Sampling-based Planning 2

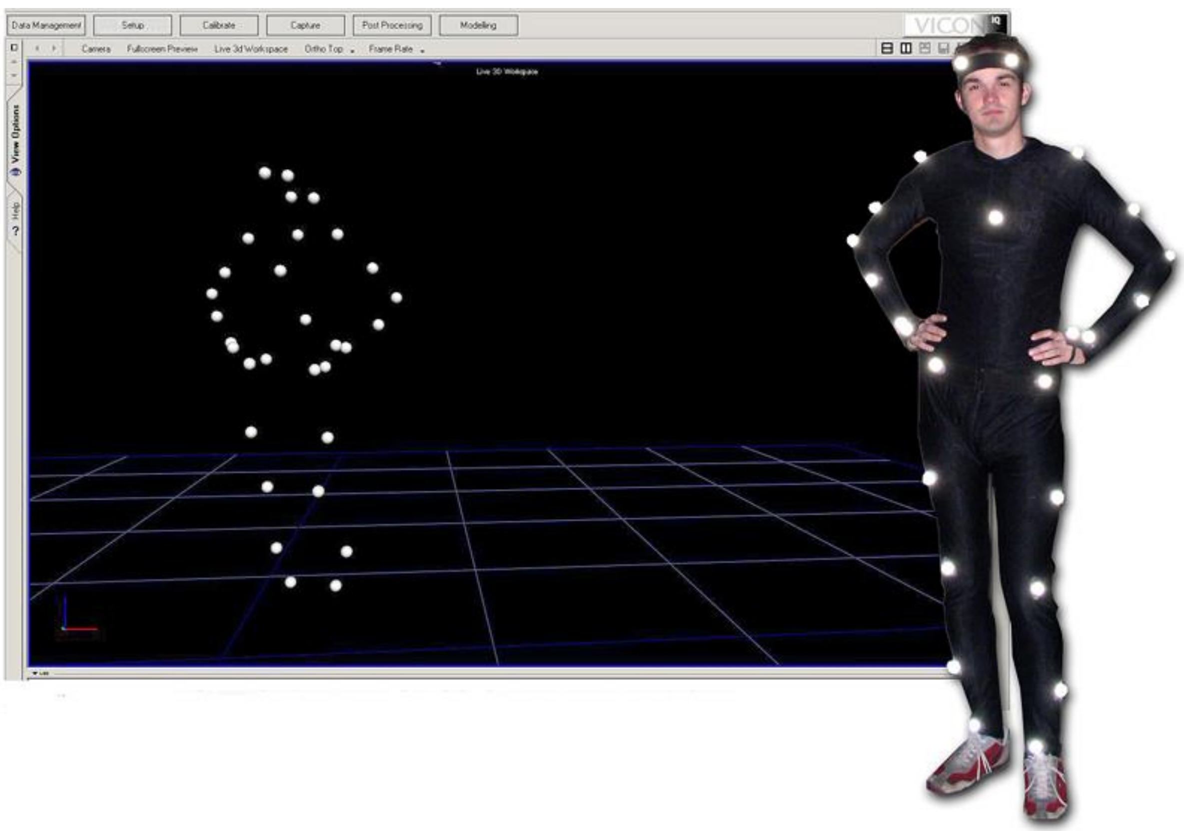
Jane Li
Assistant Professor
Mechanical Engineering & Robotics Engineering
http://users.wpi.edu/~zli11

Problem with KD-tree

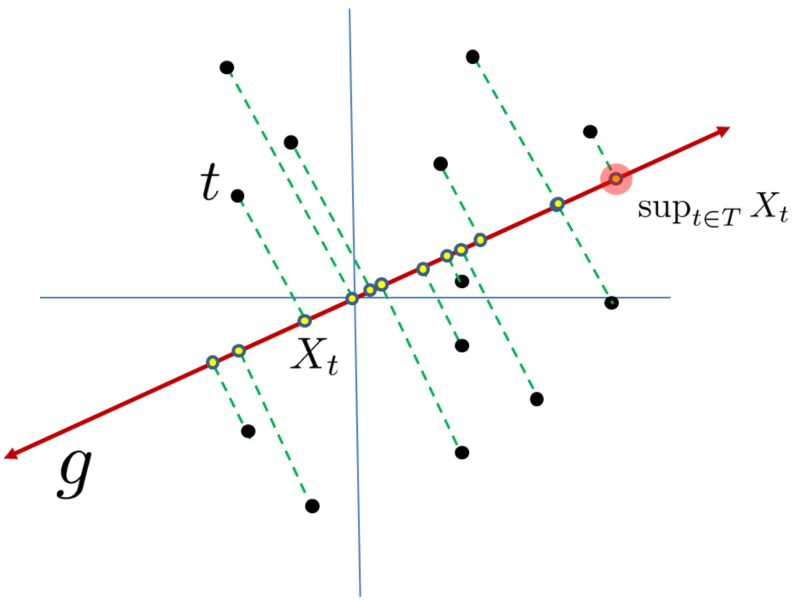
- Curse of dimension
  - N-dimensional configuration space, requires N level to halve the cell diameters
- Other (popular) trees for space partition in nearest neighbor search?
  - Random projection (RP) tree
  - Principal direction (PD) tree

Dimension Reduction

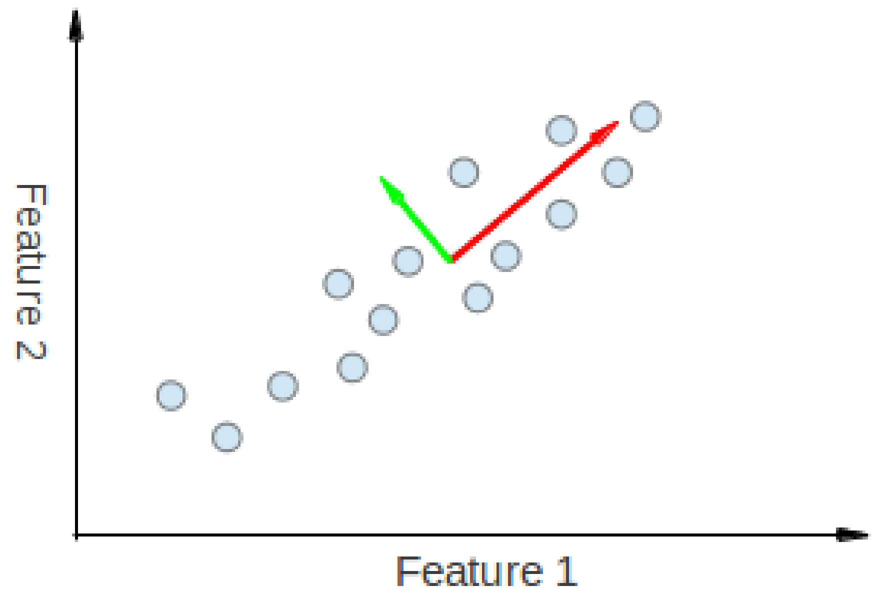
- A lot of data which superficially lie in a very high-dimensional space, actually have **low intrinsic dimension**



Dimension Reduction



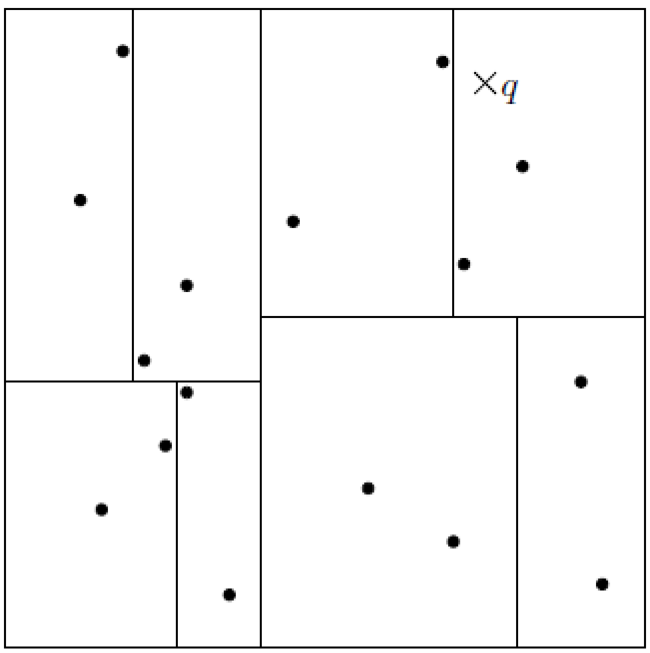
Random projection



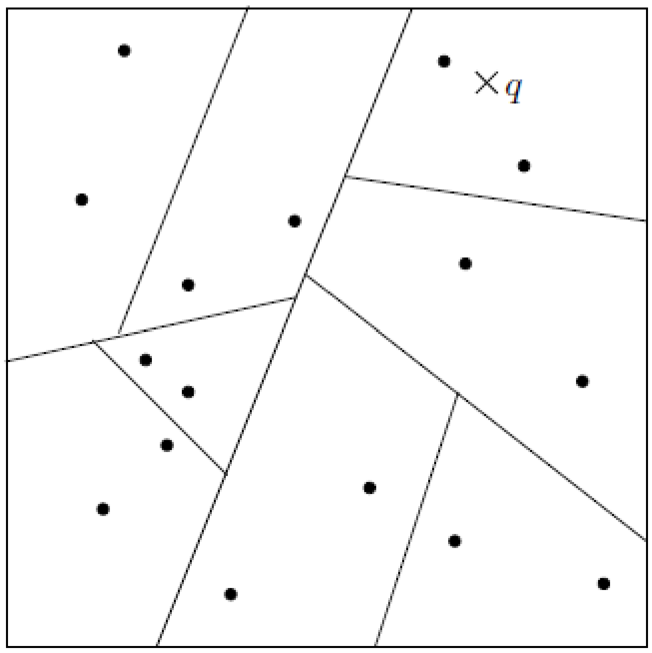
Principle Component Analysis

Random Projection Tree

- If the N-dimensional data has intrinsic dimension n, then an RP tree halves the diameter in just d levels – no dependence on N



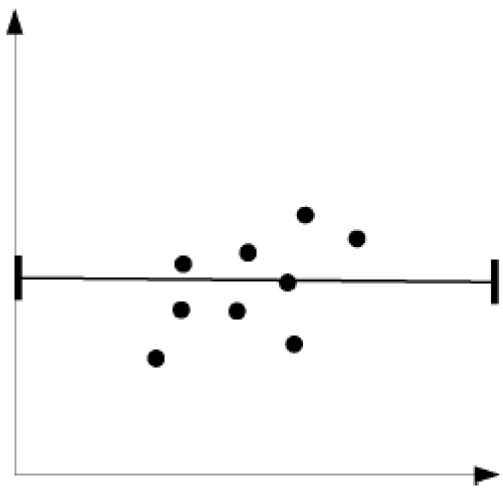
KD-tree
Pick coordinate direction and split at median



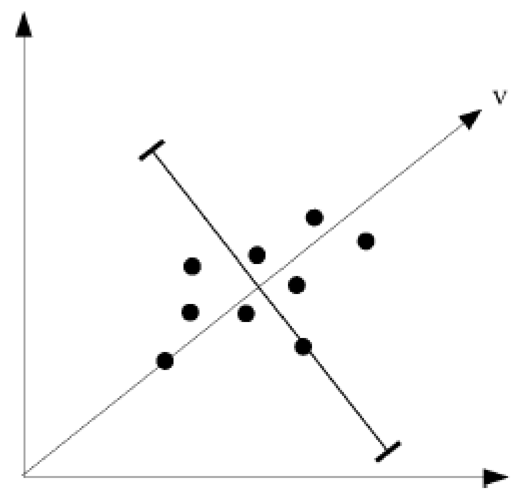
RP-tree
Pick random direction and split at median plus noise

Principal Direction Tree

- Principle value decomposition



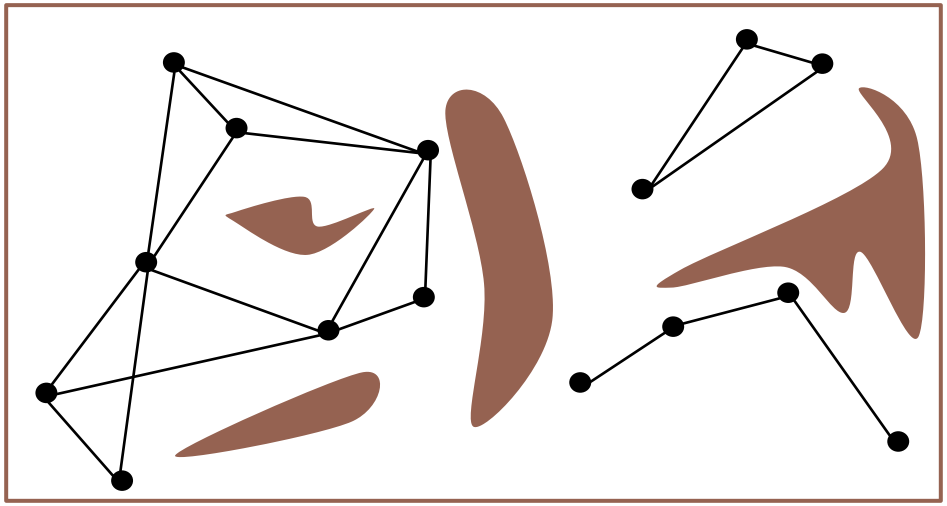
KD-tree
Partition with hyperplane perpendicular to an axis



PD-tree
Partition with hyperplane perpendicular to the principle axis direction

Recap

- Last time, we discussed PRMs



- Two issues with the PRM:
    1. Uniform random sampling misses **narrow passages**
    2. Exploring **whole space**, but all we want is a path

Outline

- Sampling strategies
- RRTs

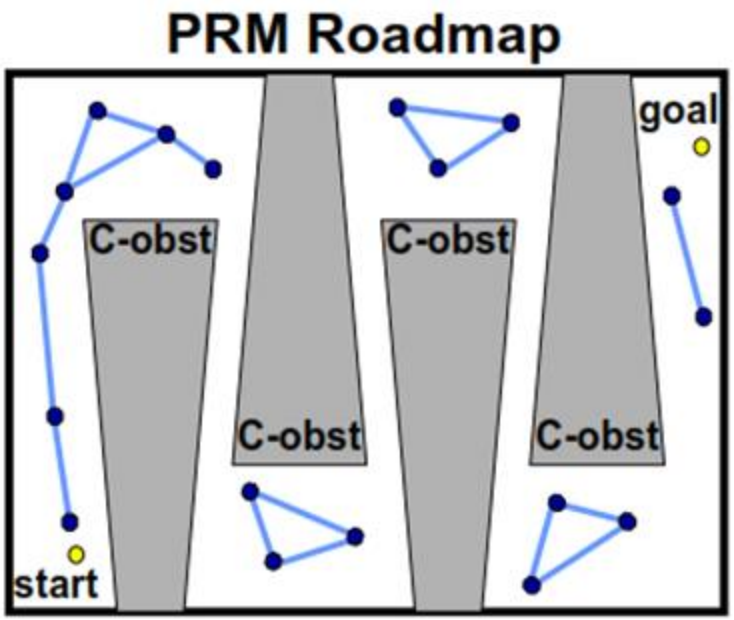
Sampling Strategies

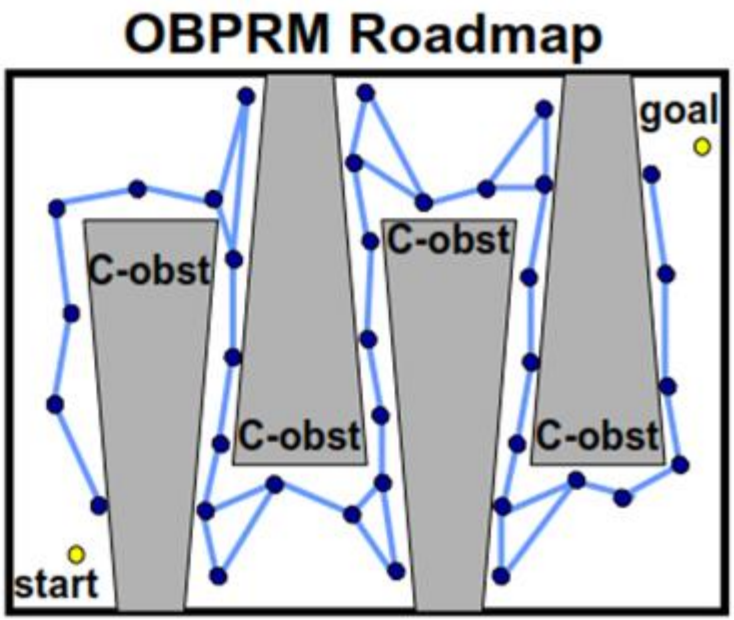
- Uniform random sampling – Most common
  - The bigger the area, the more likely it will be sampled
  - Problem – Narrow passages

- Different sampling strategies?

Obstacle-based PRM

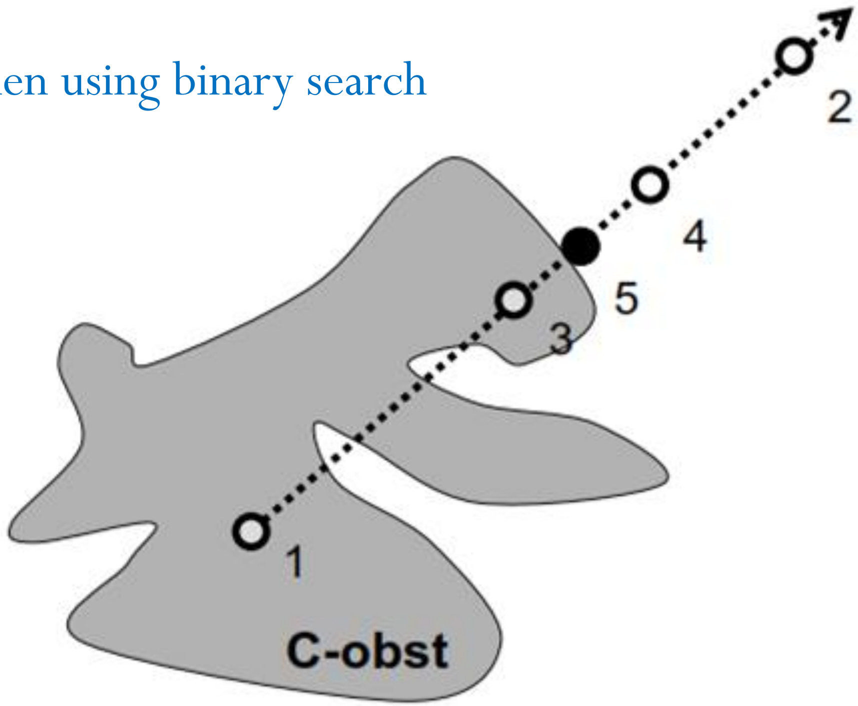
- To navigate narrow passage, we must sample in them
  - Uniform sampling – Most PRM points fall in where planning is easy
  - Sample near C-obstacles?
    - But we cannot explicitly construct C-obstacles …



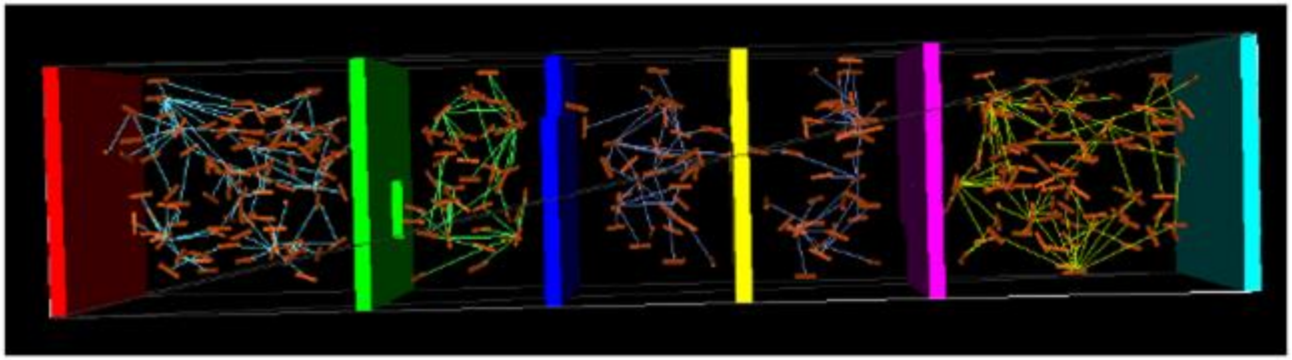


Obstacle-based PRM

- How to find points on C-obstacles?
  - Find a point in the C-obstacles – a collision configuration
  - Select a random direction in C-space
  - Find a free point in that direction
  - Find the boundary point between then using binary search

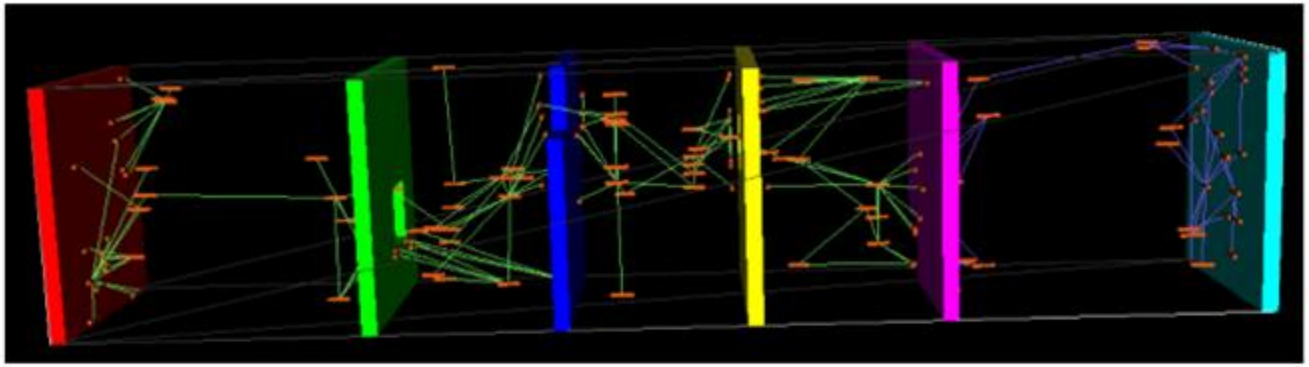


PRM VS OBPRM



PRM

- 328 nodes
- 4 major CCs



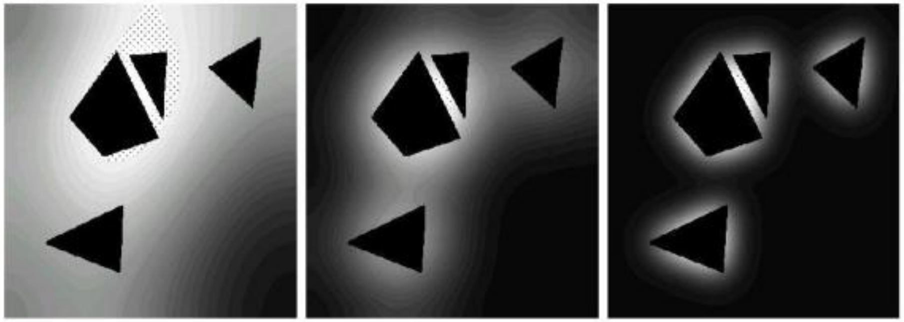
OBPRM

- 161 nodes
- 2 major CCs

Gaussian Sampling

- Gaussian sampler
  - Find a q_1
  - Pick a q_2 from a Gaussian distribution centered at q_1
  - If **both** are in collision or collision-free, discard them, if one free, keep it

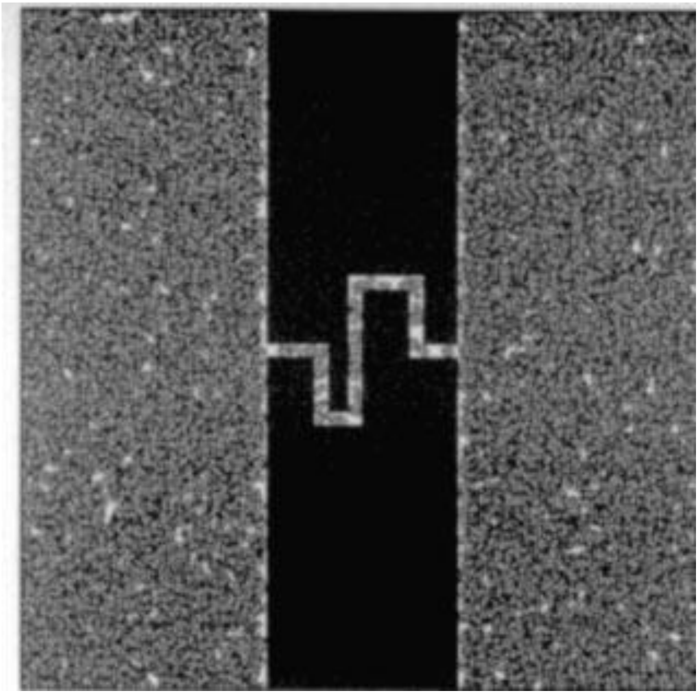
What is the effect?



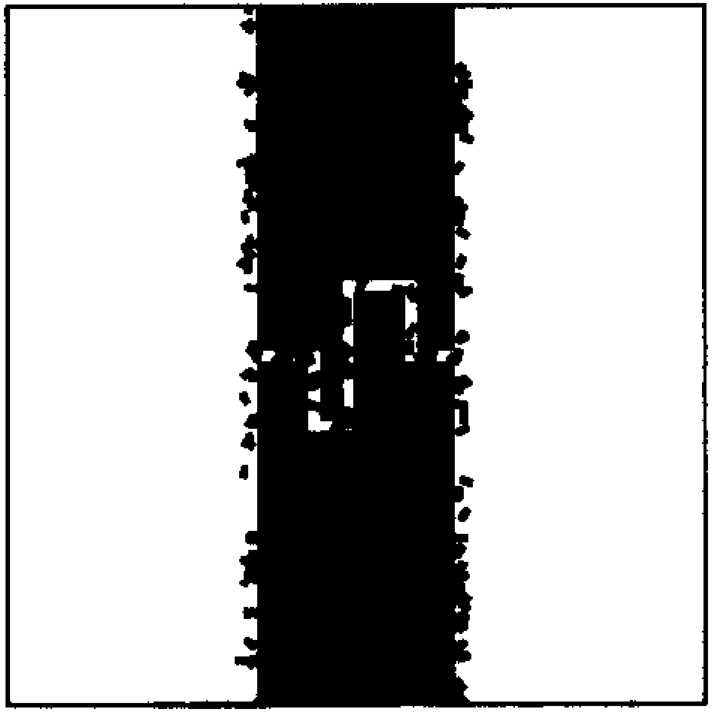
Sampling distribution for varying Gaussian width
(width decreasing from left to right)

Boor, Valérie, Mark H. Overmars, and A. Frank van der Stappen. "The gaussian sampling strategy for probabilistic roadmap planners." *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. Vol. 2. IEEE, 1999.

Gaussian Sampling



Milestones (13,000) created by uniform sampling before the narrow passage was adequately sampled



Milestones (150) created by Gaussian sampling

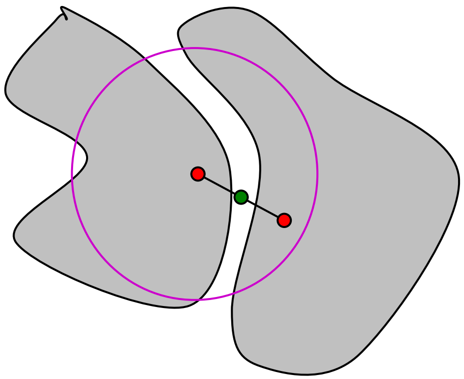
The gain is not in sampling fewer milestones, but in connecting fewer pairs of milestones

Bridge

- Bridge sampler

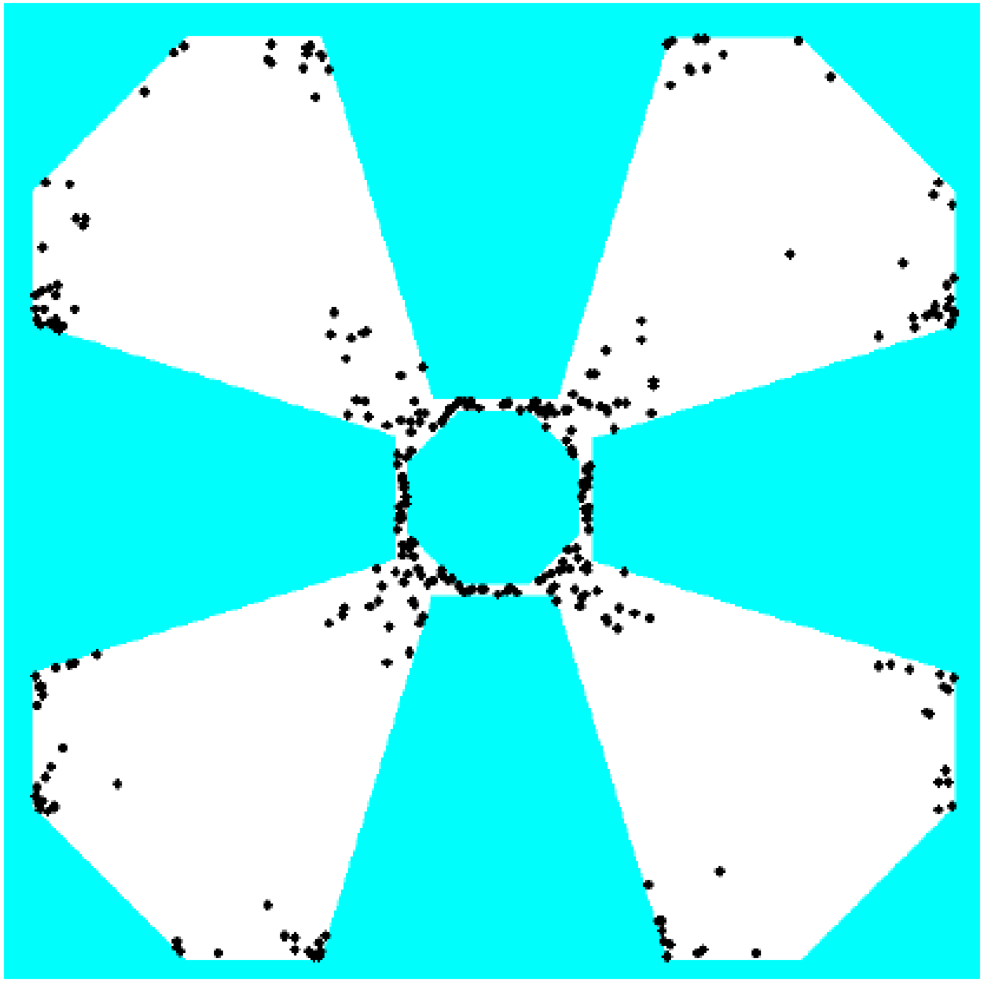
What is the effect?

  - Sample a q_1 that is in collision
  - Sample a q_2 in neighborhood of q_1 using some probability distribution (e.g. gaussian)
  - If q_2 in collision, get the midpoint of (q_1, q_2)
  - Check if midpoint is in collision, if not, add it as a node

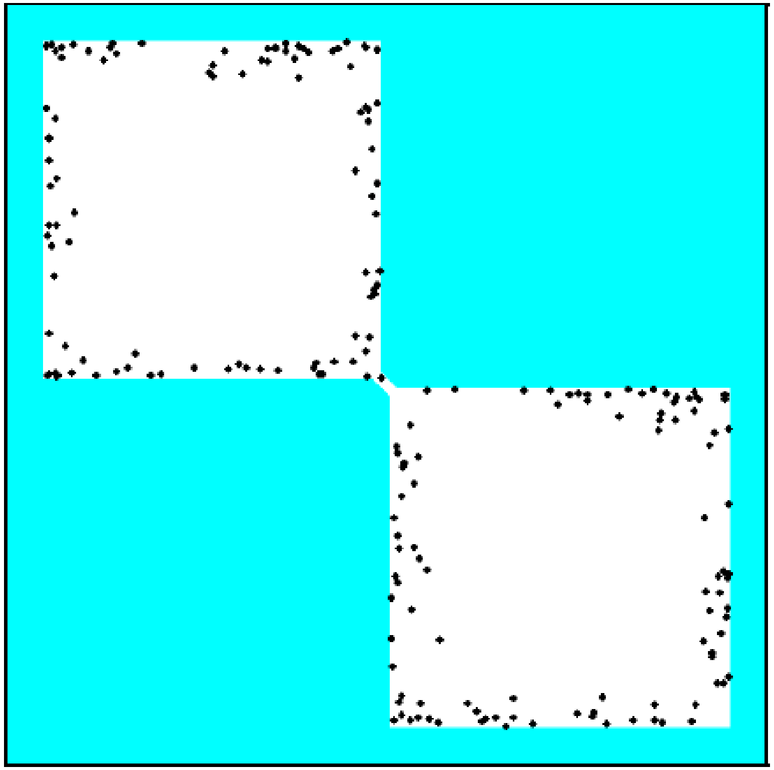


Hsu, David, et al. "The bridge test for sampling narrow passages with probabilistic roadmap planners." *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*. Vol. 3. IEEE, 2003.

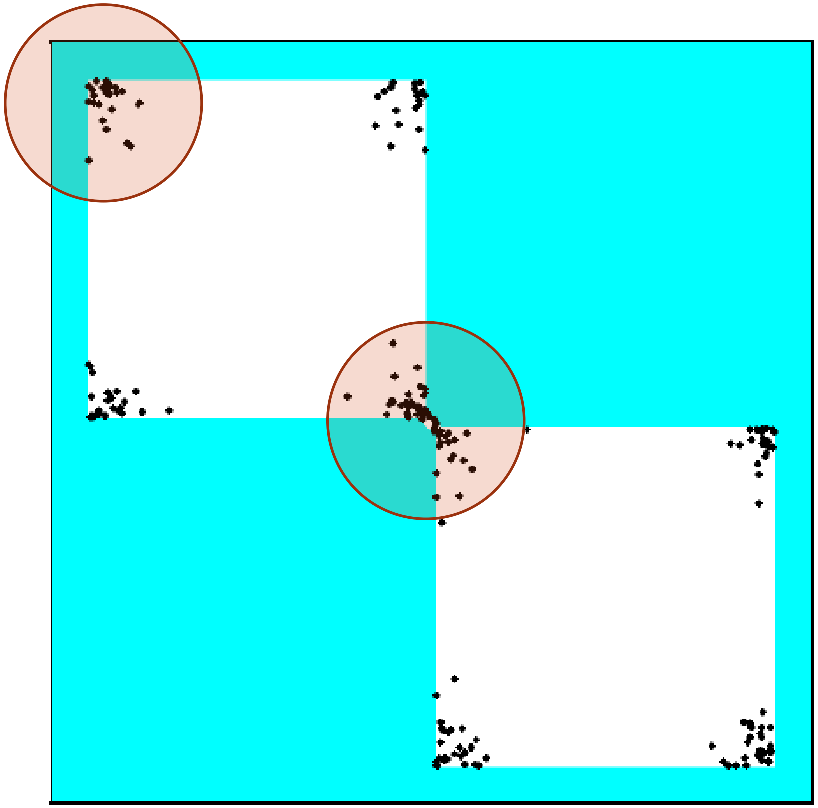
Bridge



Bridge

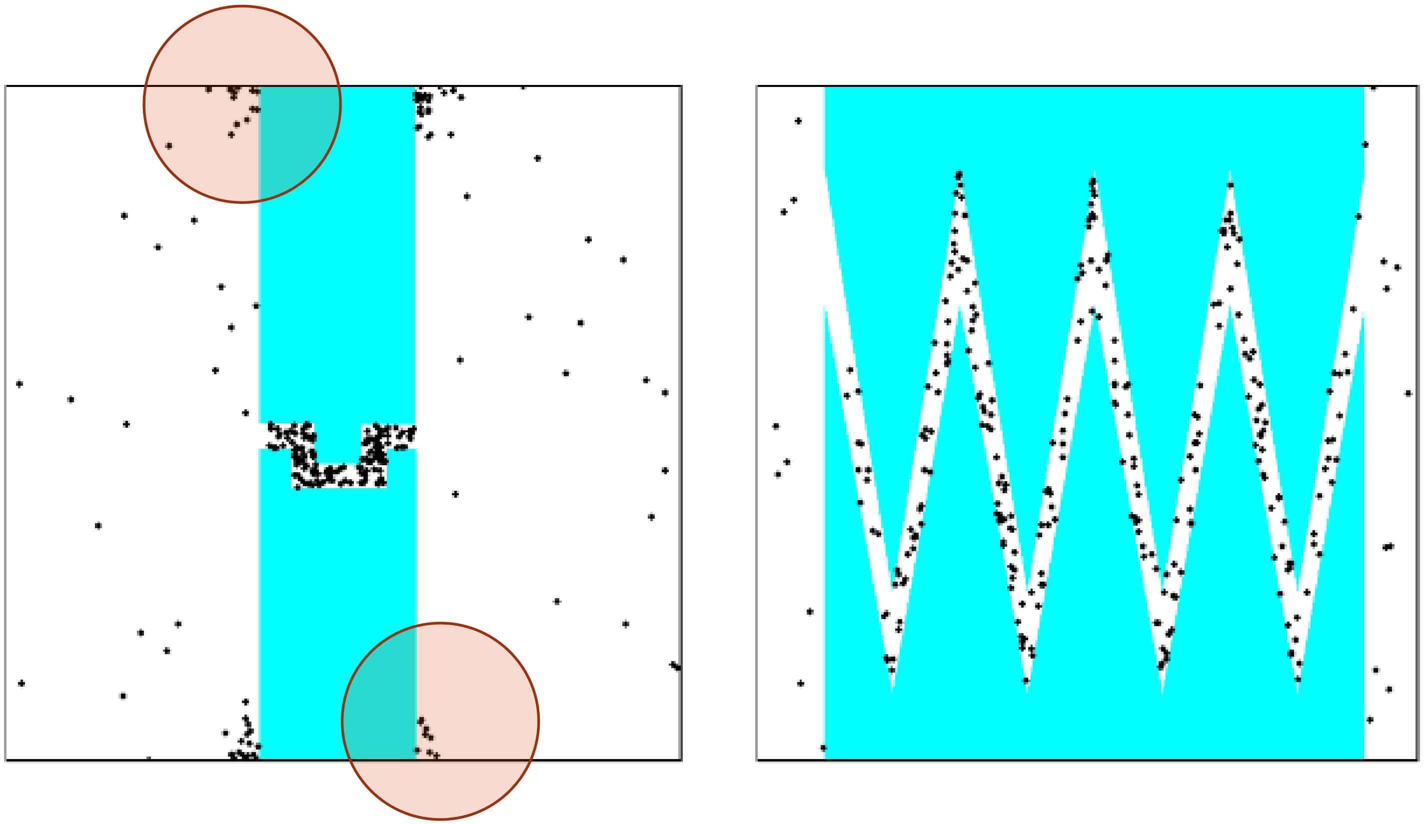


Gaussian



Bridge test

Bridge



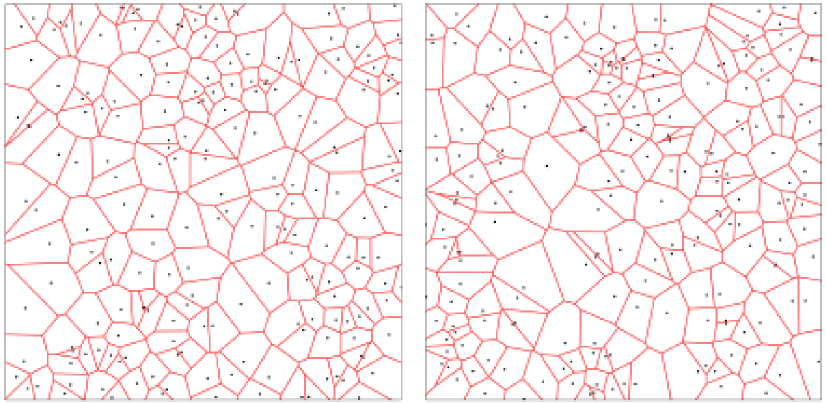
Bridge Sampling performs well in narrow passages

What's going on at the corners?

Deterministic Sampling

What to do?

- The problem:
  - Random sampling (biased or not) can be unpredictable and irregular
    - Each time your run your algorithm you get a **different sequence of samples**, so **performance varies**
    - In the limit, space will be sampled well, but in **finite time result may be irregular**



(a) 196 pseudorandom samples (b) 196 pseudorandom samples

Figure 5.3: Irregularity in a collection of (pseudo)random samples can be nicely observed with Voronoi diagrams.

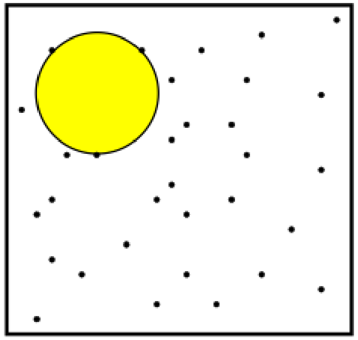
Deterministic Sampling

- What do we care about?
  - Dispersion

$$\delta(P) = \sup_{x \in X} \left\{ \min_{p \in P} \{\rho(x,p)\} \right\}.$$

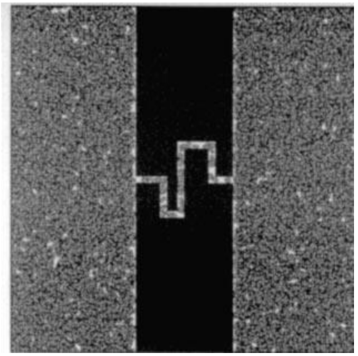
P is a finite set of points, (*X*, ρ) is a metric space (ρ is a distance metric)

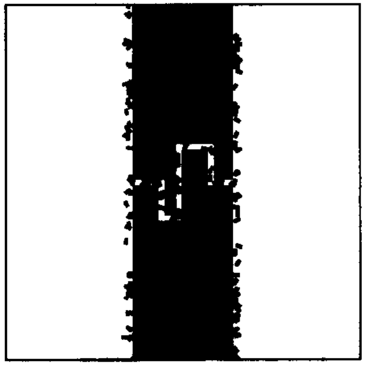
In English: the radius of the largest empty ball



(a) L_2 dispersion

  - What does it mean?
    - Intuitively, the dispersion quantifies how well a space is covered by a set of points S in terms of the largest open Euclidean ball that touches none of the points.





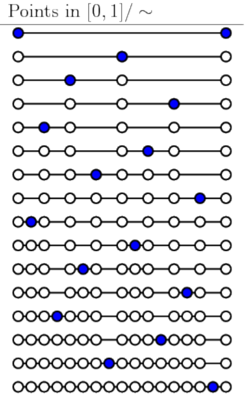
Why Dispersion?

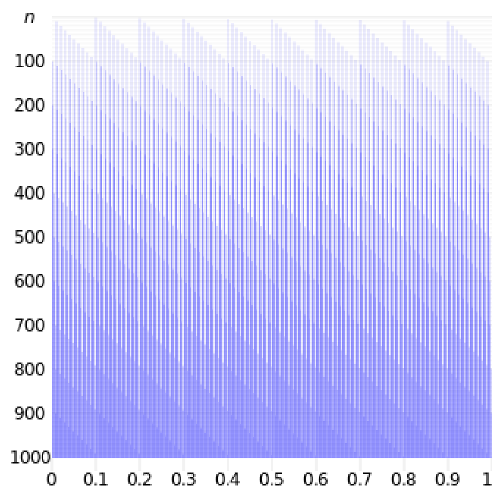
- **Low-discrepancy** sequences are also called **quasi-random** or **sub-random** sequences
  - Common use as a replacement of uniformly distributed random numbers
- Examples
  - Van der Corput sequence (for base = 10)

$$g_b(n) = \sum_{k=0}^{L-1} d_k(n) b^{-k-1}$$

$$\left\{\frac{1}{10}, \frac{2}{10}, \frac{3}{10}, \frac{4}{10}, \frac{5}{10}, \frac{6}{10}, \frac{7}{10}, \frac{8}{10}, \frac{9}{10}, \frac{1}{100}, \frac{11}{100}, \frac{21}{100}, \frac{31}{100}, \frac{41}{100}, \frac{51}{100}, \frac{61}{100}, \frac{71}{100}, \frac{81}{100}, \frac{91}{100}, \frac{2}{100}, \frac{12}{100}, \frac{22}{100}, \frac{32}{100}, \ldots\right\},$$

i	Naive Sequence	Binary	Reverse Binary	Van der Corput	Points in $[0,1]/\sim$
1	0	.0000	.0000	0	
2	1/16	.0001	.1000	1/2	
3	1/8	.0010	.0100	1/4	
4	3/16	.0011	.1100	3/4	
5	1/4	.0100	.0010	1/8	
6	5/16	.0101	.1010	5/8	
7	3/8	.0110	.0110	3/8	
8	7/16	.0111	.1110	7/8	
9	1/2	.1000	.0001	1/16	
10	9/16	.1001	.1001	9/16	
11	5/8	.1010	.0101	5/16	
12	11/16	.1011	.1101	13/16	
13	3/4	.1100	.0011	3/16	
14	13/16	.1101	.1011	11/16	
15	7/8	.1110	.0111	7/16	
16	15/16	.1111	.1111	15/16	





Deterministic Sampling

- Deterministic Sampling
  - Similar to discretization we saw in Discrete Motion Planning, but order of samples matters
- Sequences?
  - Van der Corput sequence – 1D
  - Halton sequence
    - **n-dimensional generalization** of van der Corput sequence
  - Hammersley sequence
    - Adaptation of Halton sequence that yields a **better distribution**. BUT **need to know number of samples** in advance.

Multi- vs. Single-Query Roadmaps

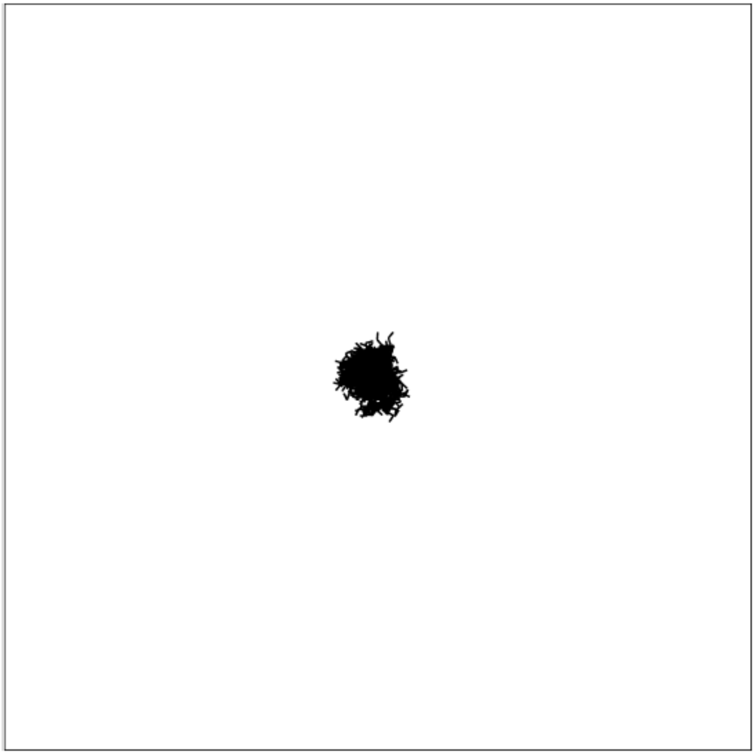
- Multi-query roadmaps
  - Pre-compute roadmap
  - Re-use roadmap for answering queries
  - The roadmap must cover the free space well
    - Why try to capture the connectivity of the whole space when all you need is one path?
- Single-query roadmaps
  - Dynamic environment
    - Compute a roadmap from scratch for each new query

Single-query Methods

- Key idea
  - Build a tree instead of a general graph.
  - The tree grows in C_{free}
    - Like PRM, captures some connectivity
    - Unlike PRM, only explores what is connected to q_{start}
- Algorithms:
  - Single-Query BiDirectional Lazy PRM (SBL-PRM)
  - Expansive Space Trees (EST)
  - **Rapidly-exploring Random Tree (RRT)**

Naïve Tree Algorithm

- Steps
  - Pick a node at random
  - Sample a new node near it
  - Grow tree from the random node to the new node



```
q_node = q_start
For i = 1 to NumberSamples
        q_rand = Sample near q_node
        Add edge e = (q_rand , q) if collision-free
        q_node = Pick random node of tree
```

Grow a RRT

- Steps
  - Grow a tree rooted at the starting configuration,
  - Randomly sample from the search space
  - For each sample, try to connect it to the nearest node of the tree
    - Success – add a new node
    - Fail – discard the sample

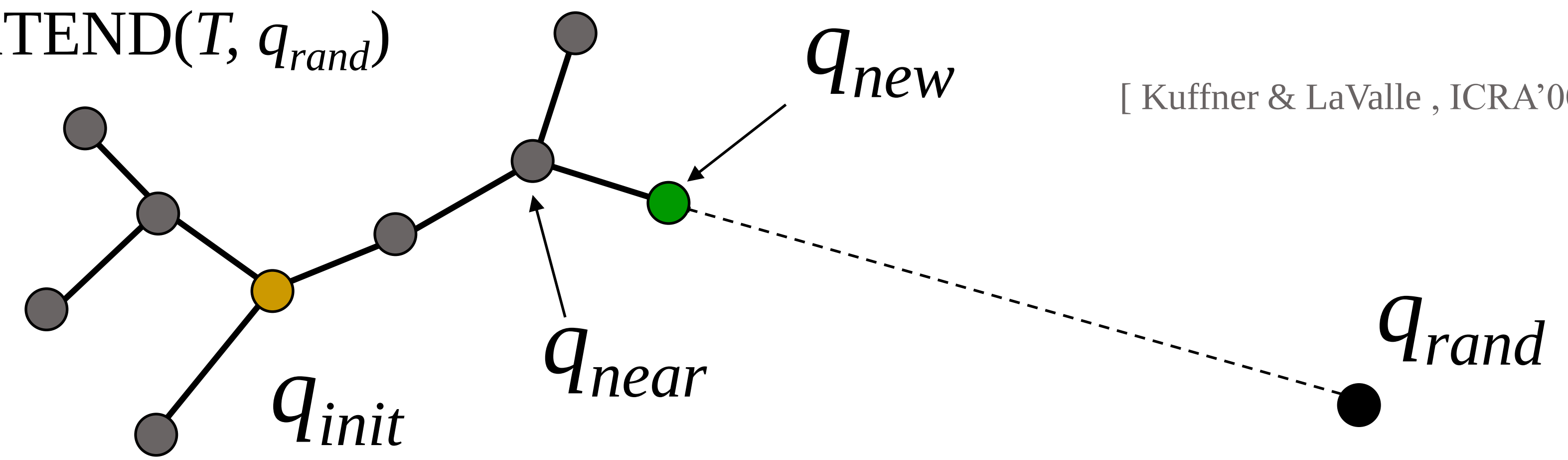
Basic RRT Algorithm

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

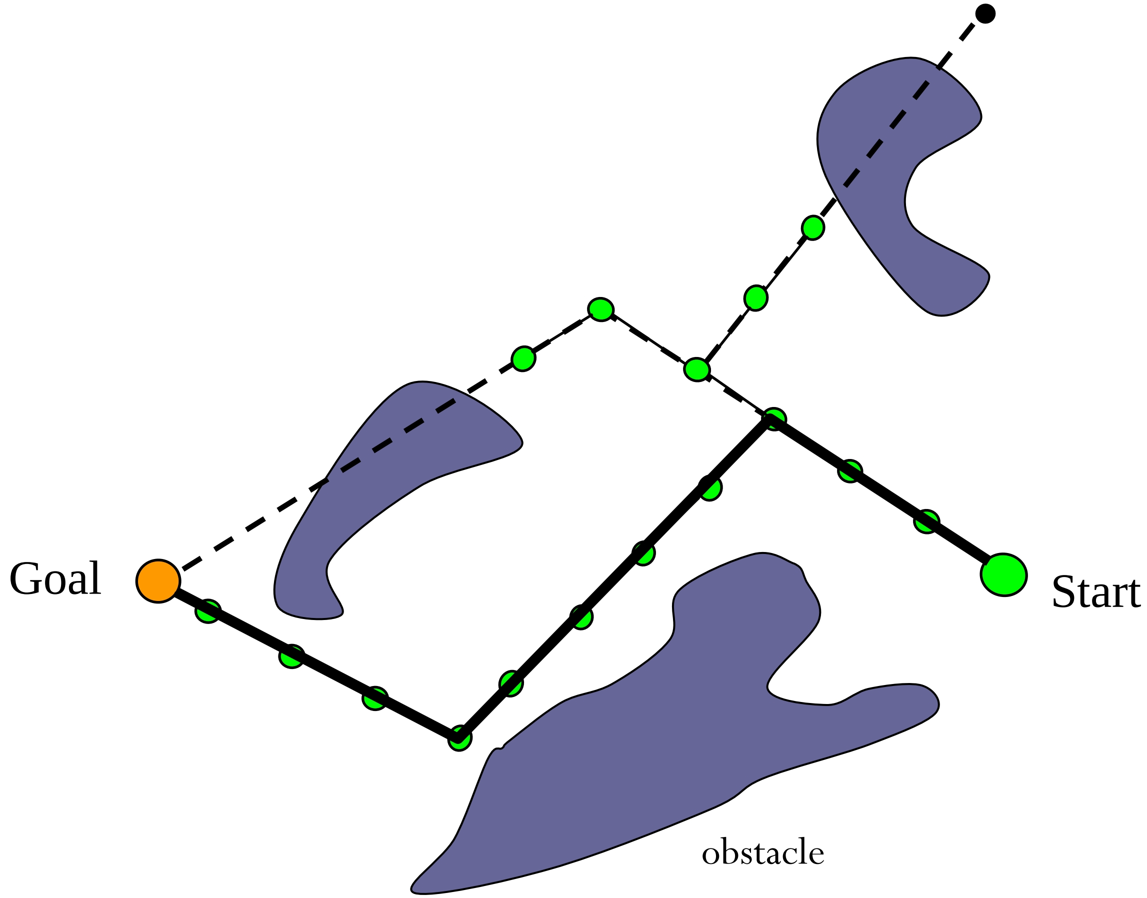
EXTEND(T, q_{rand})



[Kuffner & LaValle , ICRA'00]

RRT Growing in Empty Space

RRT with Obstacles and Goal bias

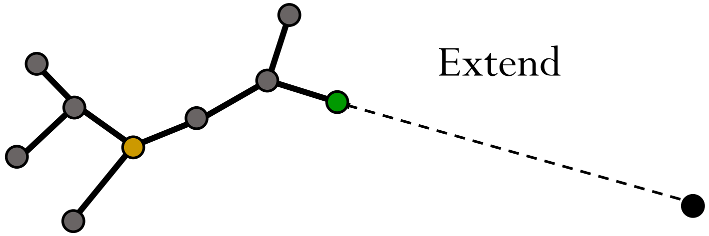


Sample Bias

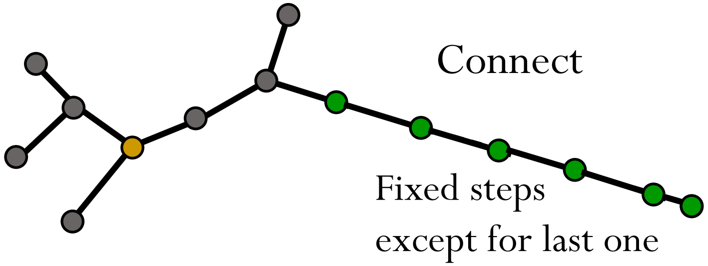
- Bias
  - Toward larger spaces
  - Toward goal
    - When generating a random sample, with some probability pick the goal instead of a random node when expanding
    - This introduces another parameter
    - 5-10% is the right choice [James Kuffner]
  - What happens if you set probability of sampling goal to 100%?

RRT Extension Types

- RRT-Extend
  - Take one step toward a random sample



- RRT-Connect
  - Step toward random sample until it is either
    - Reached
    - You hit an obstacle



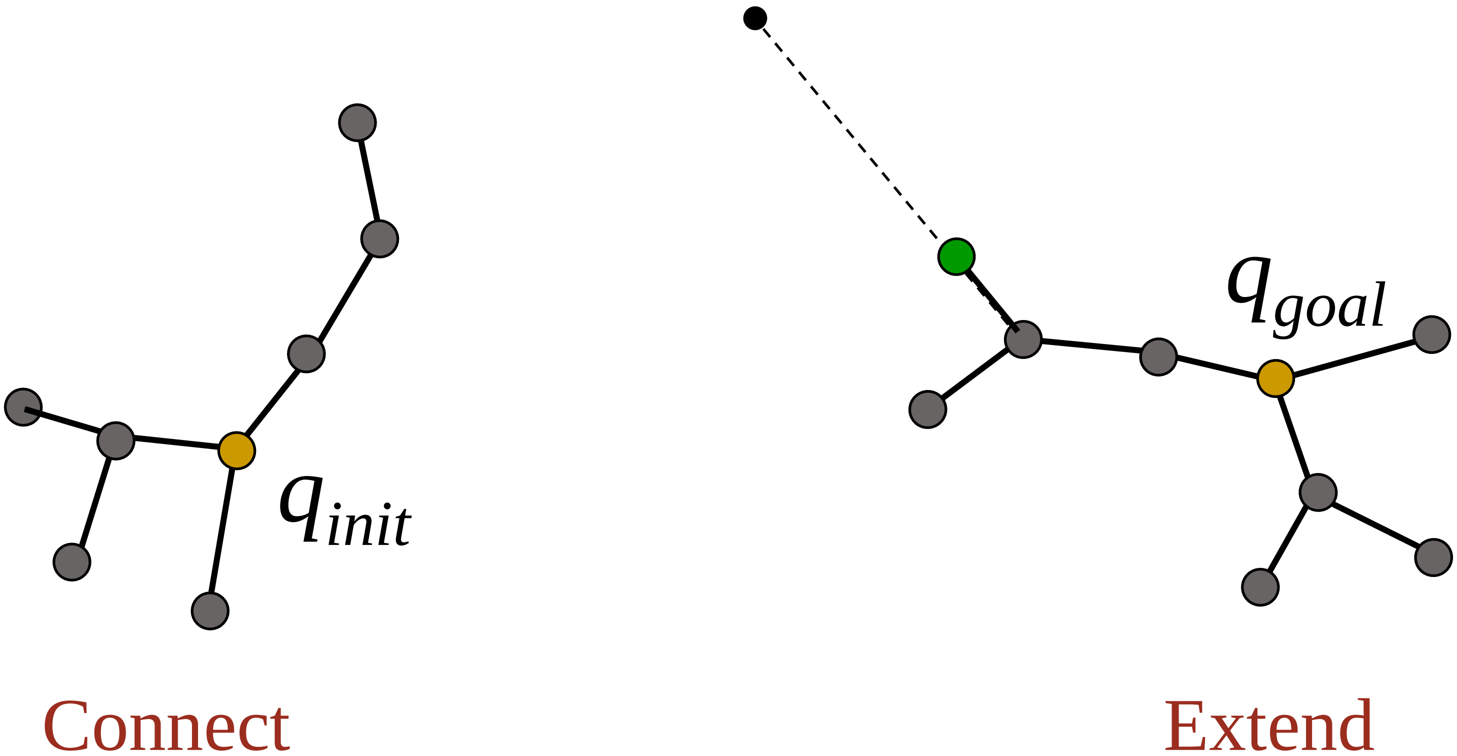
BiDirectional RRTs

- BiDirectional RRT
  - Grow trees from both start and goal
  - Try to get trees to connect to each other
  - Trees can both use Extend or both use Connect or one use Extend and one Connect
- BiDirectional RRT with Connect for both trees
  - Preferred, since this variant has only one parameter, the step size

Example of BiDirectional RRT

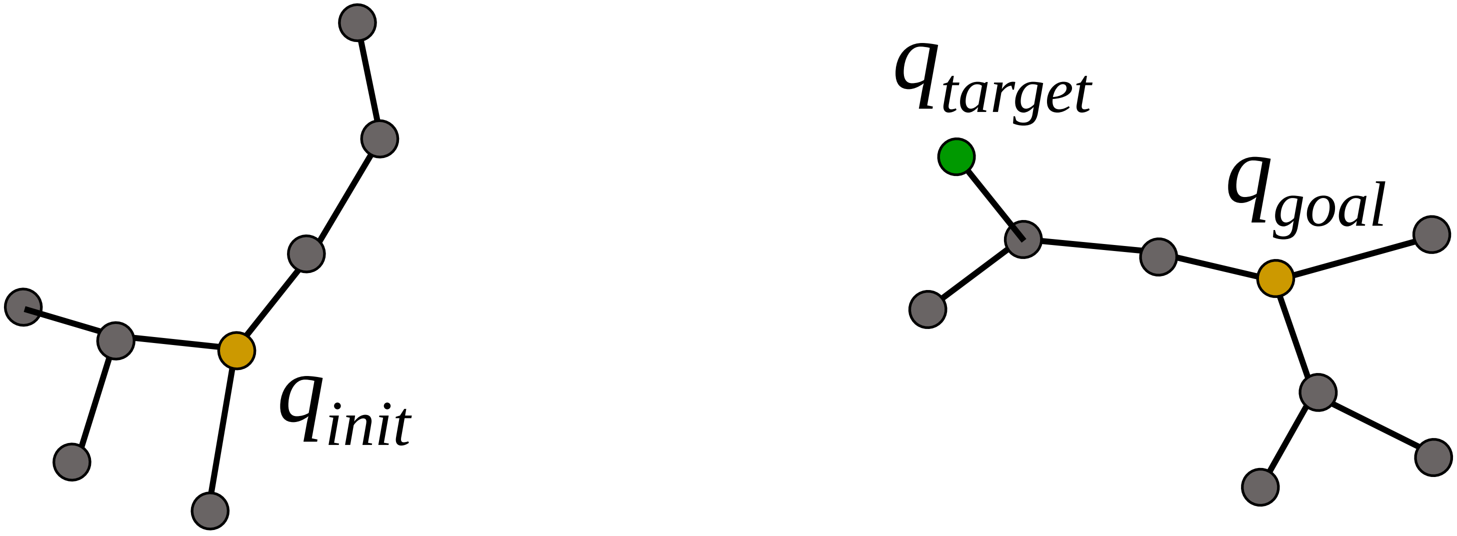
Example of BiDirectional RRT

1) One tree grown using random target



Example of BiDirectional RRT

2) New node becomes target for other tree

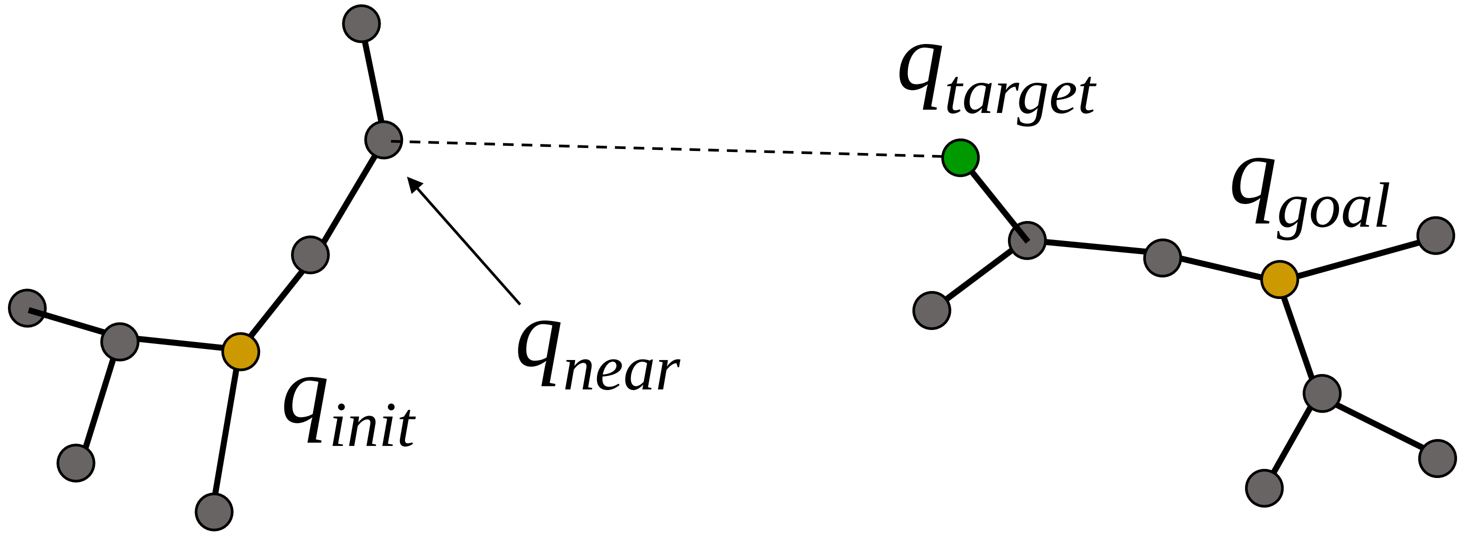


Connect

Extend

Example of BiDirectional RRT

3) Calculate node "nearest" to target

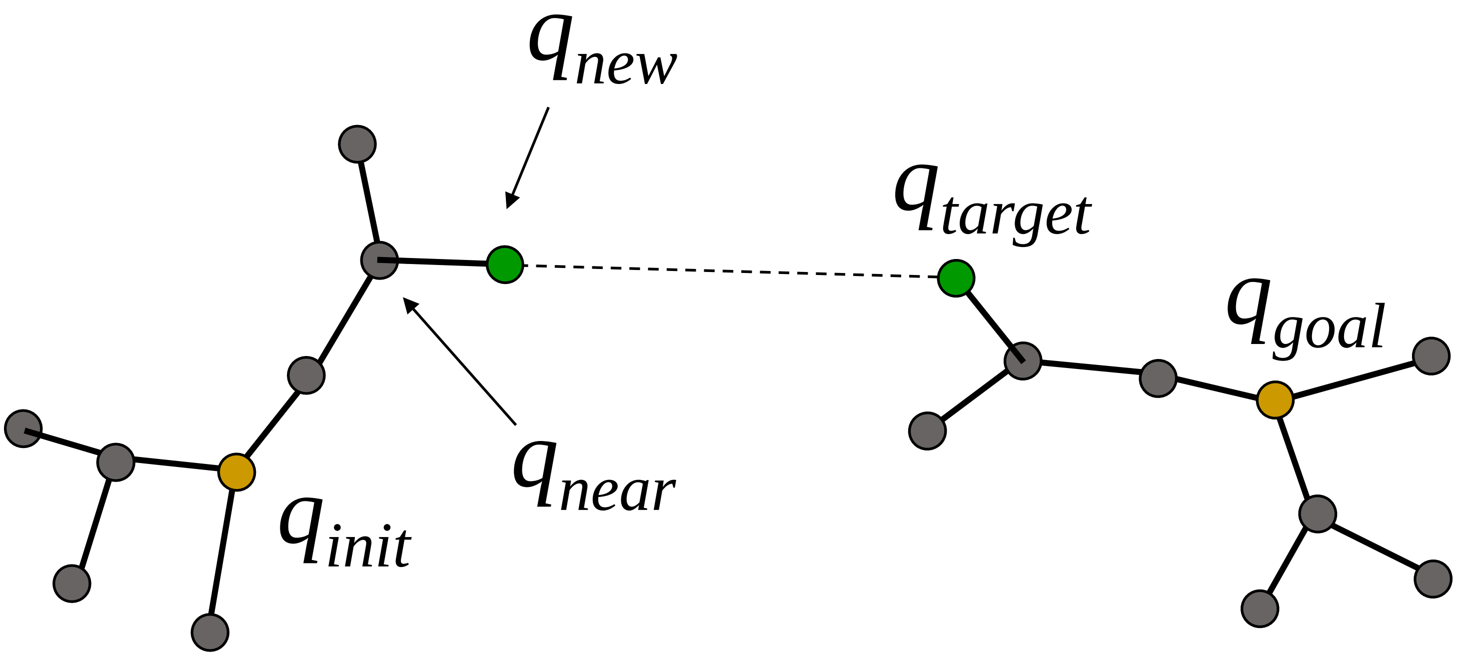


Connect

Extend

Example of BiDirectional RRT

4) Try to add new collision-free branch

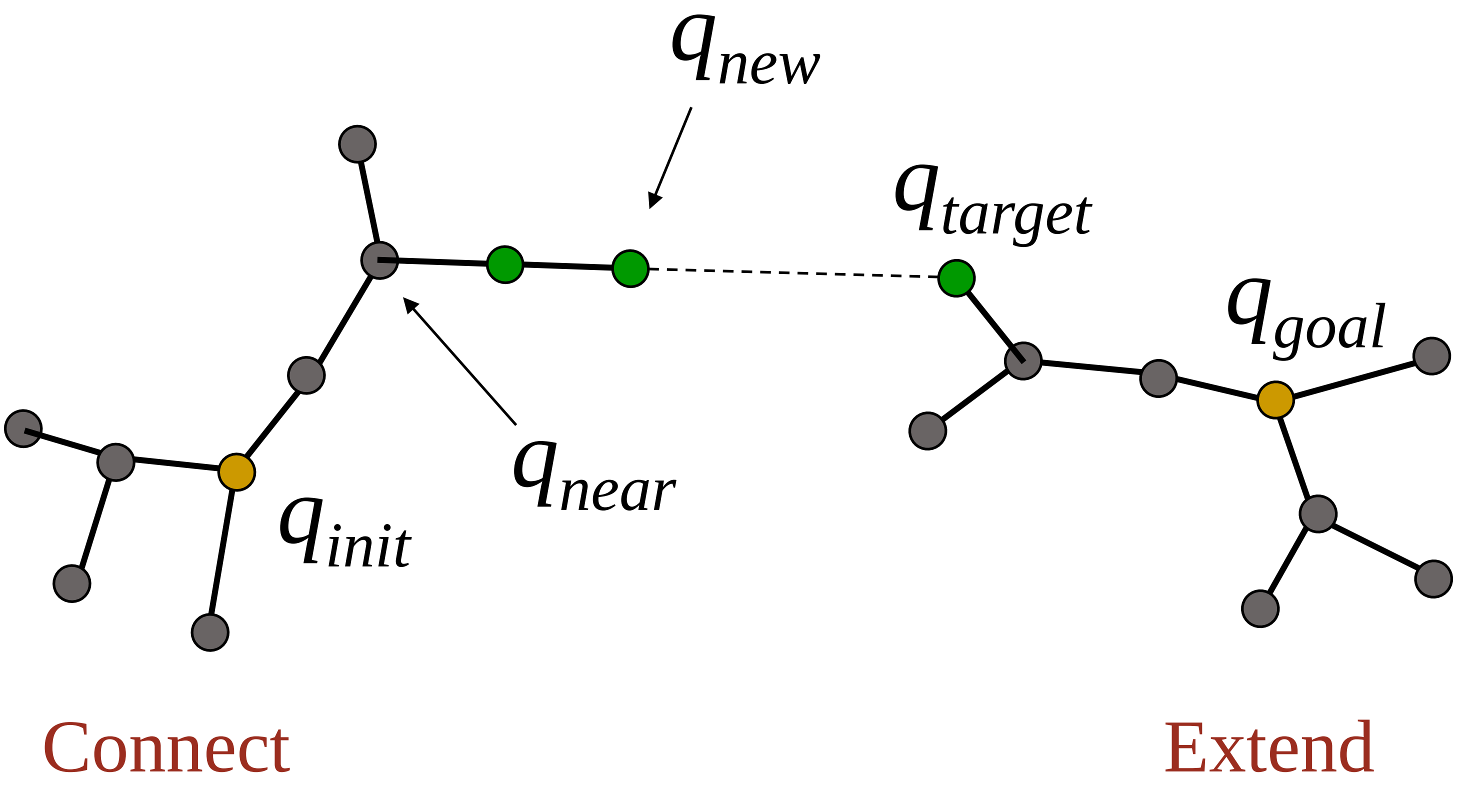


Connect

Extend

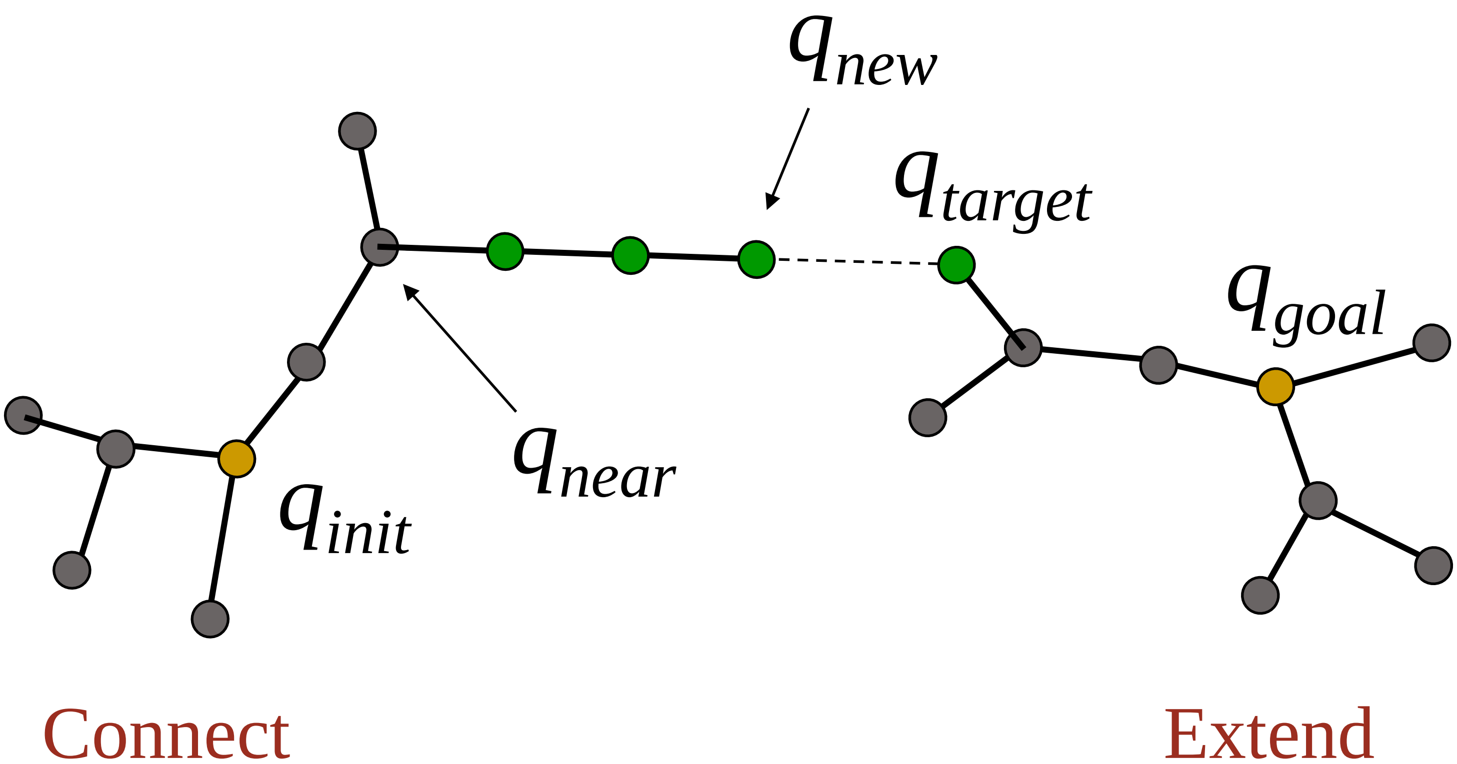
Example of BiDirectional RRT

5) If successful, keep extending branch



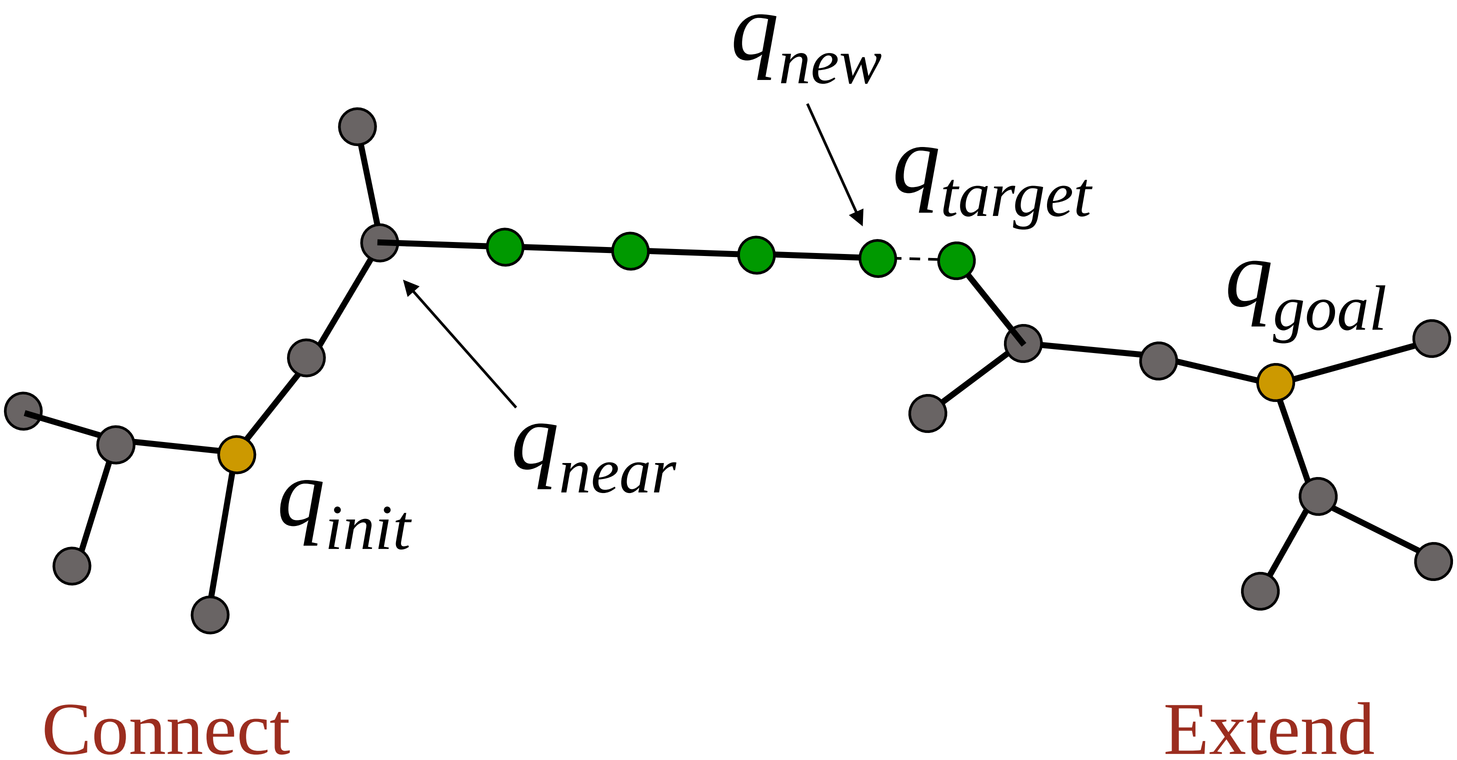
Example of BiDirectional RRT

5) If successful, keep extending branch



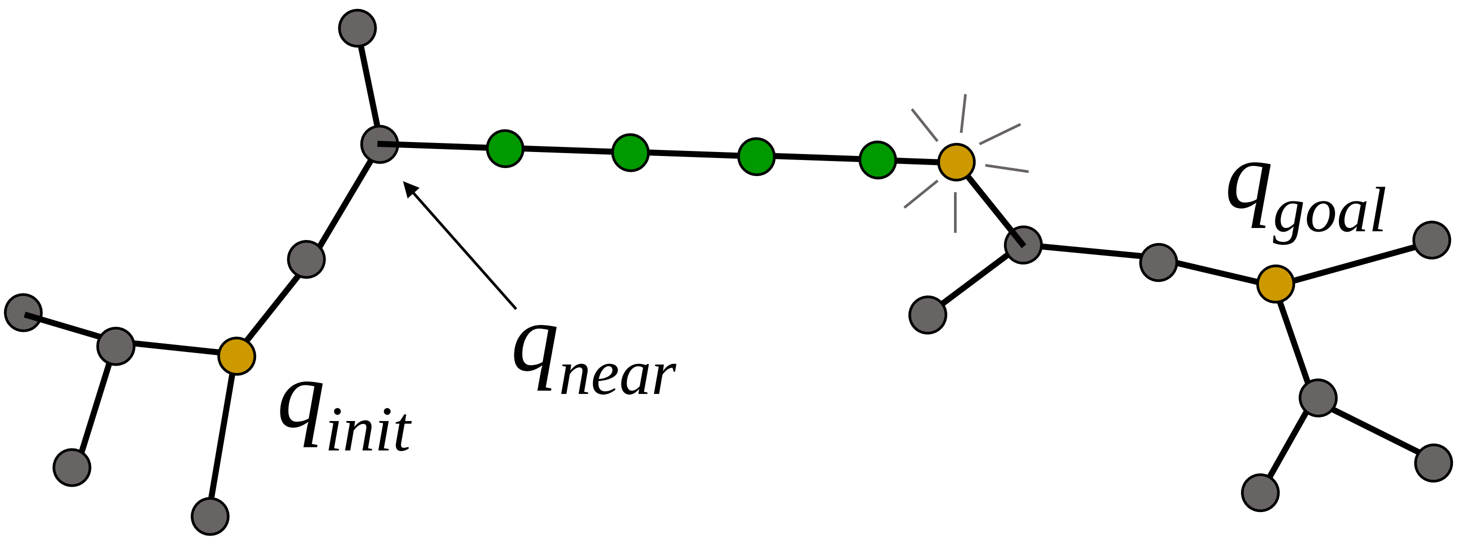
Example of BiDirectional RRT

5) If successful, keep extending branch



Example of BiDirectional RRT

6) Path found if branch reaches target

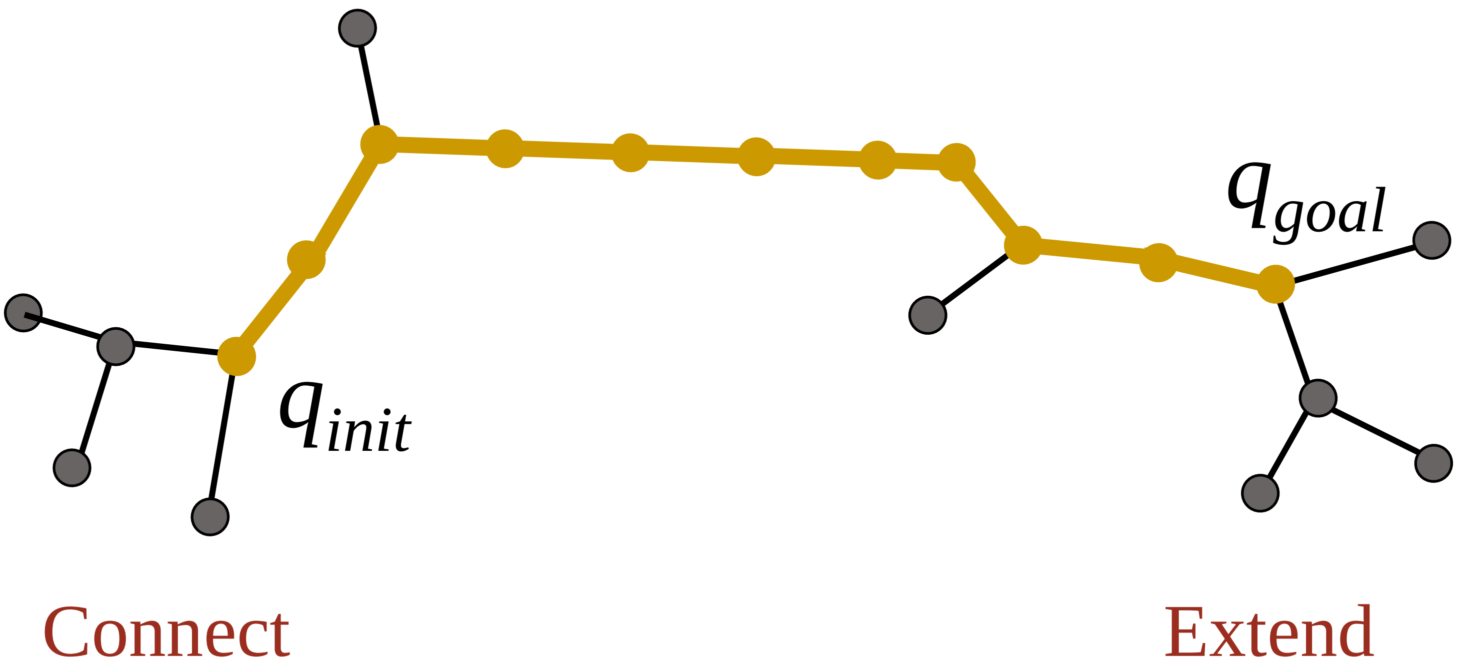


Connect

Extend

Example of BiDirectional RRT

7) Return path connecting start and goal



Tree Swapping and Balancing

- Tree Swapping and Balancing
- Some use Tree Balancing:

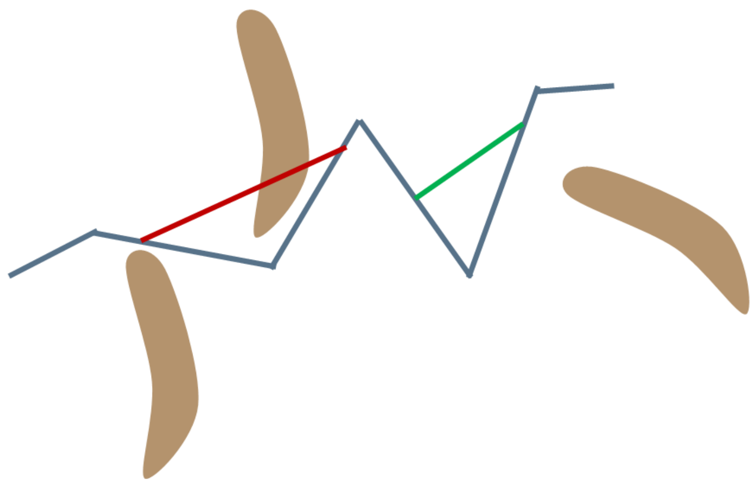
```
RDT_BALANCED_BIDIRECTIONAL(q_I, q_G)
T_a.init(q_I); T_b.init(q_G);
for i = 1 to K do
    q_n ← NEAREST(S_a, α(i));
    q_s ← STOPPING-CONFIGURATION(q_n, α(i));
    if q_s ≠ q_n then
        T_a.add_vertex(q_s);
        T_a.add_edge(q_n, q_s);
        q'_n ← NEAREST(S_b, q_s);
        q'_s ← STOPPING-CONFIGURATION(q'_n, q_s);
        if q'_s ≠ q'_n then
            T_b.add_vertex(q'_s);
            T_b.add_edge(q'_n, q'_s);
        if q'_s = q_s then return SOLUTION;
    if |T_b| > |T_a| then SWAP(T_a, T_b);
return FAILURE
```

- What is a situation where this would *help* performance?
- What is a situation where this would *hurt* performance?

Path Smoothing/Optimization

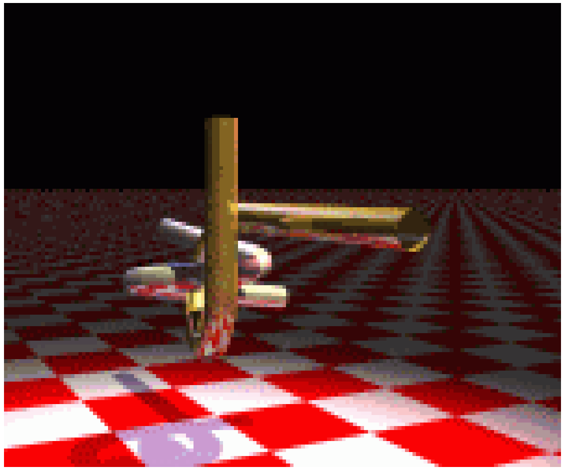
- RRTs produce notoriously bad paths
  - Not surprising since no consideration of path quality

- ALWAYS smooth/optimize the returned path
  - Many methods exists, e.g. shortcut smoothing (from previous lecture)



RRT Examples: The Alpha Puzzle

- VERY hard 6DOF motion planning problem (long, winding narrow passage)



- "*In 2001, it was solved by using a balanced bidirectional RRT, developed by James Kuffner and Steve LaValle. There are no special heuristics or parameters that were tuned specifically for this problem. On a current PC (circa 2003), it consistently takes a few minutes to solve*" –RRT website
- RRT became famous in large part because it was able to solve this puzzle

RRT Analysis

The limit of the distribution of vertices

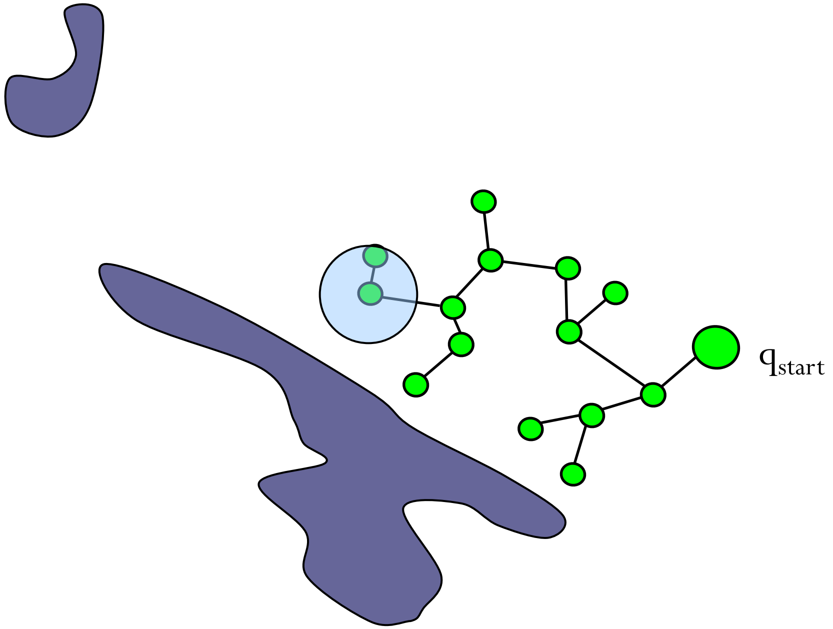
- THEOREM: $\boldsymbol{X_k}$ converges to $\boldsymbol{X}$ with probability 1 as time goes to infinity
  - $\boldsymbol{X_k}$: The RRT vertex distribution at iteration k
  - $\boldsymbol{X}$: The distribution used for generating samples
- If using uniform distribution,
  - Tree nodes converge to the free space

Probabilistic Completeness

- Definition:
  - A path planner is ***probabilistically complete*** if, given a solvable problem, the probability that the planner solves the problem goes to 1 as time goes to infinity.

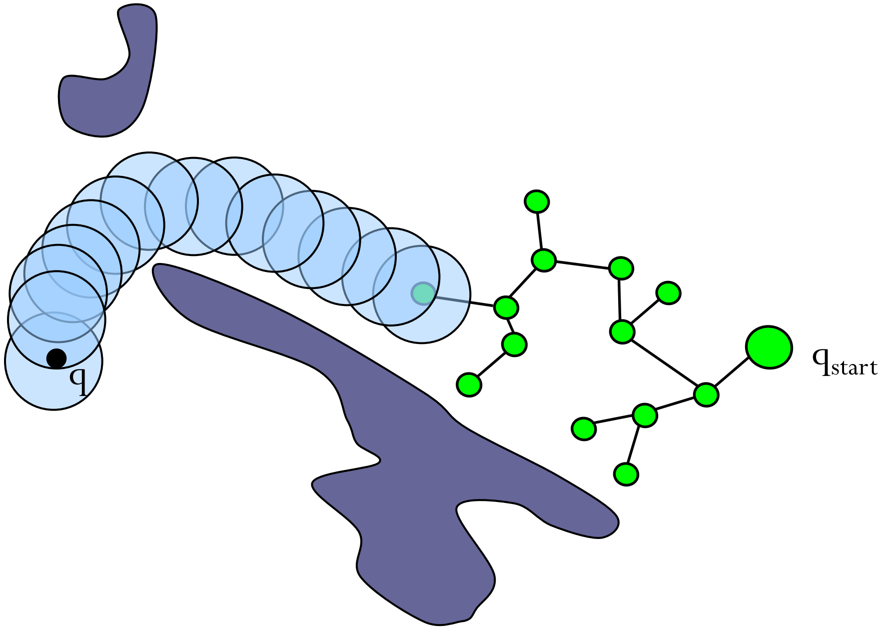
- Will RRT explore the whole space?

Proof of RRT Probabilistic Completeness



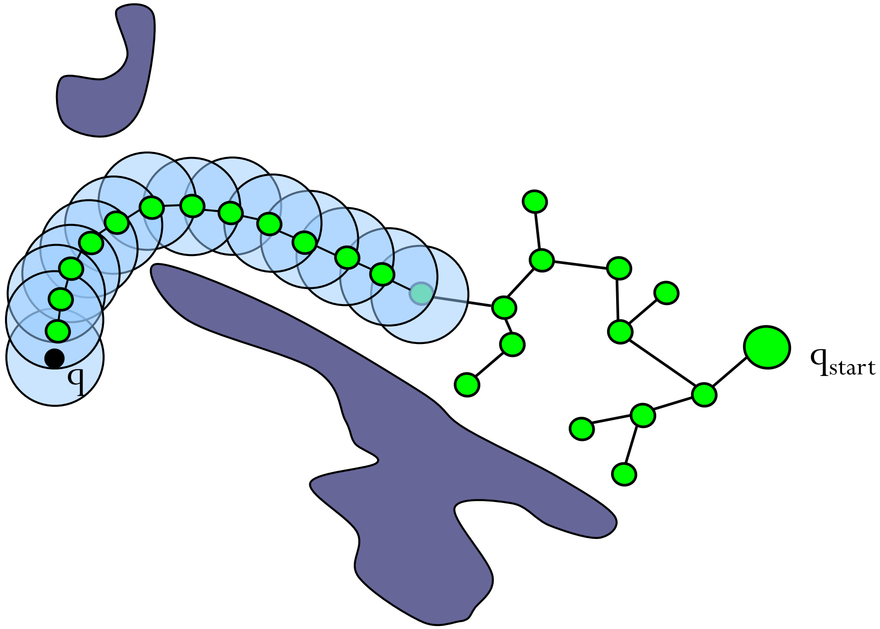
Kuffner and LaValle, ICRA, 2000

Proof of RRT Probabilistic Completeness



Kuffner and LaValle, ICRA, 2000

Proof of RRT Probabilistic Completeness



Kuffner and LaValle, ICRA, 2000

Probabilistic Completeness

- As the RRT reaches all of Q_{free},
  - The probability that q_{rand} immediately becomes a new vertex approaches 1.
- So, is RRT probabilistically complete?

Sampling-Based Planning

- The good:
  - Provides fast **feasible** solution
  - Popular methods have **few** parameters
  - Works on **practical** problems
  - Works in **high**-dimensions

Sampling-Based Planning

- The bad:
  - No quality guarantees on paths quality
    - In practice: smooth/optimize path afterwards
  - No termination when there is no solution
    - In practice: set an arbitrary timeout
  - Probabilistic completeness is a weak property
    - Completeness in high-dimensions is impractical

Readings

- Read "Non-holonomic motion planning guide" linked on class webpage