is linear
- Tweaking these factors can give us desired success rate
- PRM avoids creating large number of nodes in
  areas where connections are obtained easily

However,

- The bounds computed here are not very easy to use as they depend on the properties of postulated connecting path $\gamma(t)$ from a to b (difficult to measure a-priori)

(ε,α,β)–Expansiveness

$$\texttt{reach}(S) = \{x \in \mathcal{Q}_{\text{free}} \mid \exists y \in S \textit{ such that } \overline{xy} \subset \mathcal{Q}_{\text{free}}\}.$$



A space $\mathcal{Q}_{\text{free}}$ *is* ϵ*-good if* $\mu(\texttt{reach}(x)) \geq \epsilon\mu(\mathcal{Q}_{\text{free}})$ *for all* $x \in \mathcal{Q}_{\text{free}}$

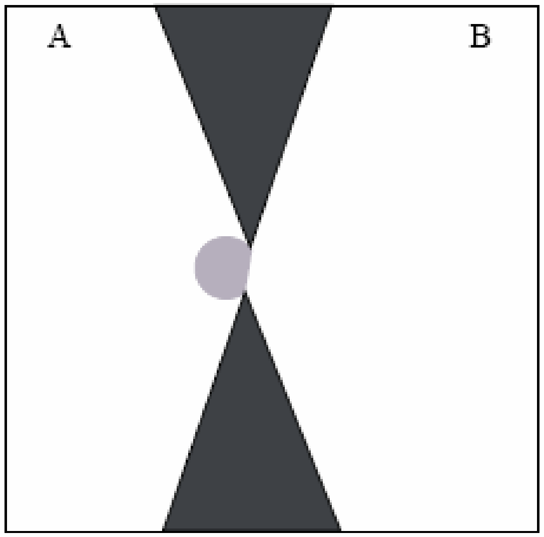
Definitions (contd.)

β-*LOOKOUT*

Let β be a constant in [0,1] and *S* be a subset of a component Q_{freei} of the free space Q_{free}

The β-*LOOKOUT* of *S* is the set

$$\beta - Lookout(S) = \{ q \in S \mid \mu(reach(q) \setminus S) \geq \beta \times \mu(Q_{freei} \setminus S) \}$$



A is the set and β is large

Points in S that can see a β fraction of points not in S over all points not in S

Expansive Spaces

Let ε, α, β be constants in [0, 1]

The free space is (ε, α, β) expansive if it is ε-good and, for every connected subset S, we have

Ensures that a certain fraction of the free configuration space is visible from any point in the free configuration space

$$\mu(\beta - LOOKOUT\ (S)) \geq \alpha \times \mu(S)$$

Each subset of the free configuration space has a large lookout

We can abbreviate “(ε, α, β) expansive” by simply “expansive”

Consider a set of samples V

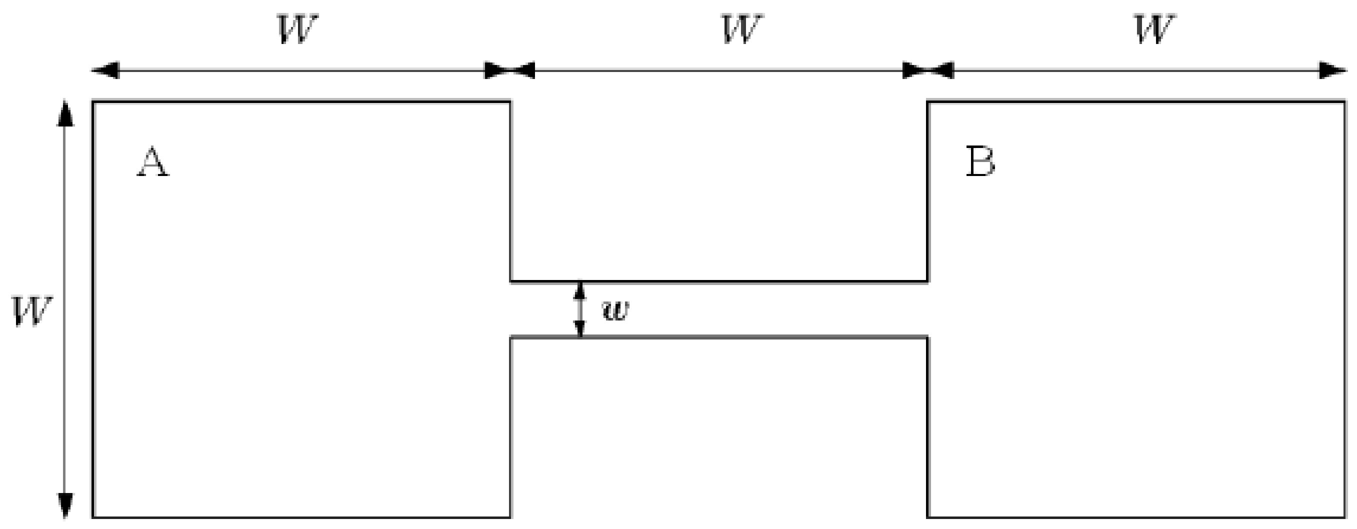
S is like the union of the reachable set from points V

Large values of α and β mean that picking points from S and adding them to V expands S a lot

Example

$\mu(\beta - LOOKOUT\ (S)) \geq \alpha \times \mu(S)$

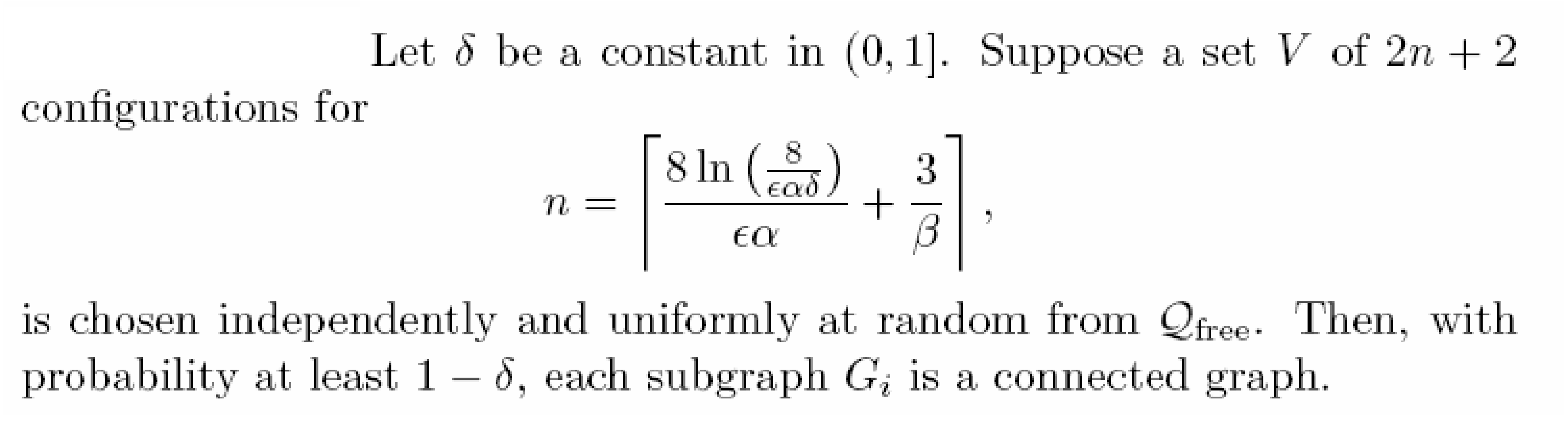
An expansive free space where ε, α. β ~ *w/W*



Points with smallest ε are inside narrow corridor and can only see approximately 3w/W

Pick a point in the upper right corner of A. It can see WW
Only a subset of A, the inner slab of area wW, can see a 2wW of the remaining WW + Ww area

Result



Let δ be a constant in $(0,1]$. Suppose a set V of $2n+2$ configurations for

$$n = \left\lceil \frac{8\ln\left(\frac{8}{\epsilon\alpha\delta}\right)}{\epsilon\alpha} + \frac{3}{\beta} \right\rceil,$$

is chosen independently and uniformly at random from $\mathcal{Q}_{\text{free}}$. Then, with probability at least $1-\delta$, each subgraph G_i is a connected graph.

Stop here

- Skipped some of Greg’s slides from before

Roadmap Coverage

Assume that F is *♏ - good.* Let ♐ be a constant in [0,1]
Let k is a positive real such that for any x ♏ [0,1]

If N is chosen such that $(1-x)^{(k/x)\log(2/x\phi)} \leq x\phi/2$

Then the roadmap generates a set of milestones that adequately covers *F*, with probability at least *1- ♐*

This still does not allow us to compute N.

$$N \geq \frac{K}{\in}(\log\frac{1}{\in} + \log\frac{2}{\phi})$$

However, it tells us that the although adequate coverage is is not guaranteed, the probability that this happens

Decreases exponentially with the number of milestones N.

Roadmap Connectivity

Assume that F is (♏, ♋, ♌) expansive .
Let ☒ be a constant in [0,1]. If N is chosen such that

Then with the probability 1 - ☒, the roadmap generates a roadmap such that no two of its components lie on the same component of F.

This tells us that the probability that a roadmap does not adequately represent F decreases exponentially with the number of milestones N. Also, number of milestones needed increases moderately when ♏, ♋ and ♌ increase.

$$N \geq \frac{16}{\varepsilon\alpha}\log\frac{8}{\varepsilon\alpha\xi}+\frac{6}{\beta}+4$$

Rapidly-Exploring Random Trees (RRTs) [Kuffner, Lavalle]

The Basic RRT
- single tree
- bidirectional
- multiple trees (forests)

RRTs with Differential Constraints
- nonholonomic
- kinodynamic systems
- closed chains

Some Observations and Analysis
- number of branches
- uniform convergence
- resolution completeness
- leaf nodes vs. interior nodes

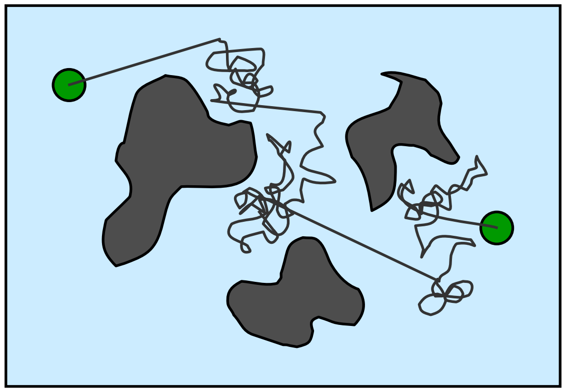
Performance & Implementation Issues
- Metrics and Metric sensitivity
- Nearest neighbors
- Collision Checking
- Choosing appropriate step sizes

High-Dimensional Planning as of 1999

Single-Query:

Barraquand, Latombe '89; Mazer, Talbi, Ahuactzin, Bessiere '92; Hsu, Latombe, Motwani '97; Vallejo, Jones, Amato '99;

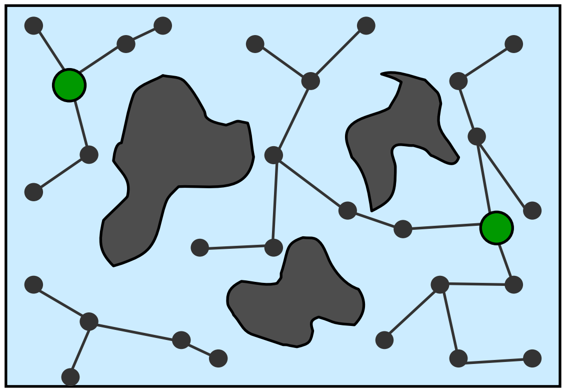
EXAMPLE: Potential-Field



Multiple-Query:

Kavraki, Svestka, Latombe, Overmars '95; Amato, Wu '96; Simeon, Laumound, Nissoux '99; Boor, Overmars, van der Stappen '99;

EXAMPLE: PRM

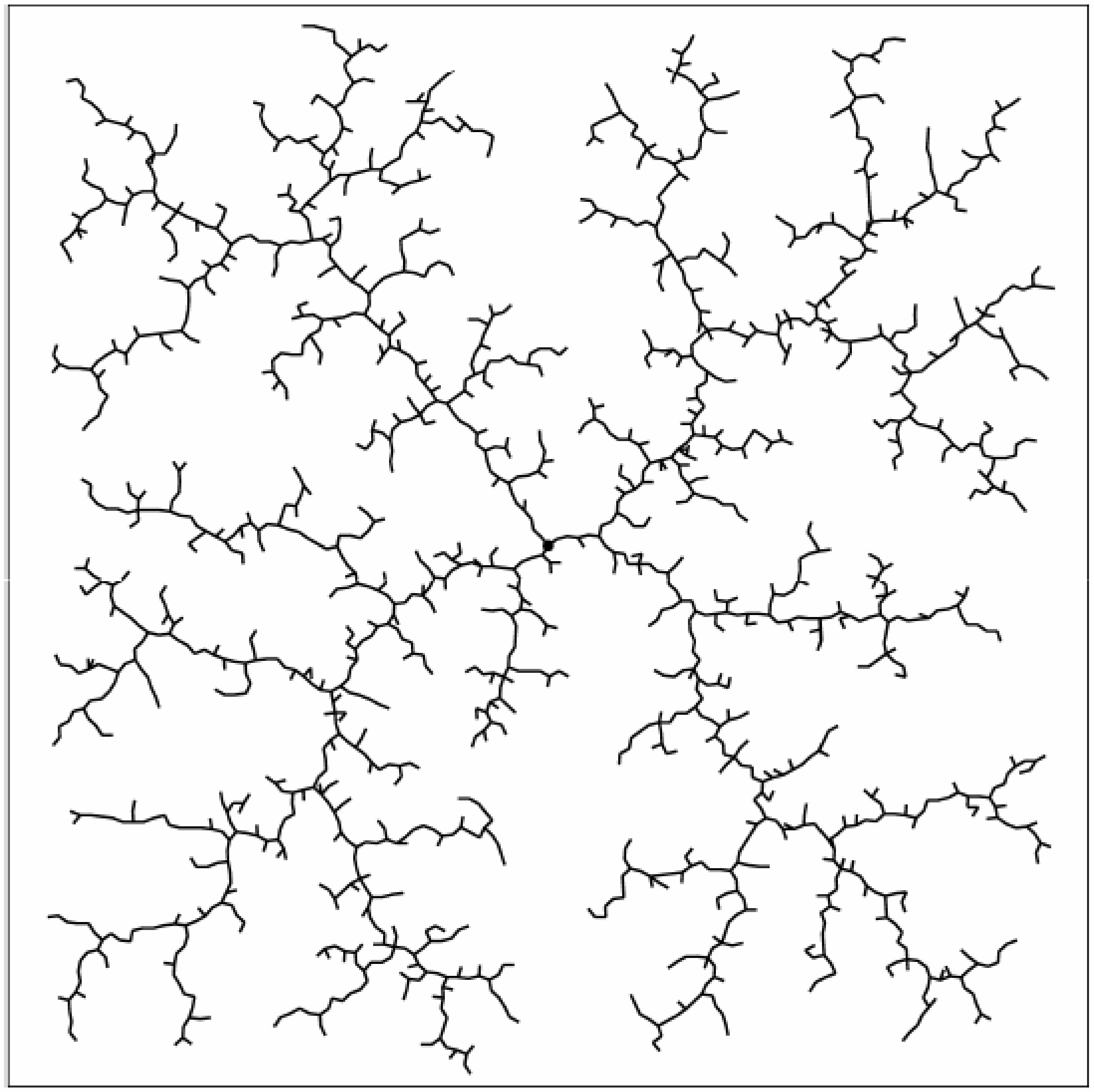


<u>TENSION</u>

Greedy, can take a long time but good when you can dive into the solution

Spreads out like uniformity but need lots of sample to cover space

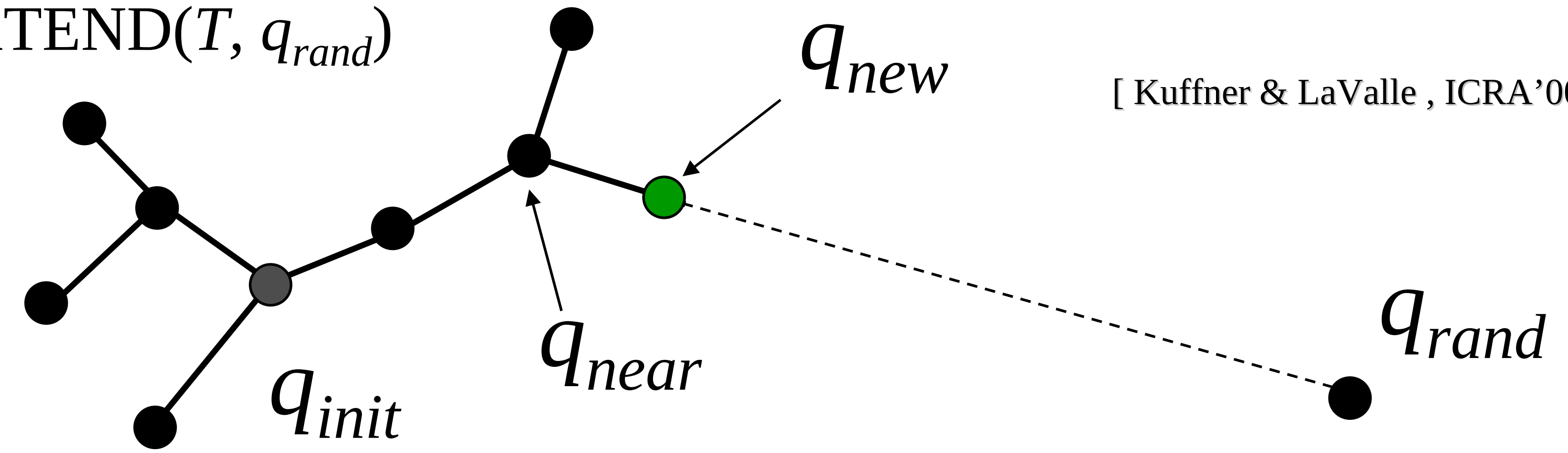
Rapidly-Exploring Random Tree



Path Planning with RRTs (Rapidly-Exploring Random Trees)

```
BUILD_RRT ($q_{init}$) {
  $T$.init($q_{init}$);
  for $k$ = 1 to K do
    $q_{rand}$ = RANDOM_CONFIG();
    EXTEND($T$, $q_{rand}$)
}
```

EXTEND(T, q_{rand})



[Kuffner & LaValle , ICRA’00]

Path Planning with RRTs (Some Details)

```
BUILD_RRT ($q_{init}$)  {
    $T$.init($q_{init}$);
    for $k$ =  1 to K do
        $q_{rand}$ = RANDOM_CONFIG();
        EXTEND($T$, $q_{rand}$)
}
```

STEP_LENGTH: How far to sample

1. Sample just at end point
2. Sample all along
3. Small Step

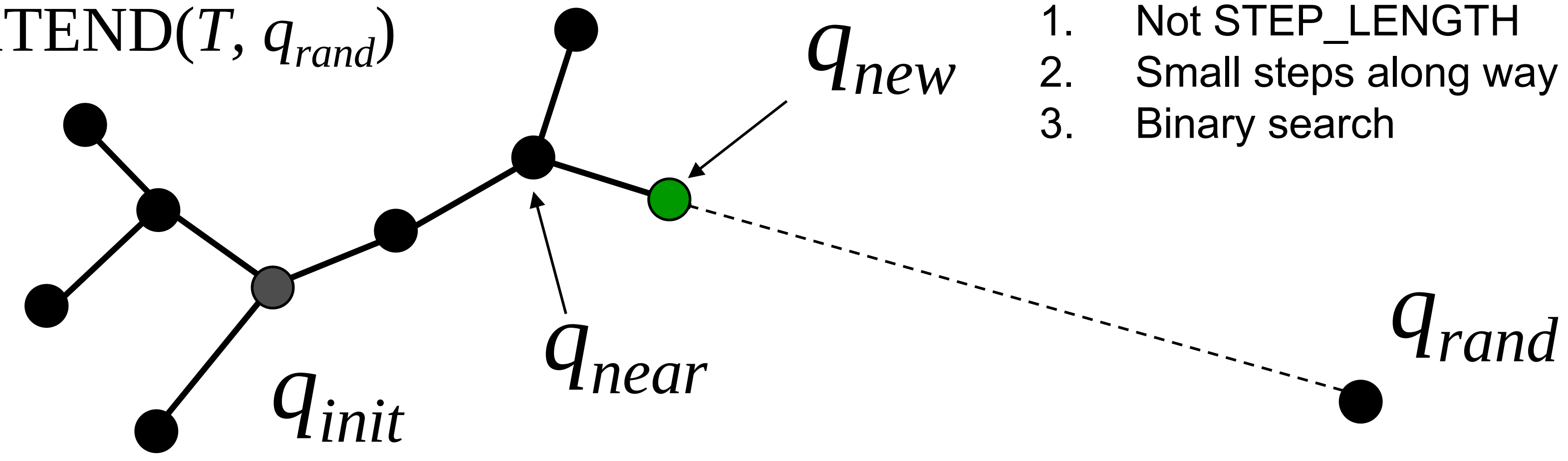
Extend returns

1. Trapped, cant make it
2. Extended, steps toward node
3. Reached, connects to node

STEP_SIZE

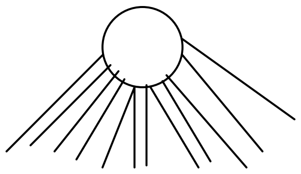
1. Not STEP_LENGTH
2. Small steps along way
3. Binary search

EXTEND(T, q_{rand})

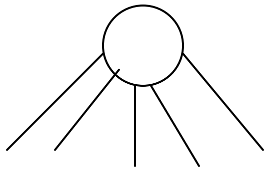


RRT vs. Exhaustive Search

- Discrete

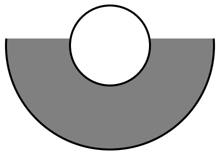


A* may try all edges

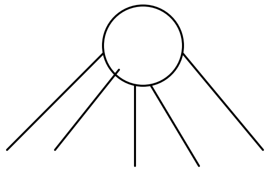


Probabilistically subsample all edges

- Continuous



Continuum of choices



Probabilistically subsample all edges

Naïve Random Tree

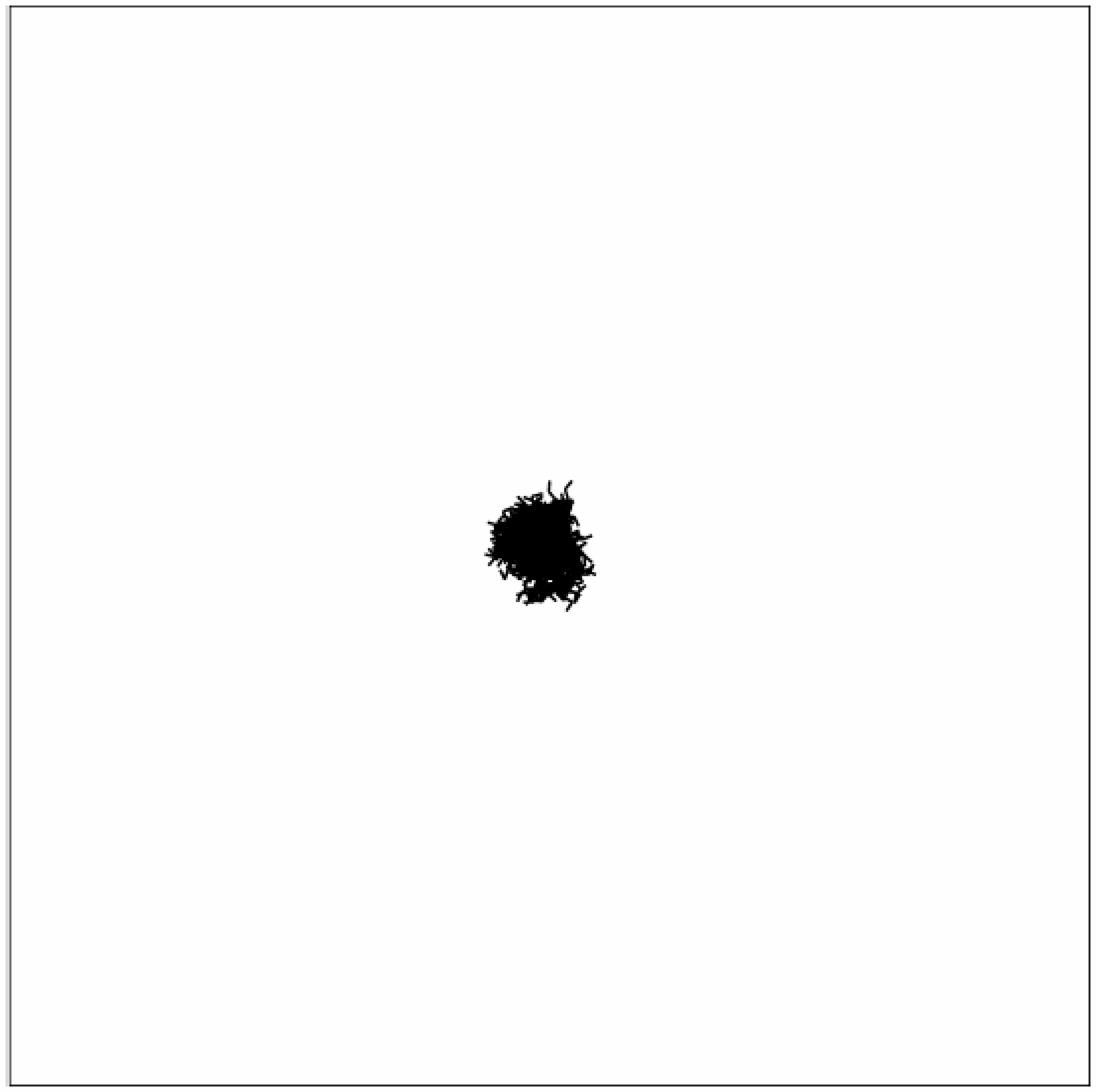
Start with middle

Sample near this node

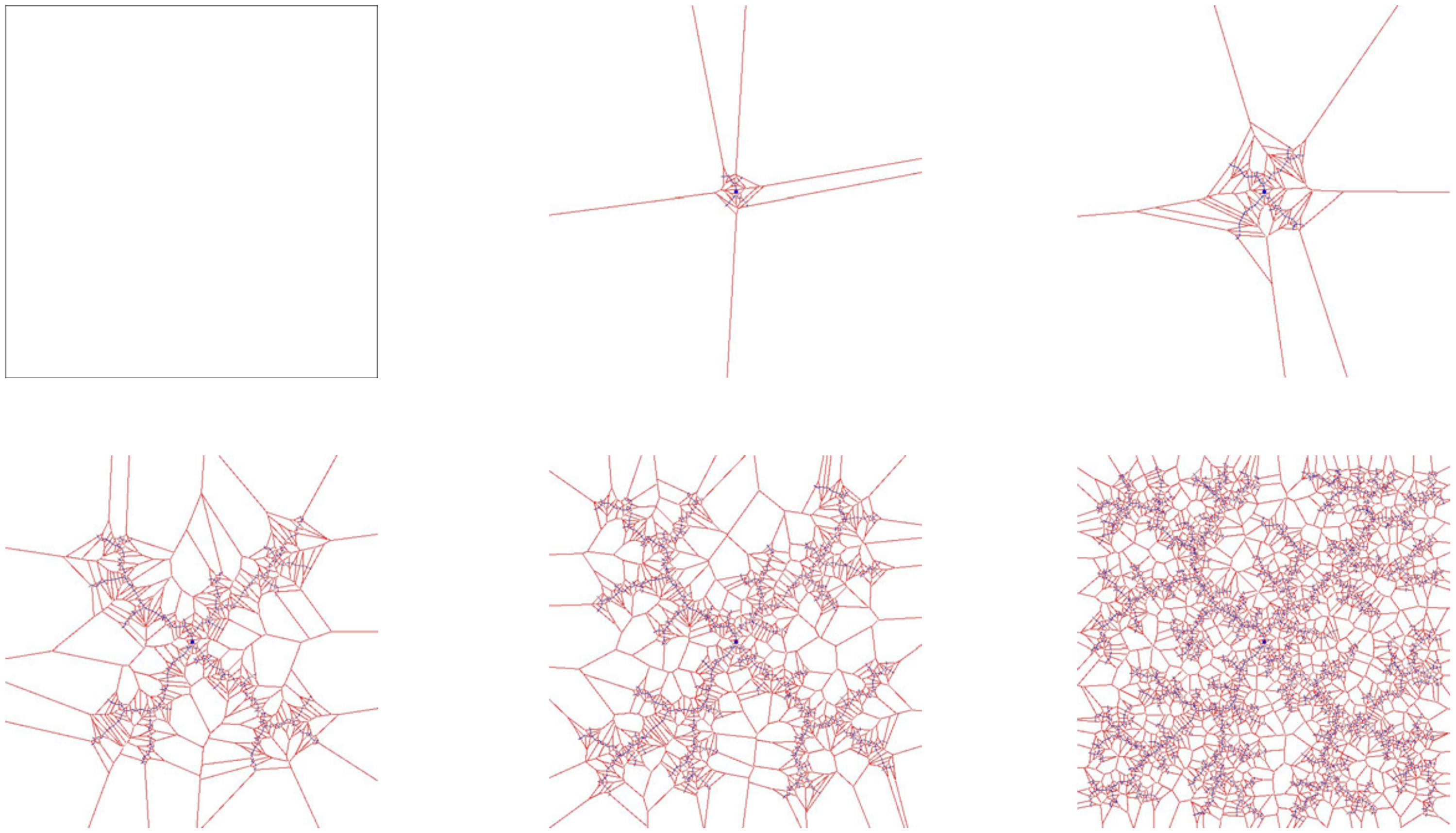
Then pick a node at random in tree

Sample near it

End up Staying in middle



RRTs and Bias toward large Voronoi regions

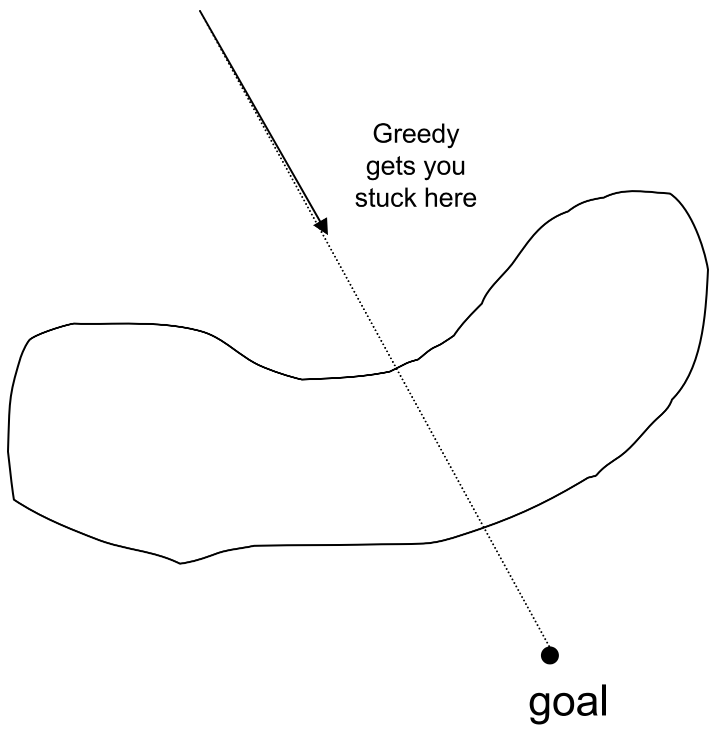


http://msl.cs.uiuc.edu/rrt/gallery.html

Biases

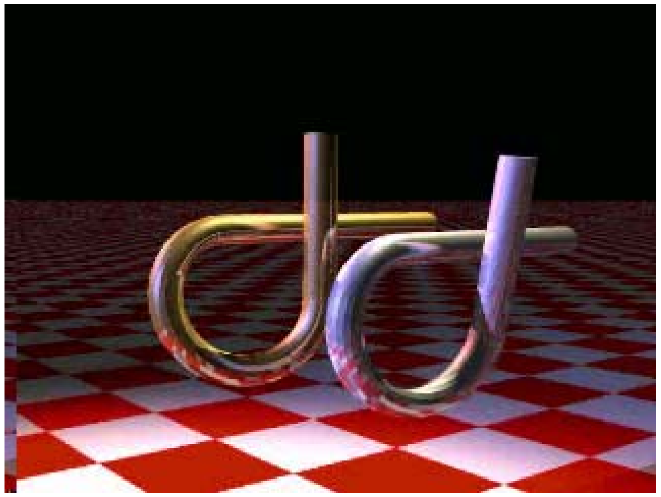
- Bias toward larger spaces
- Bias toward goal
  - When generating a random sample, with some probability pick the goal instead of a random node when expanding
  - This introduces another parameter
  - James' experience is that 5-10% is the right choice
  - If you do this 100%, then this is a RPP

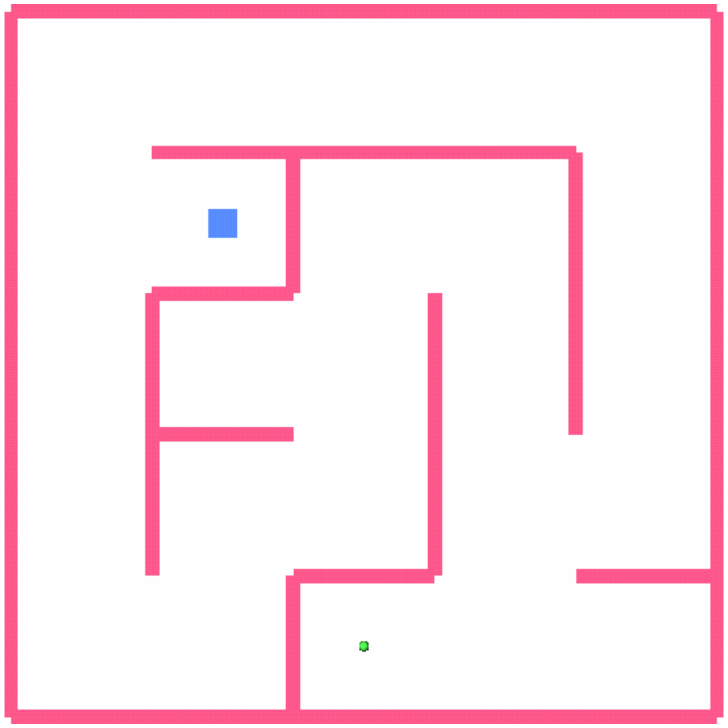
RRT vs. RPP



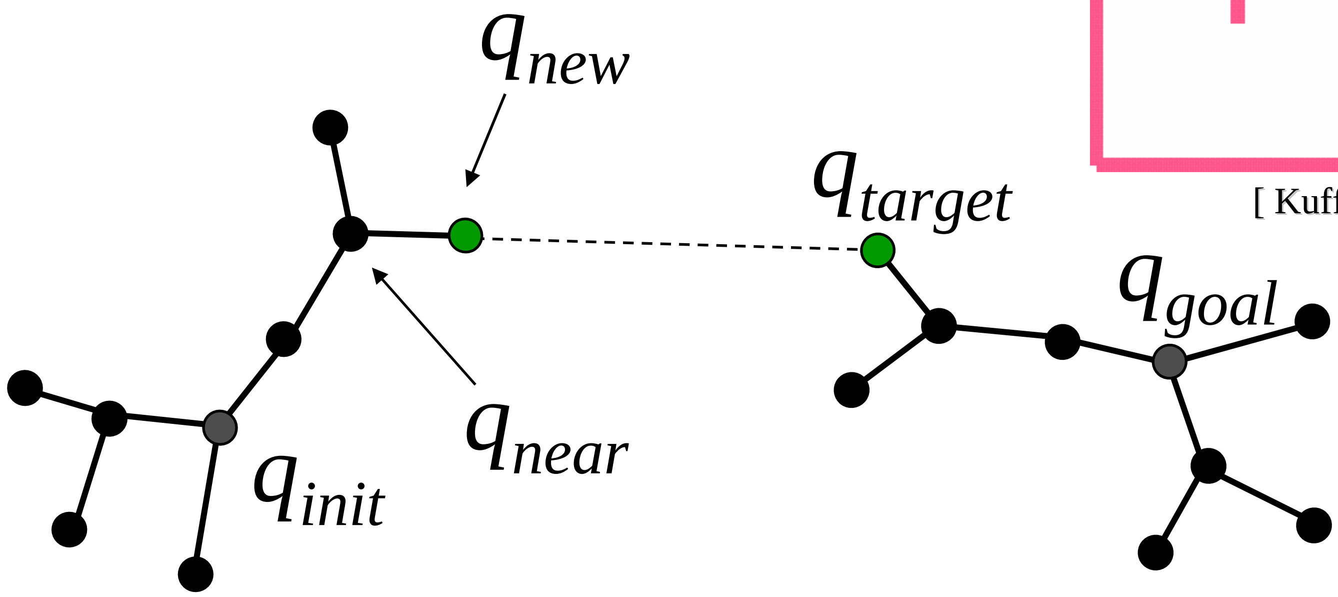
RRT's will pull away and better approximate cost-to-go

Grow two RRTs towards each other

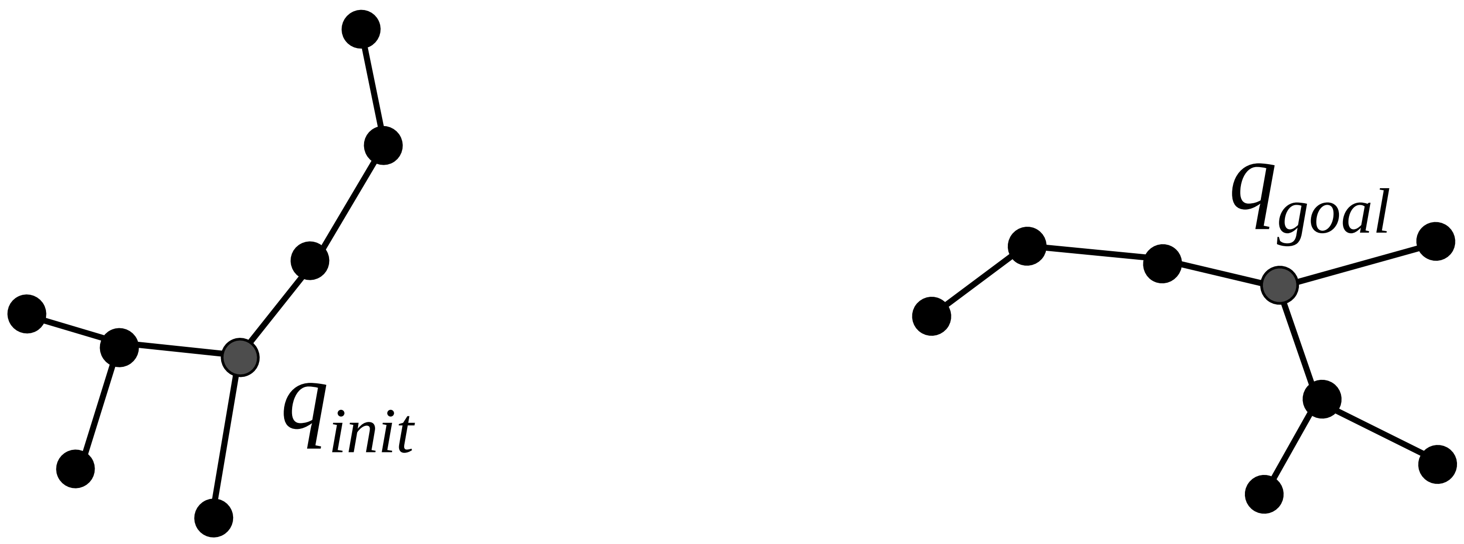




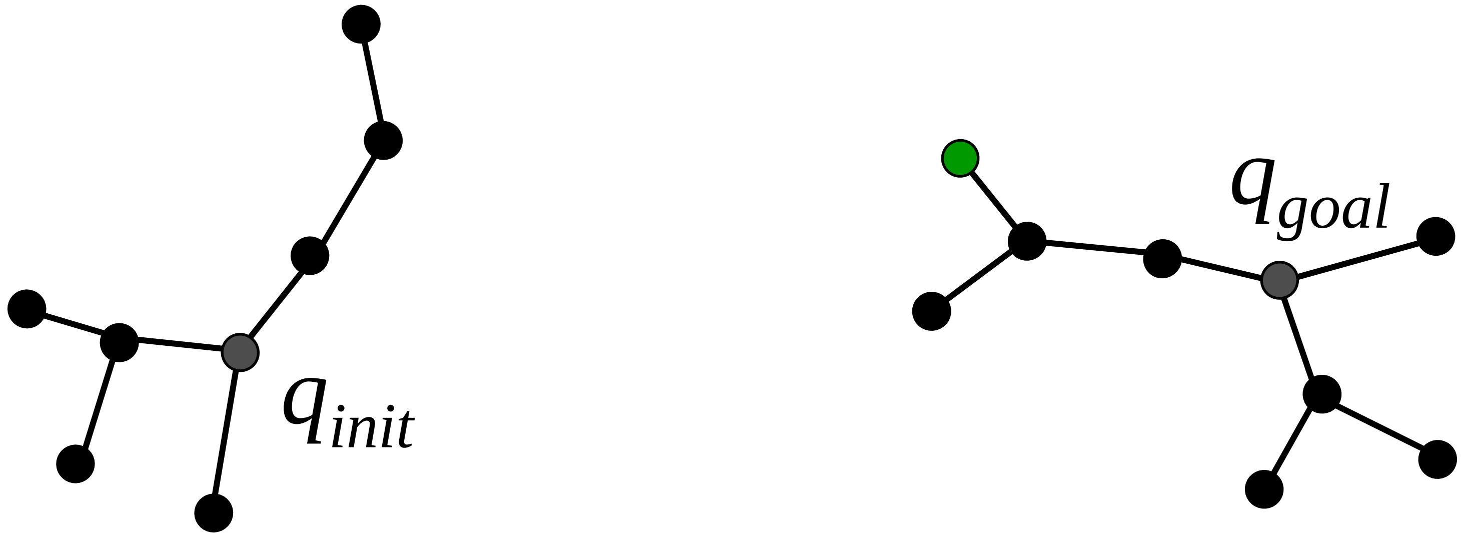
[Kuffner, LaValle ICRA ‘00]



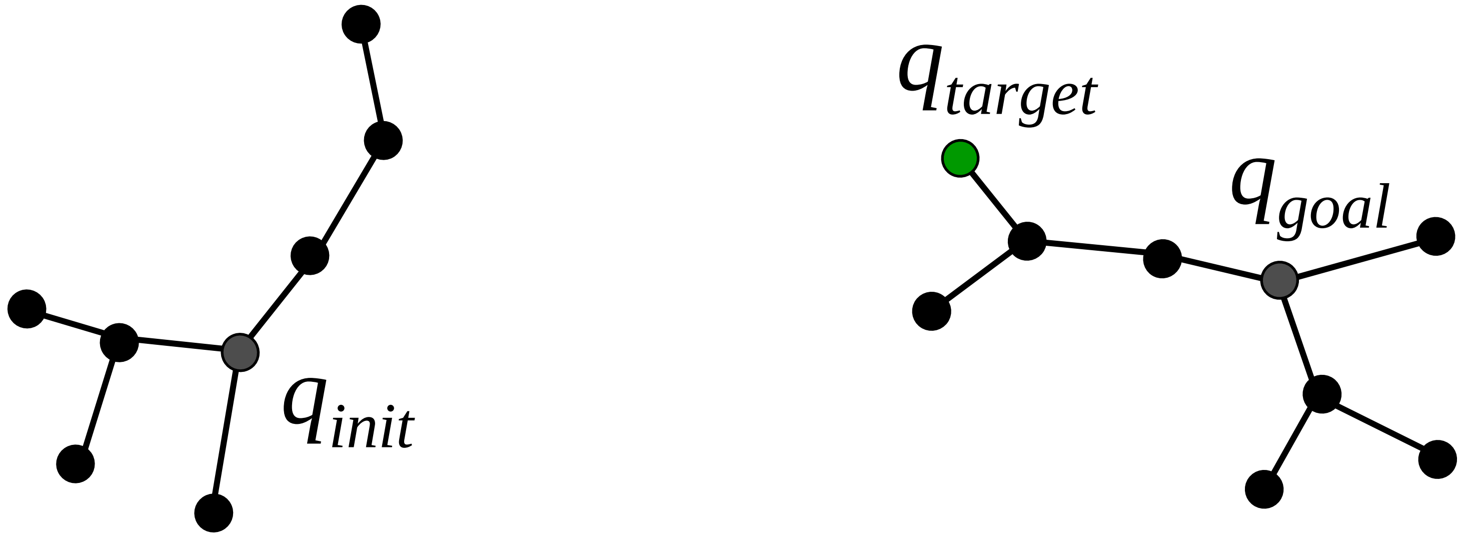
A single RRT-Connect iteration...



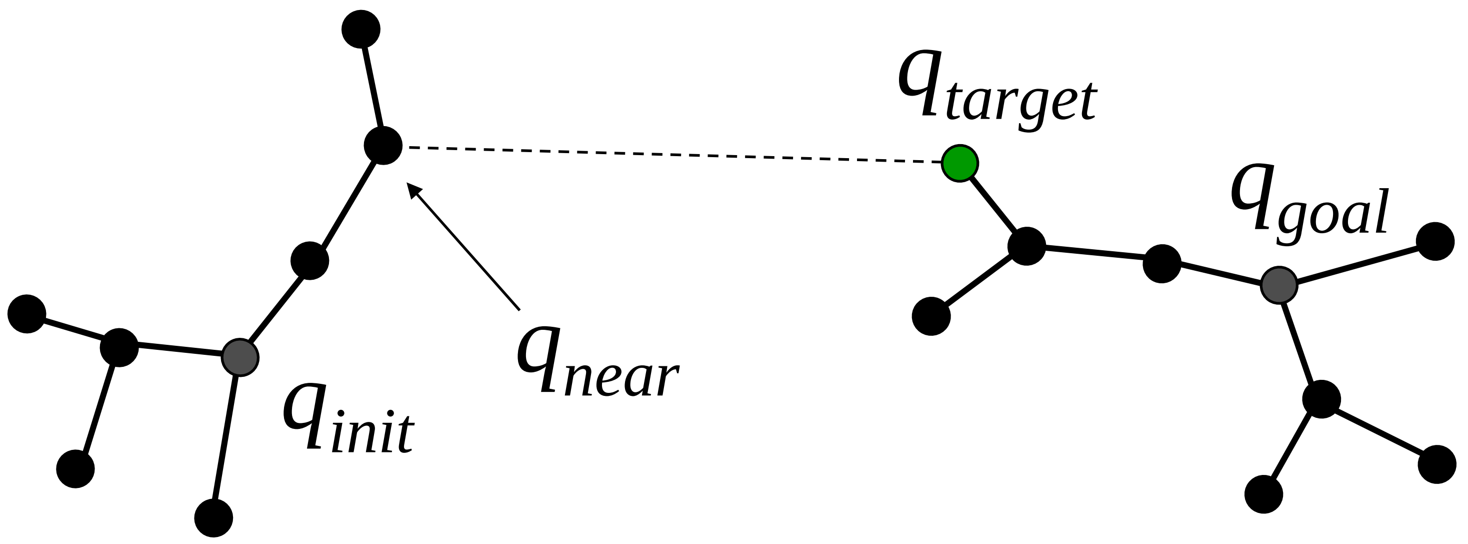
1) One tree grown using random target



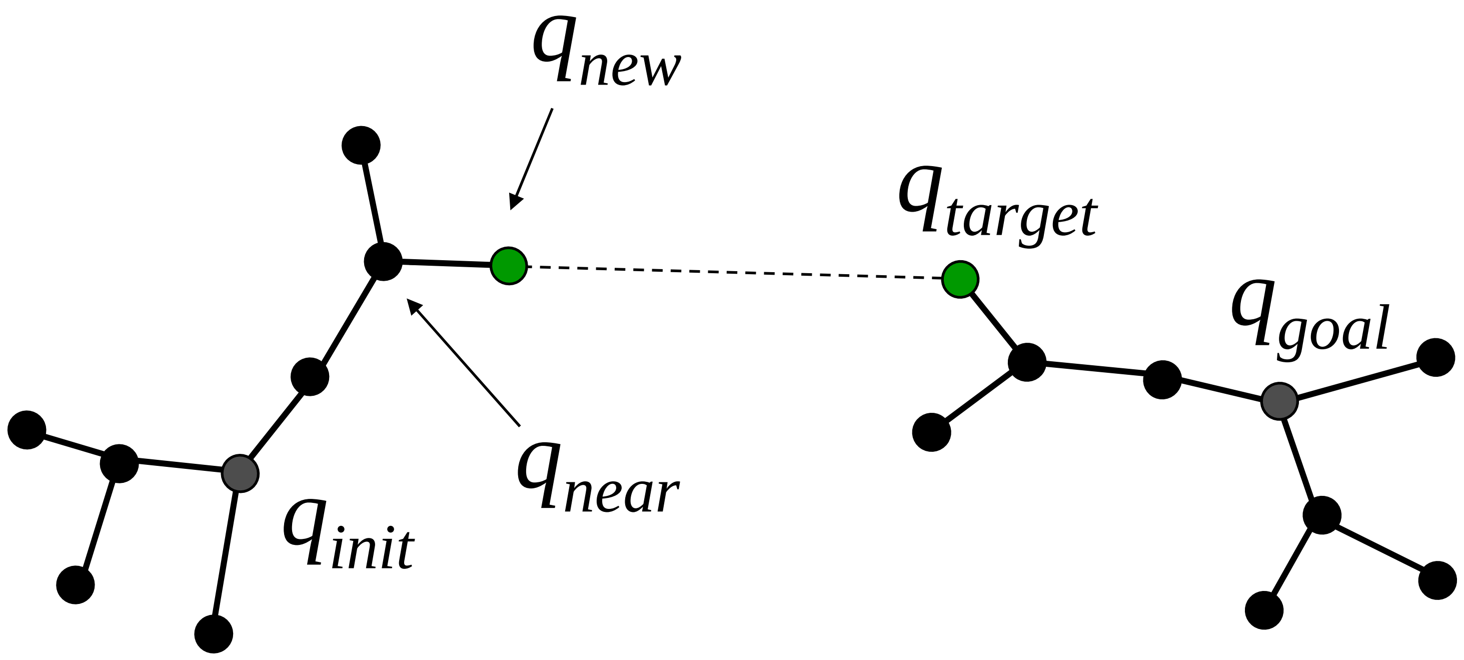
2) New node becomes target for other tree



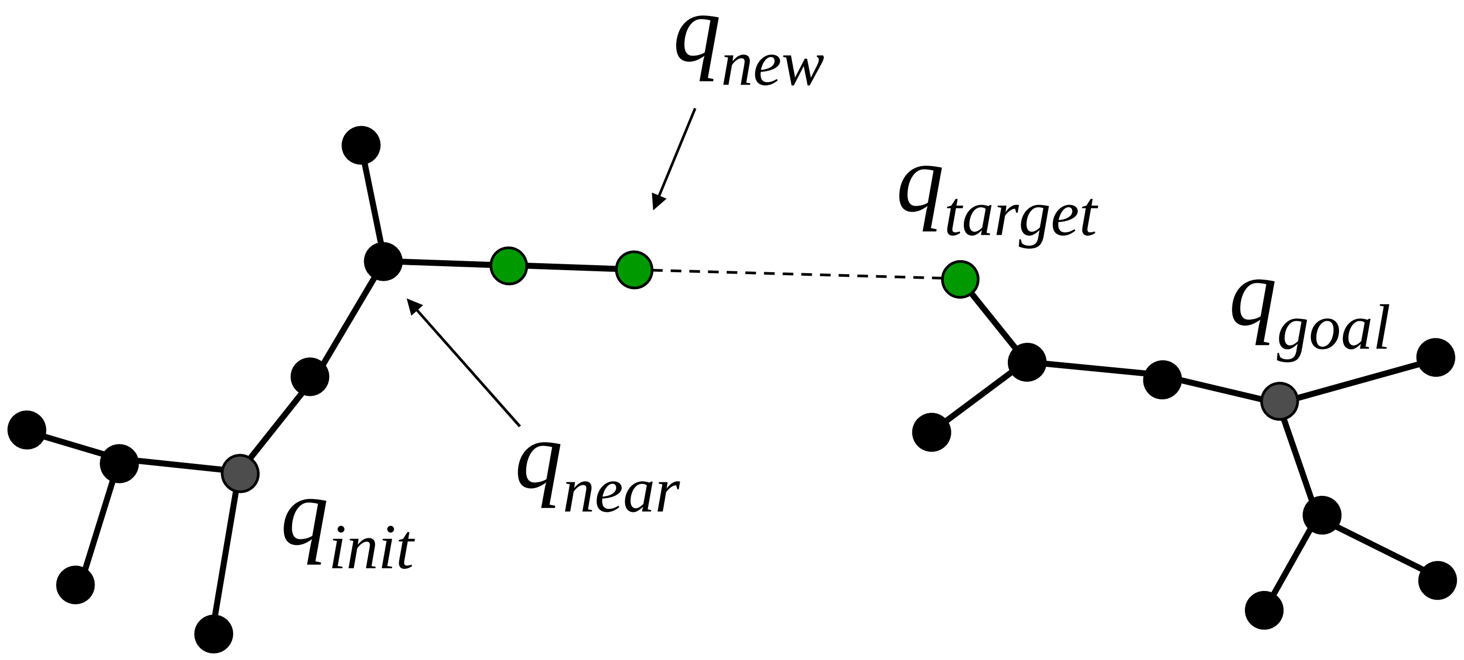
3) Calculate node “nearest” to target



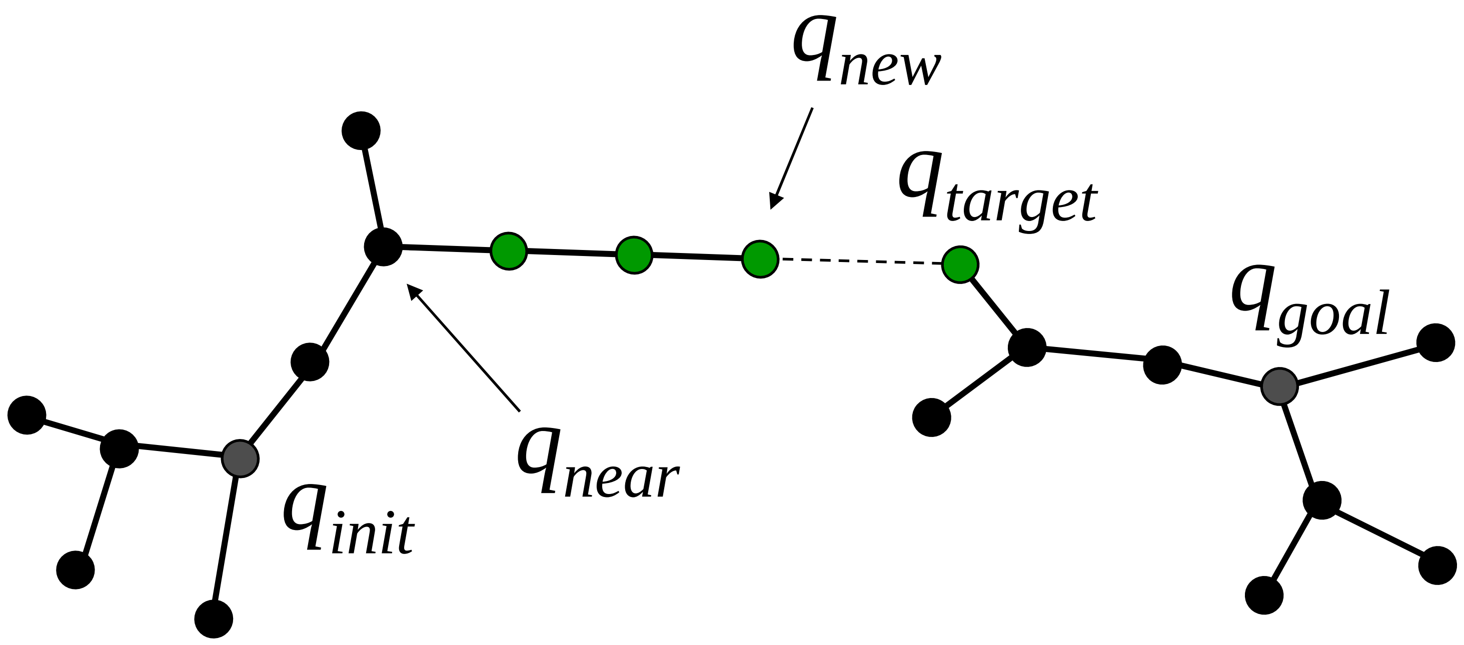
4) Try to add new collision-free branch



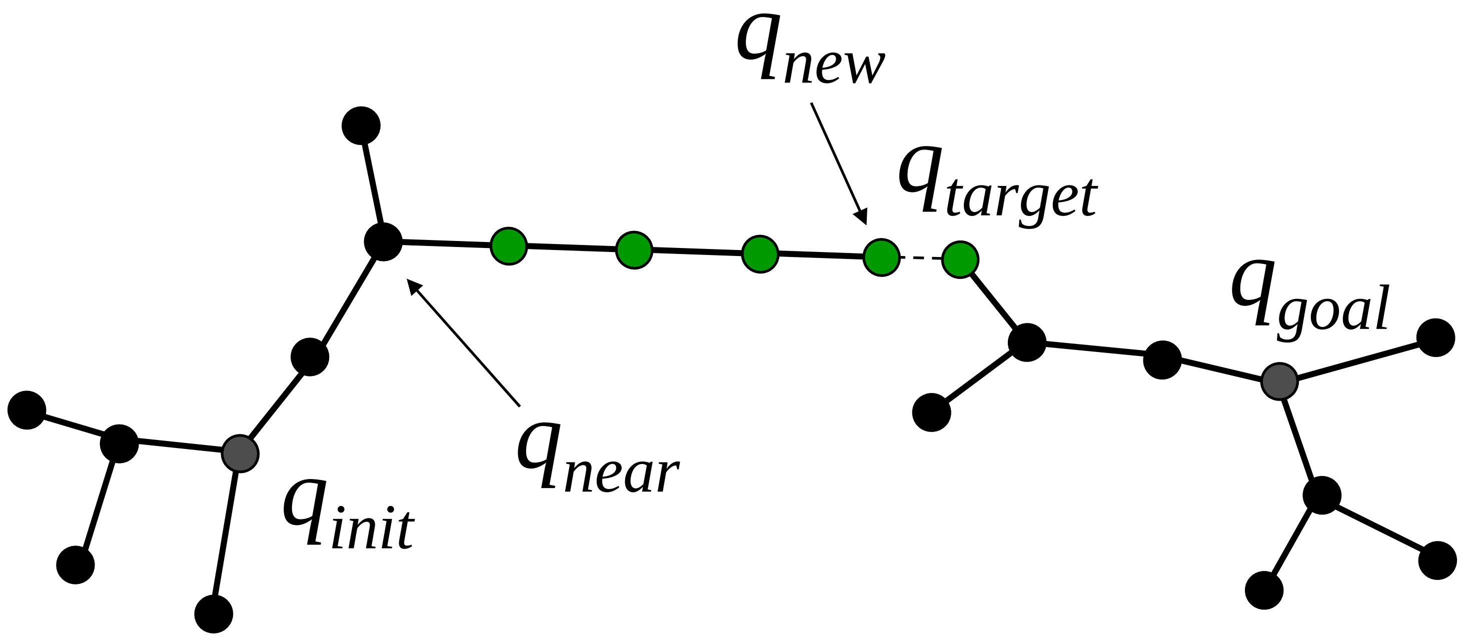
5) If successful, keep extending branch



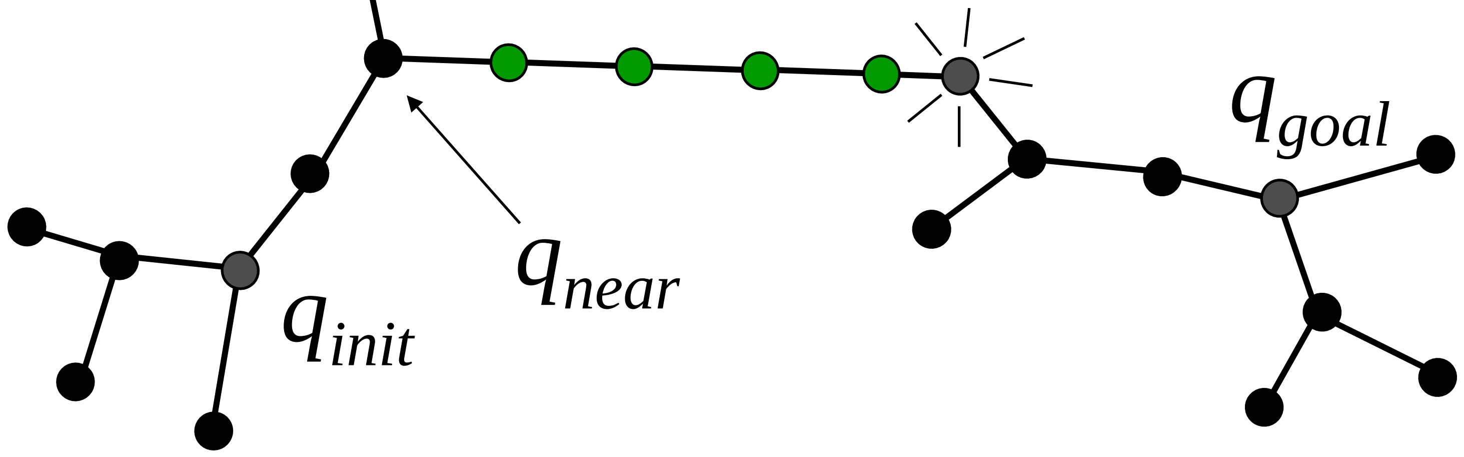
5) If successful, keep extending branch



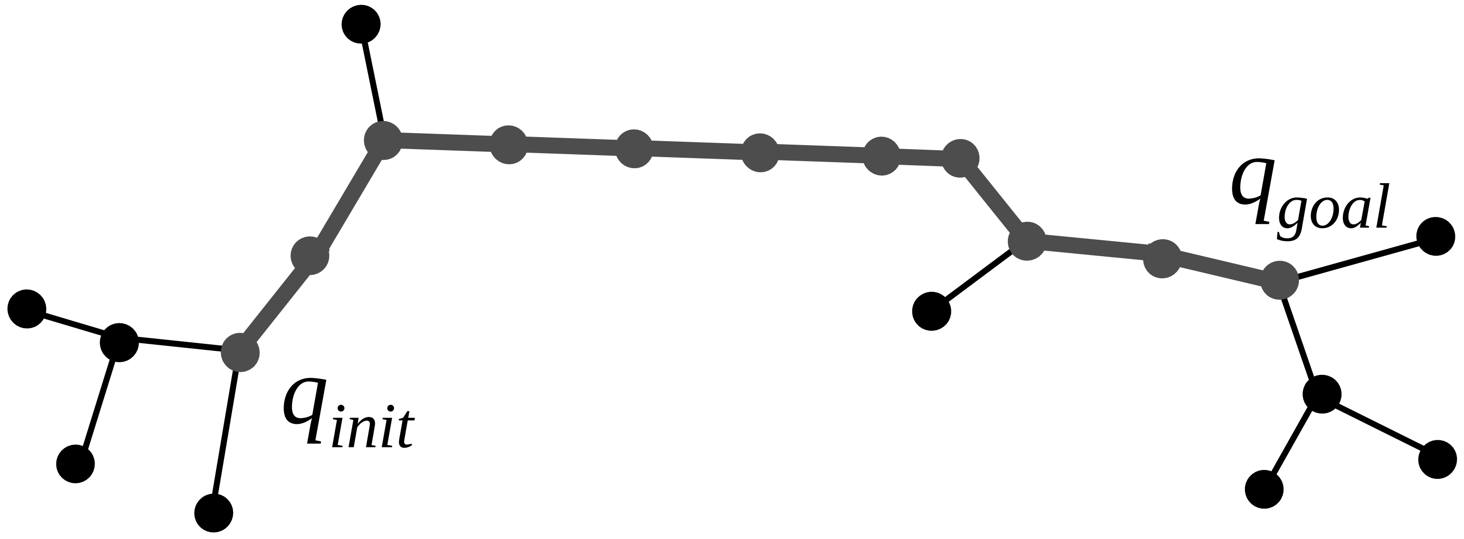
5) If successful, keep extending branch



6) Path found if branch reaches target



7) Return path connecting start and goal

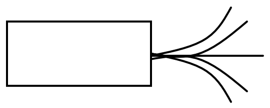


Basic RRT-Connect

```
RRT_CONNECT ($q_{init,}$ $q_{goal}$)  {
   $T_a$.$init(q_{init})$;   $T_b$.$init(q_{goal})$;
   for $k$ =  1 to K do
      $q_{rand}$ = RANDOM_CONFIG();
      if not (EXTEND($T_a$, $q_{rand}$) = Trapped) then
          if (EXTEND($T_b$, $q_{new}$) = Reached) then
               Return PATH($T_{a,}$ $T_b$);
      SWAP($T_{a,}$ $T_b$);
   Return Failure;
}
```

Instead of switching, use T_a as smaller tree. This helped James a lot

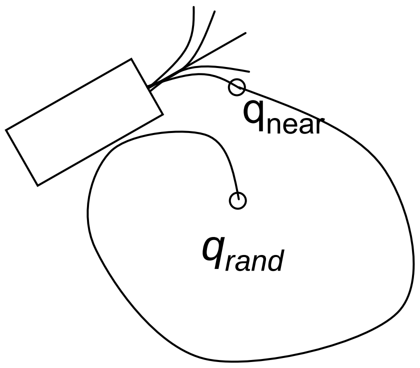
q_{near}

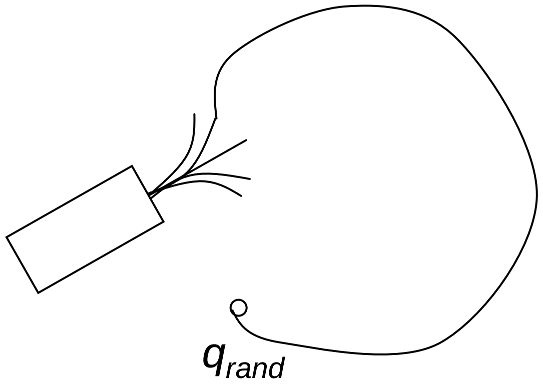


$q' = f(q,u)$ --- use action u from q to arrive at q'

chose $u_* = \arg\min(d(q_{rand}, q'))$

Is this the best?



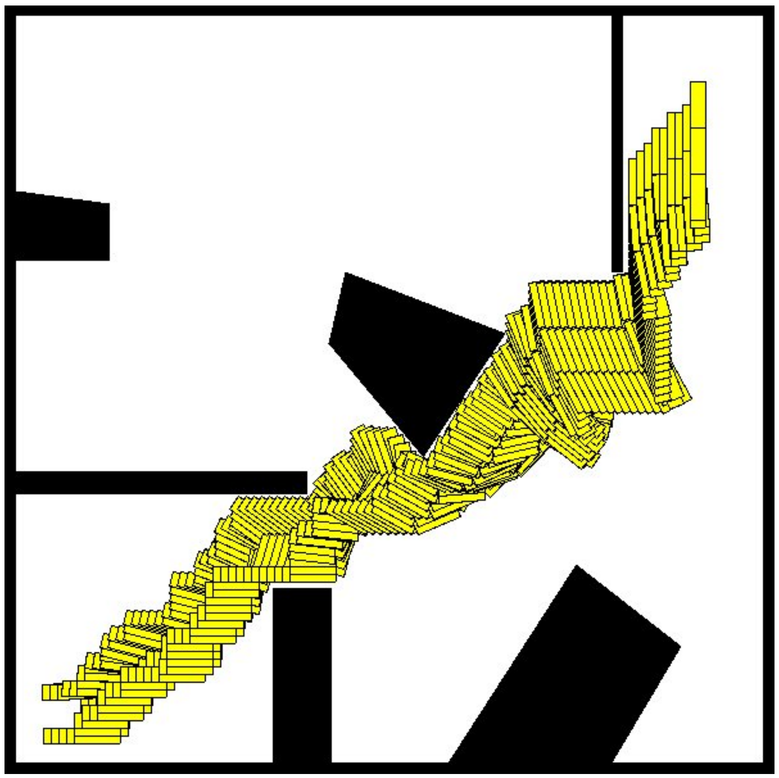


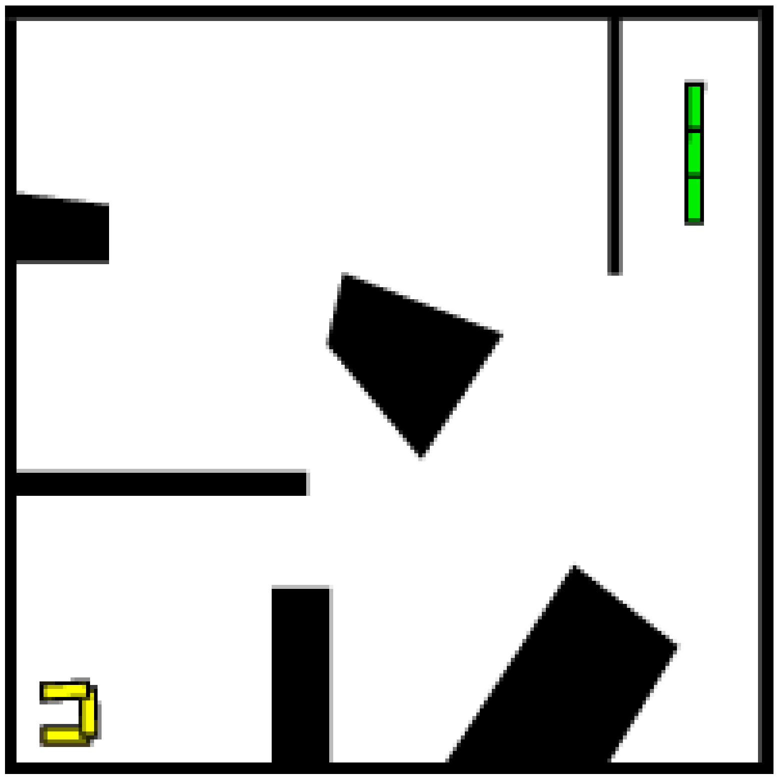
Mixing position and velocity, actually mixing position, rotation and velocity is hard

So, what do they do?

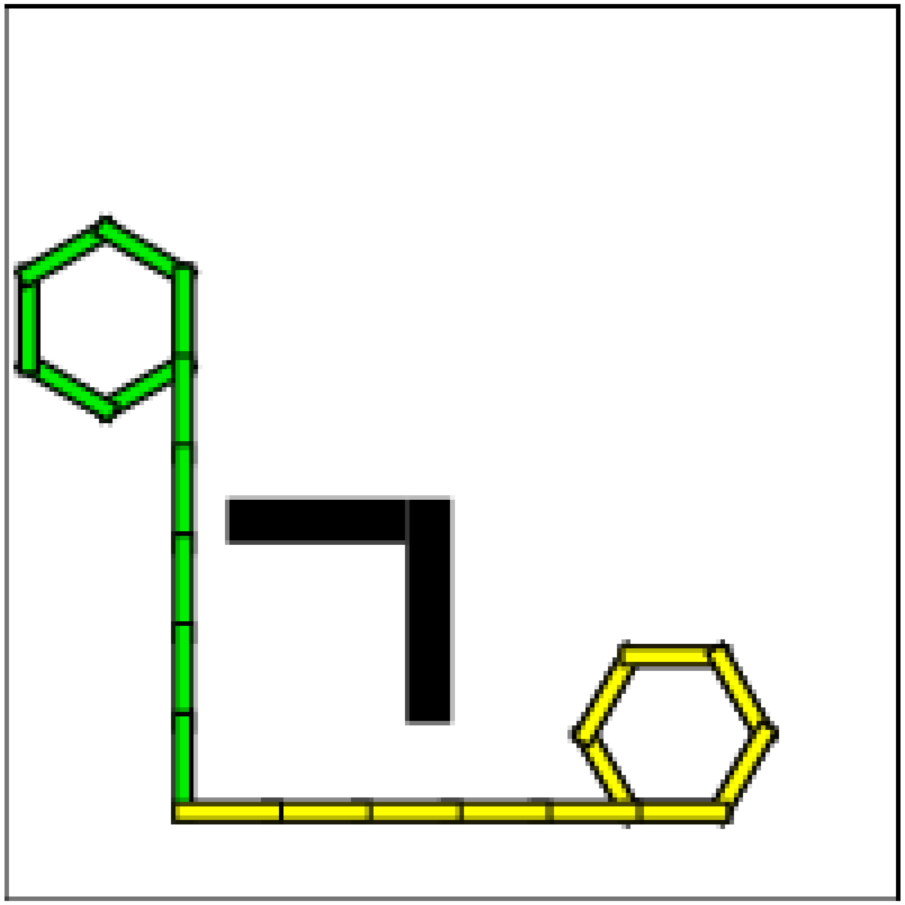
- Use nearest neighbor anyway
- As long as heuristic is not bad, it helps
  (you have already given up completeness and optimality, so what the heck?)
- Nearest neighbor calculations begin to dominate the collision avoidance (James says 50,000 nodes)
- Remember K-D trees

Articulated Robot

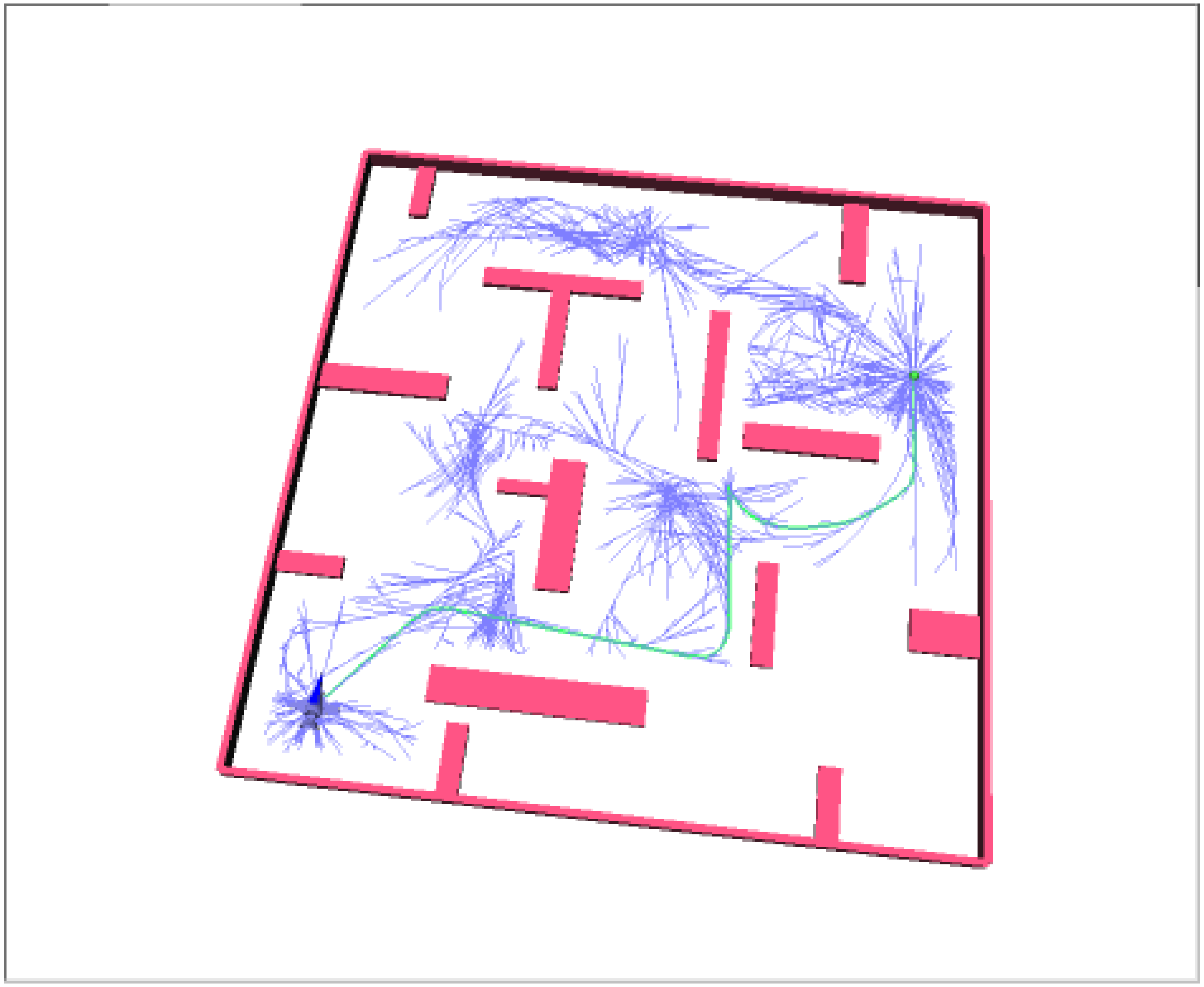




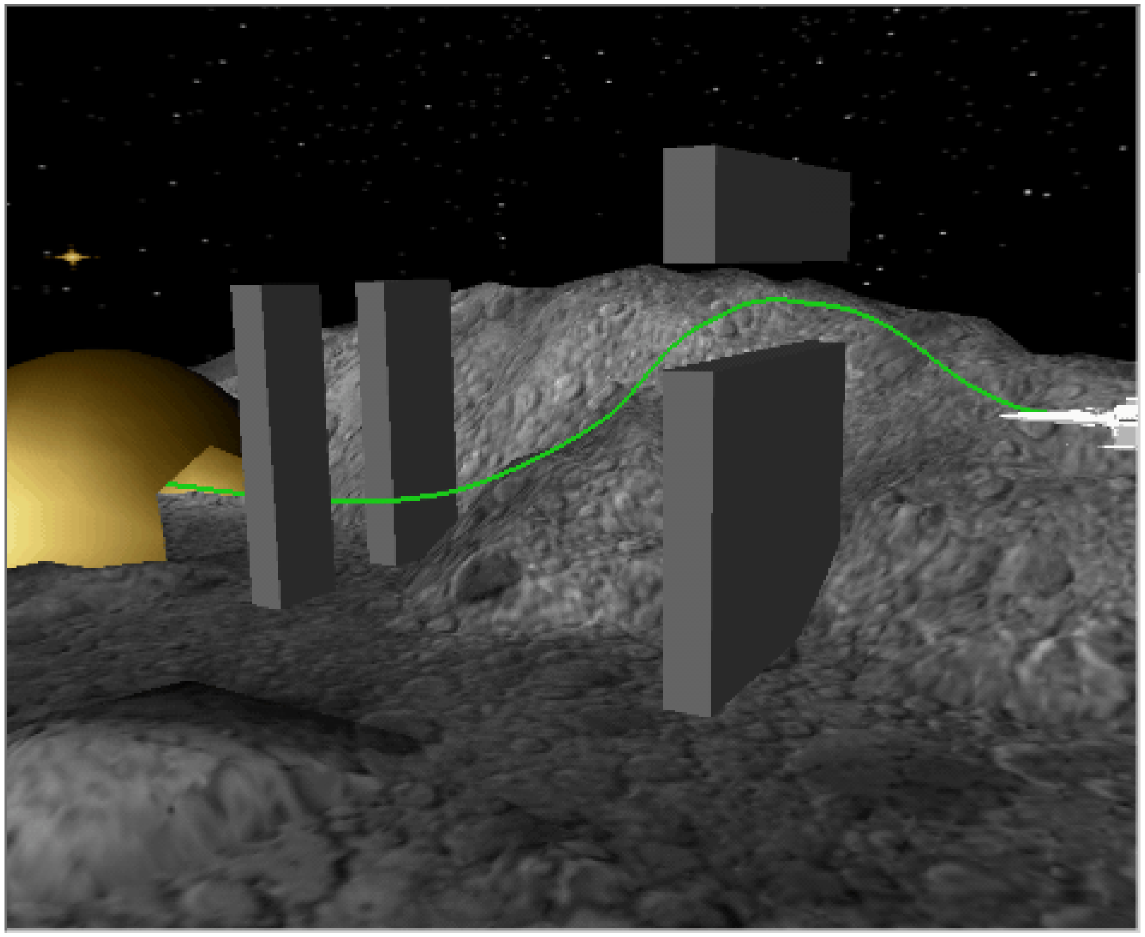
Highly Articulated Robot



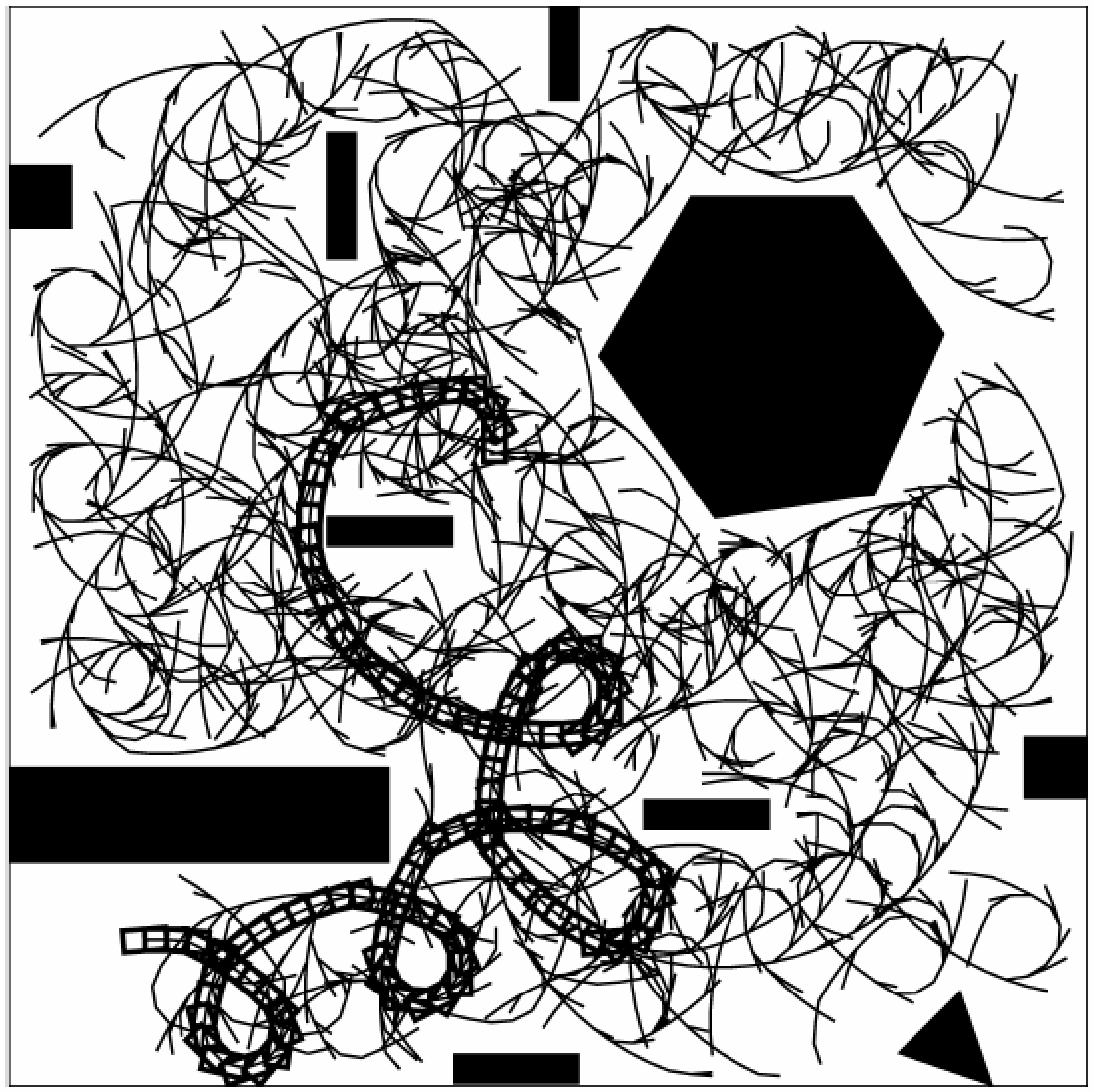
Hovercraft with 2 Thusters



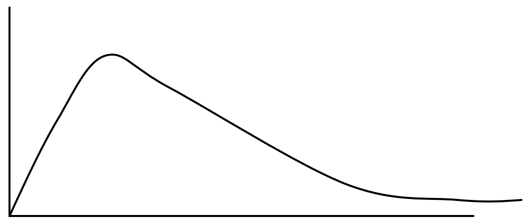
Out of This World Demo



Left-turn only forward car



Analysis



The limiting distribution of vertices:

- THEOREM: $\boldsymbol{X_k}$ converges to $\boldsymbol{X}$ in probability
  - $\boldsymbol{X_k}$: The RRT vertex distribution at iteration k
  - $\boldsymbol{X}$: The distribution used for generating samples
- KEY IDEA: As the RRT reaches all of Q_{free}, the probability that q_{rand} immediately becomes a new vertex approaches one.

Rate of convergence:

- The probability that a path is found increases exponentially with the number of iterations.

"This is the bain or the worst part of the algorithm," J. Kuffner

Open Problems

Open Problems

- Rate of convergence
- Optimal sampling strategy?

Open Issues

- Metric Sensitivity
- Nearest-neighbor Efficiency

Applications of RRTs

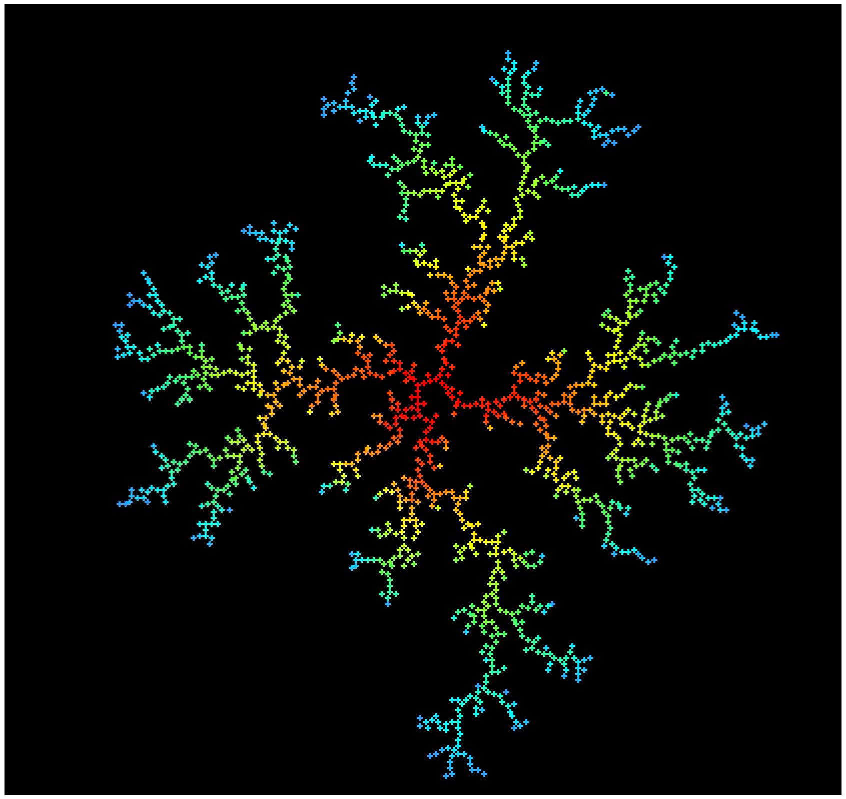
Robotics Applications
- mobile robotics
- manipulation
- humanoids

Other Applications
- biology (drug design)
- manufacturing and virtual prototyping (assembly analysis)
- verification and validation
- computer animation and real-time graphics
- aerospace

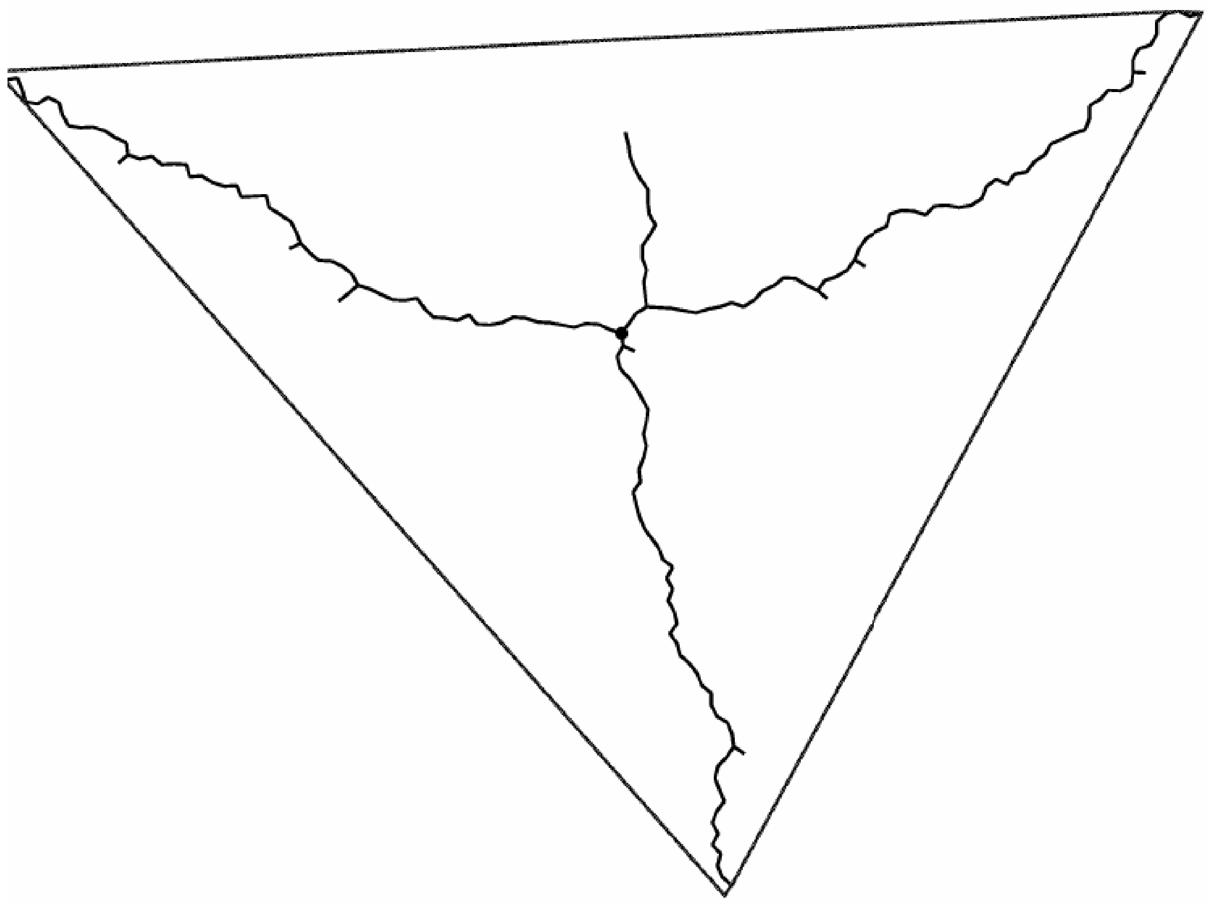
RRT extensions
- discrete planning (STRIPS and Rubik's cube)
- real-time RRTs
- anytime RRTs
- dynamic domain RRTs
- deterministic RRTs
- parallel RRTs
- hybrid RRTs

Diffusion Limited Aggregation

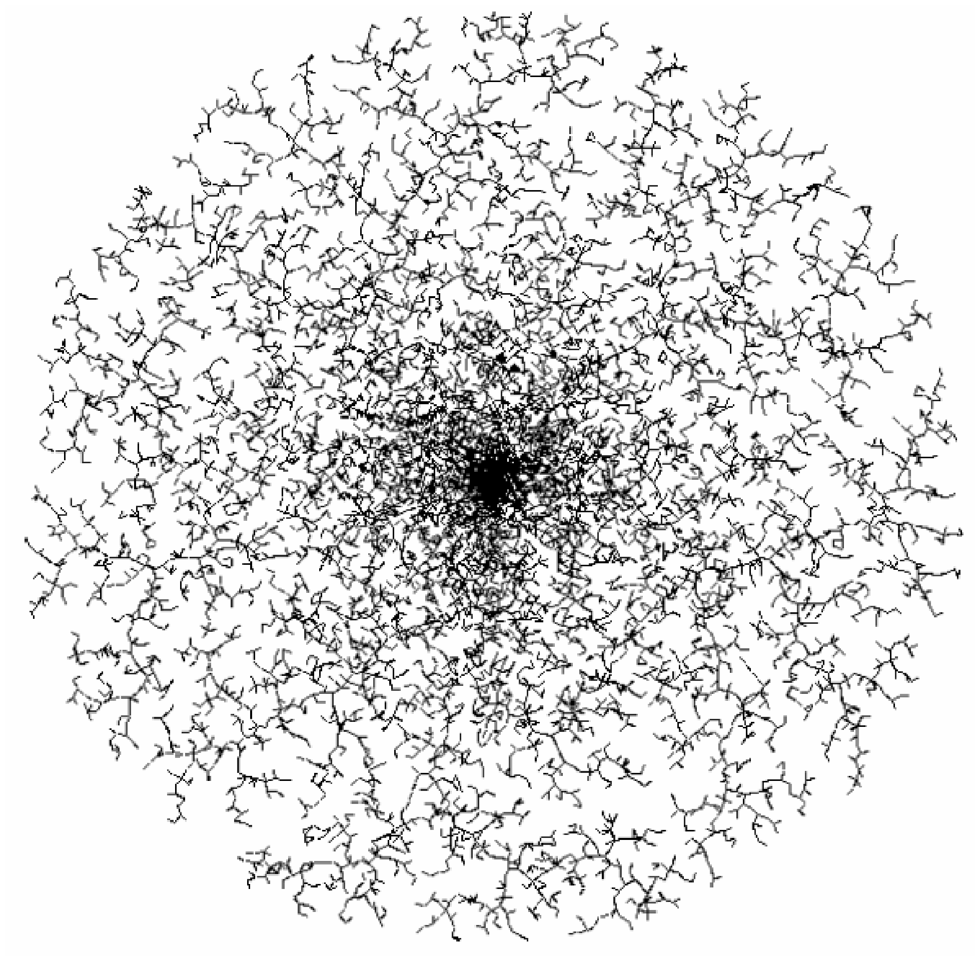
- Often used to model natural physical processes (e.g. snow accumulation, rust, etc.)



Exploring Infinite Space



Polar Sampling



RRT Summary

Advantages

- Single parameter
- Balance between greedy search and exploration
- Converges to sampling distribution in the limit
- Simple and easy to implement

Disadvantages

- Metric sensitivity
- Nearest-neighbor efficiency
- Unknown rate of convergence
- “long tail” in computation time distribution

Links to Further Reading

- Steve LaValle's online book:
  "Planning Algorithms" *(chapters 5 & 14)*
  http://planning.cs.uiuc.edu/

- The RRT page:
  http://msl.cs.uiuc.edu/rrt/

- Motion Planning Benchmarks
  Parasol Group, Texas A&M
  http://parasol.tamu.edu/groups/amatogroup/benchmarks/mp/

PRT (Prob. Roadmap of Trees)

- Basic idea:
  - Generate a set of trees in the configuration space
  - Merge the trees by finding nodes that can be connected
- Algorithm
  - pick several random nodes
  - Generate trees T_1, T_2 .... T_n (EST or RRT)
  - Merge trees
    - generate a representative super-node
    - Using PRS ideas to pick a neighborhood of trees
    - Δ is now the tree-merge algorithm
  - For planning
    - generate trees from initial and goal nodes towards closest supernodes
    - try to merge with “roadmap” of connected trees
- Note that PRS and tree-based algorithms are special cases