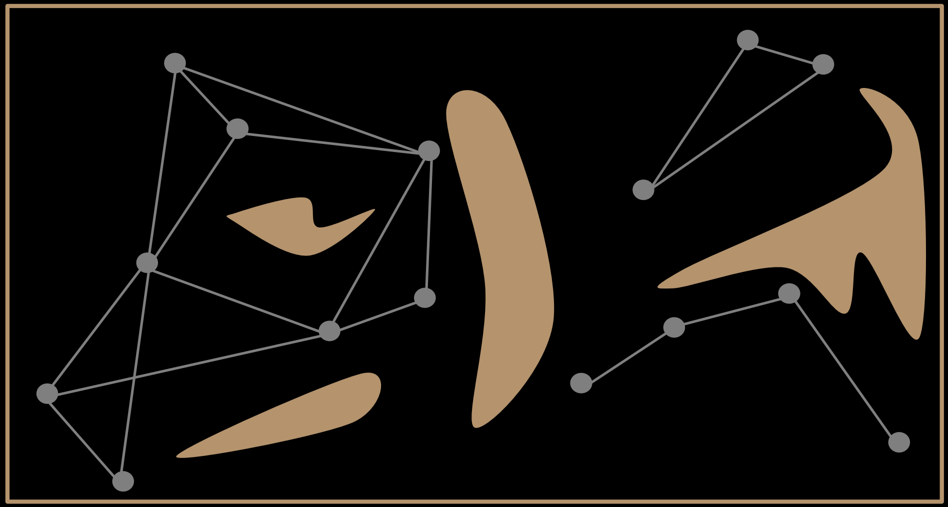
# Sampling-based Planning 2

A lot of Material from Howie Choset, Nancy Amato, Sujay Bhattacharjee, G.D. Hager, S. LaValle, J. Kuffner

# 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

- Most common is uniform random sampling
  - The bigger the area, the more likely it will be sampled
  - Problem: Narrow passages

- Are narrow passages inherently bad?
  - Does A* running on a 2D grid have problems with narrow passages?

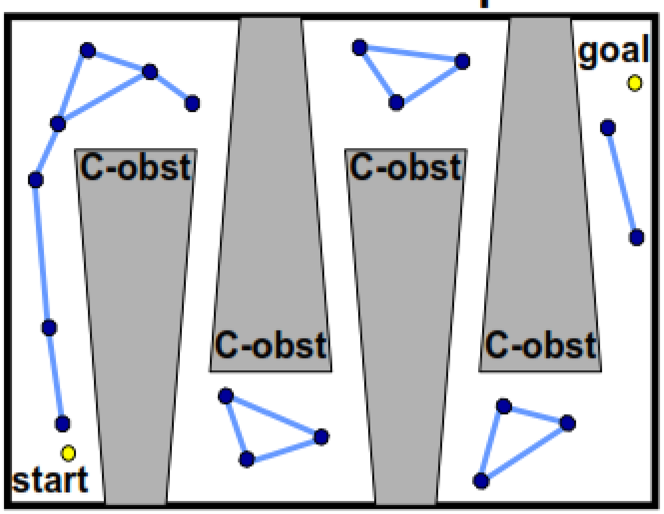
# OBPRM: An Obstacle-Based PRM

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

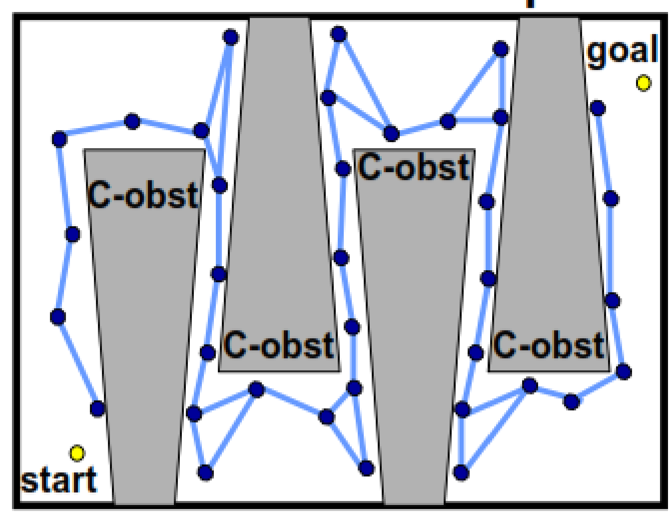
**To Navigate Narrow Passages we must sample in them**

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

**PRM Roadmap**



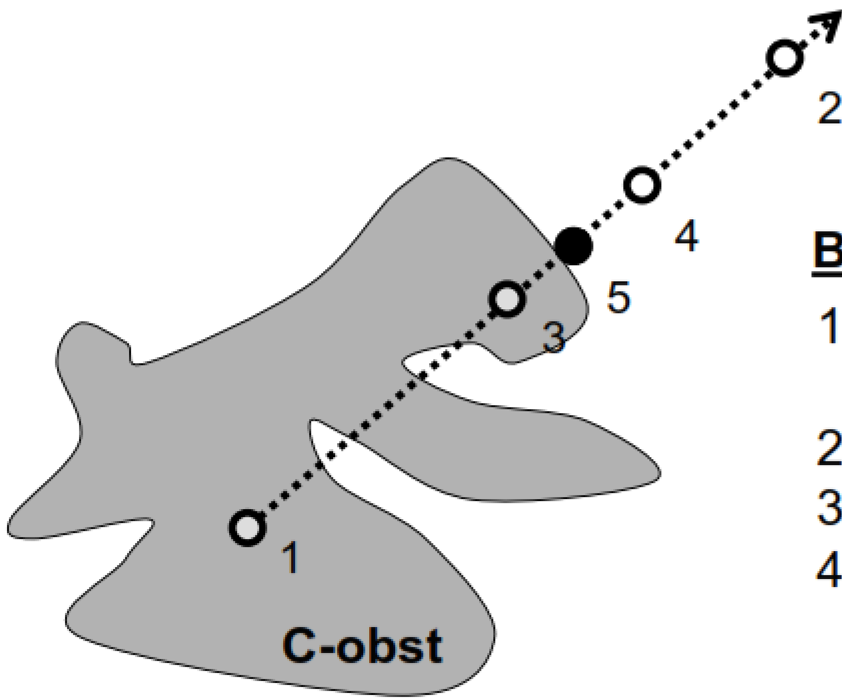

**OBPRM Roadmap**




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

- we cannot explicitly construct the C-obstacles...

# OBPRM: Finding Points on C-obstacles

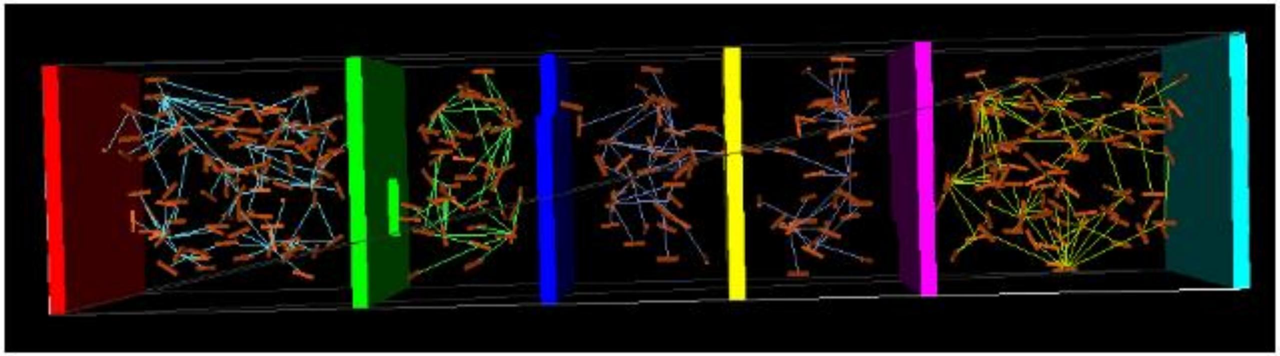



**Basic Idea (for workspace obstacle S)**

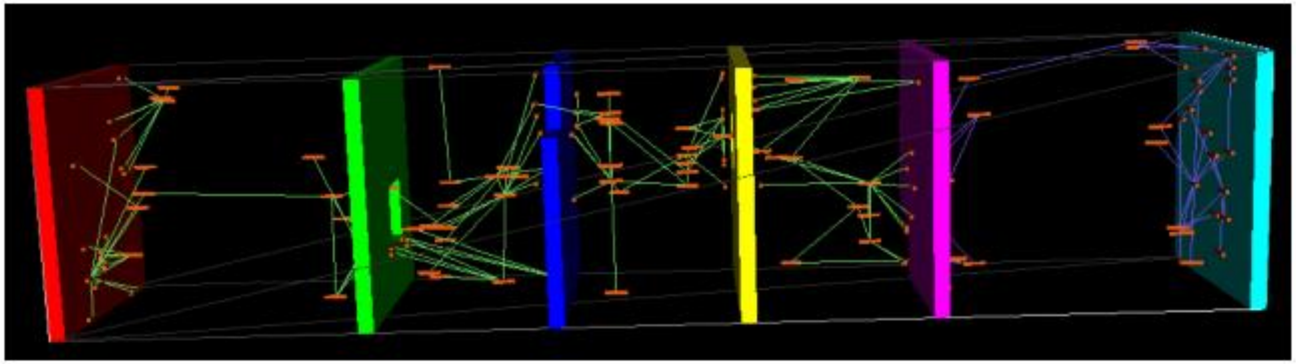
1. Find a point in S's C-obstacle (robot placement colliding with S)
2. Select a random direction in C-space
3. Find a free point in that direction
4. Find boundary point between them using binary search (collision checks)

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

# PRM vs OBPRM Roadmaps



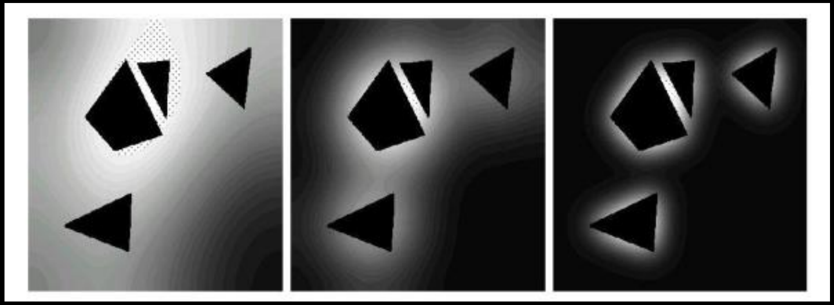
**PRM**
- 328 nodes
- 4 major CCs



**OBPRM**
- 161 nodes
- 2 major CCs

# Sampling strategies: Gaussian

- Gaussian sampler
  - Find a $q_1$ at random
  - 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

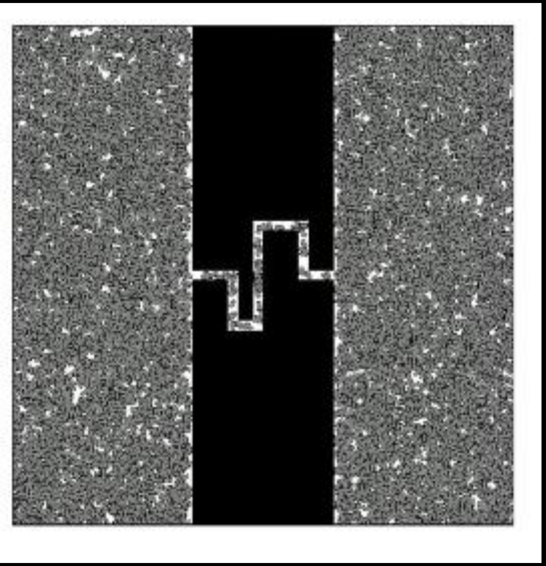


Sampling distribution for varying Guassian width
(width decreasing from left to right)

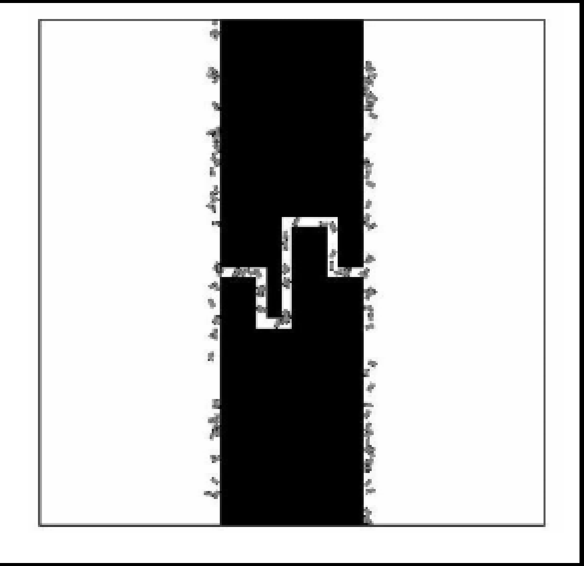
Boor, Valérie, Mark H. Overmars, and A. Frank van der Stappen. "The gaussian sampling strategy for probabilistic roadmap planners." *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*. Vol. 2. IEEE, 1999.

# Sampling Strategies: Gaussian

- Performs well in narrow passages



Uniform Random Sampling



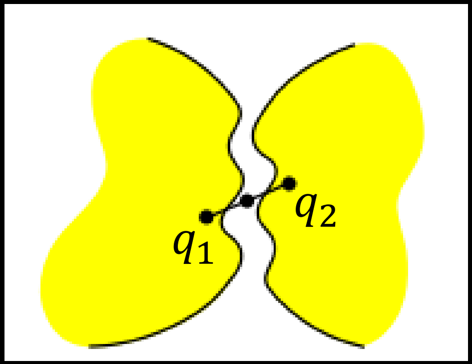
Gaussian Sampling

# Sampling Strategies

- Can we come up with a case where obstacle-biased sampling is worse than uniform random sampling?

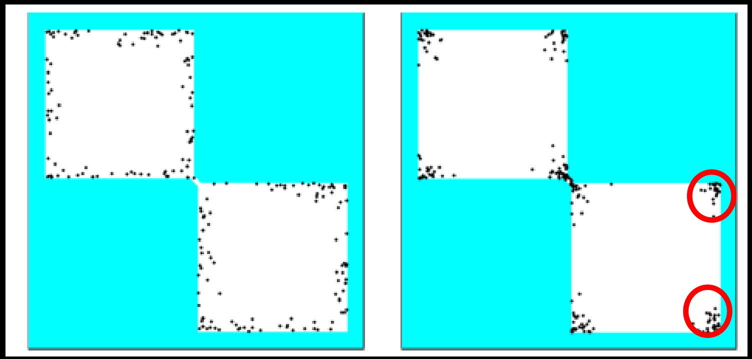
# Sampling Strategies: Bridge

- 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.

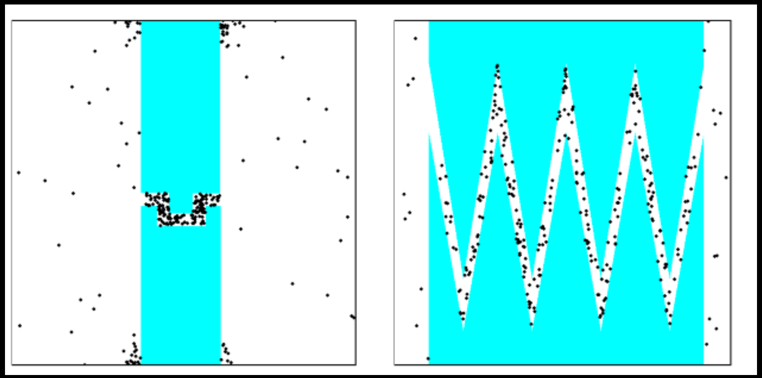
# Sampling Strategies: Bridge



Gaussian Sampling

Bridge Sampling

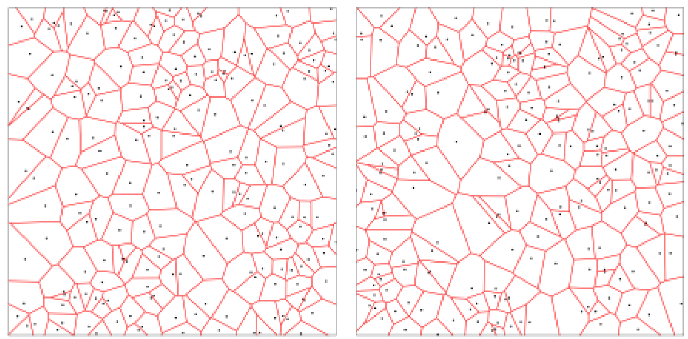
What's going on at the corners?



Bridge Sampling performs well in narrow passages

# Sampling Strategies: Deterministic

- 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.

  - Can we do better?

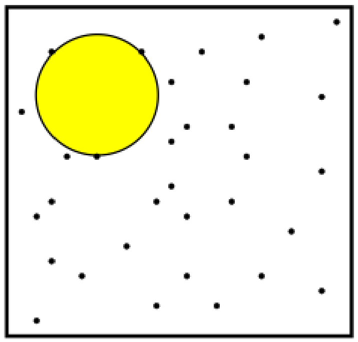
# Sampling Strategies: Deterministic

- 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*, $\rho$) is a metric space ($\rho$ is a distance metric)

In English: the radius of the largest empty ball



(a) $L_2$ dispersion

# Sampling Strategies: Deterministic

- What do we care about?
  - Discrepancy

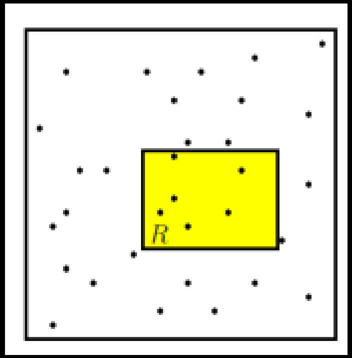
$$D(P, \mathcal{R}) = \sup_{R \in \mathcal{R}} \left\{ \left| \frac{|P \cap R|}{k} - \frac{\mu(R)}{\mu(X)} \right| \right\}$$

*P* is a finite set of points, $k = |P|$

*R* is the range space (set of axis-aligned boxes)

*X* is a metric space, $\mu$ is a measure of volume

- In English: The largest volume estimation error that can be obtained over all sets in $\mathcal{R}$
- Each term captures how well *P* can be used to estimate the volume of *R*.
  - Example: If $\mu(R)$ is 1/2 of $\mu(X)$, then we want 1/2 of the points in *P* to be in *R*.




Discrepancy

# Sampling Strategies: Deterministic

- *Deterministic* Sampling
  - Similar to discretization we saw in Discrete Motion Planning, but *order* of samples matters
  - Example: van der Corput sequence

| $i$ | 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 | |

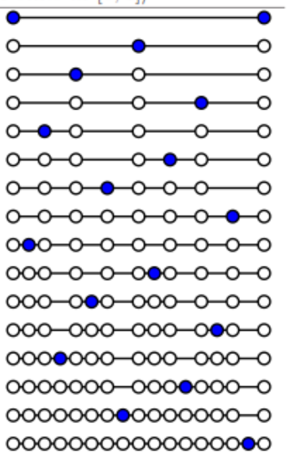
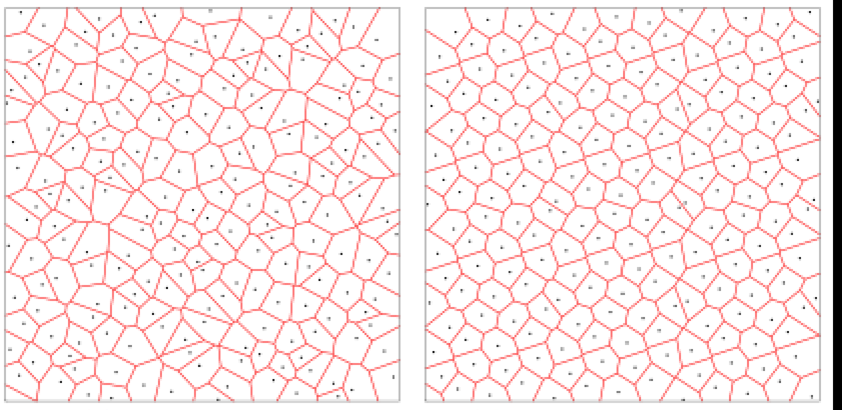


Figure 5.2: The van der Corput sequence is obtained by reversing the bits in the binary decimal representation of the naive sequence.

# Sampling Strategies: Deterministic

- 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.



(a) 196 Halton points (b) 196 Hammersley points

- See LaValle's book for formulas.

Break

# Rapidly-exploring Random Trees (RRTs)

# Single-query methods

- Motivation: Why try to capture the connectivity of the whole space when all you need is one path?

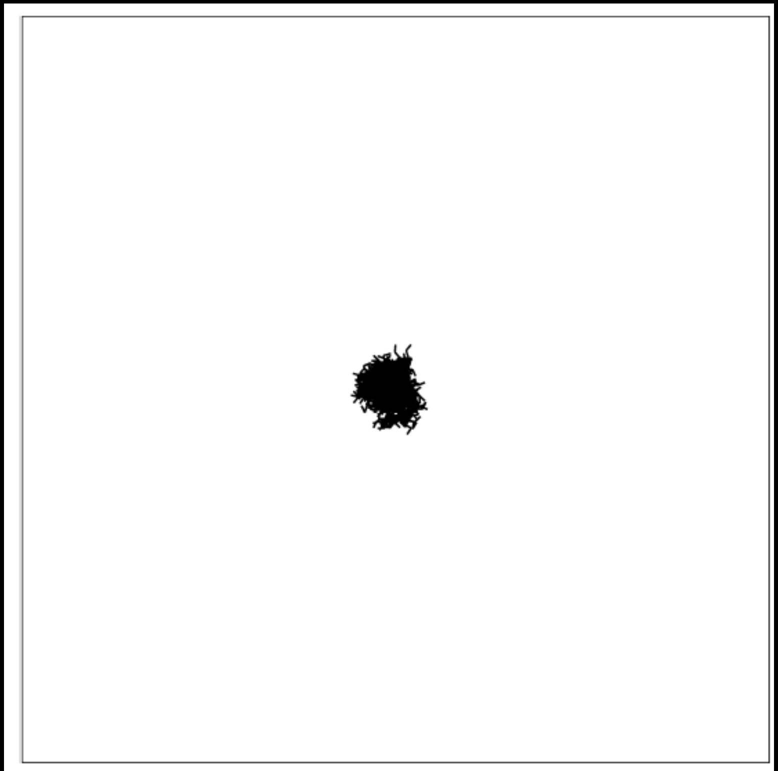
- Algorithms:
  - Single-Query BiDirectional Lazy PRM (SBL-PRM)
  - Expansive Space Trees (EST)
  - **Rapidly-exploring Random Tree (RRT)**

- 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 $q_{start}$

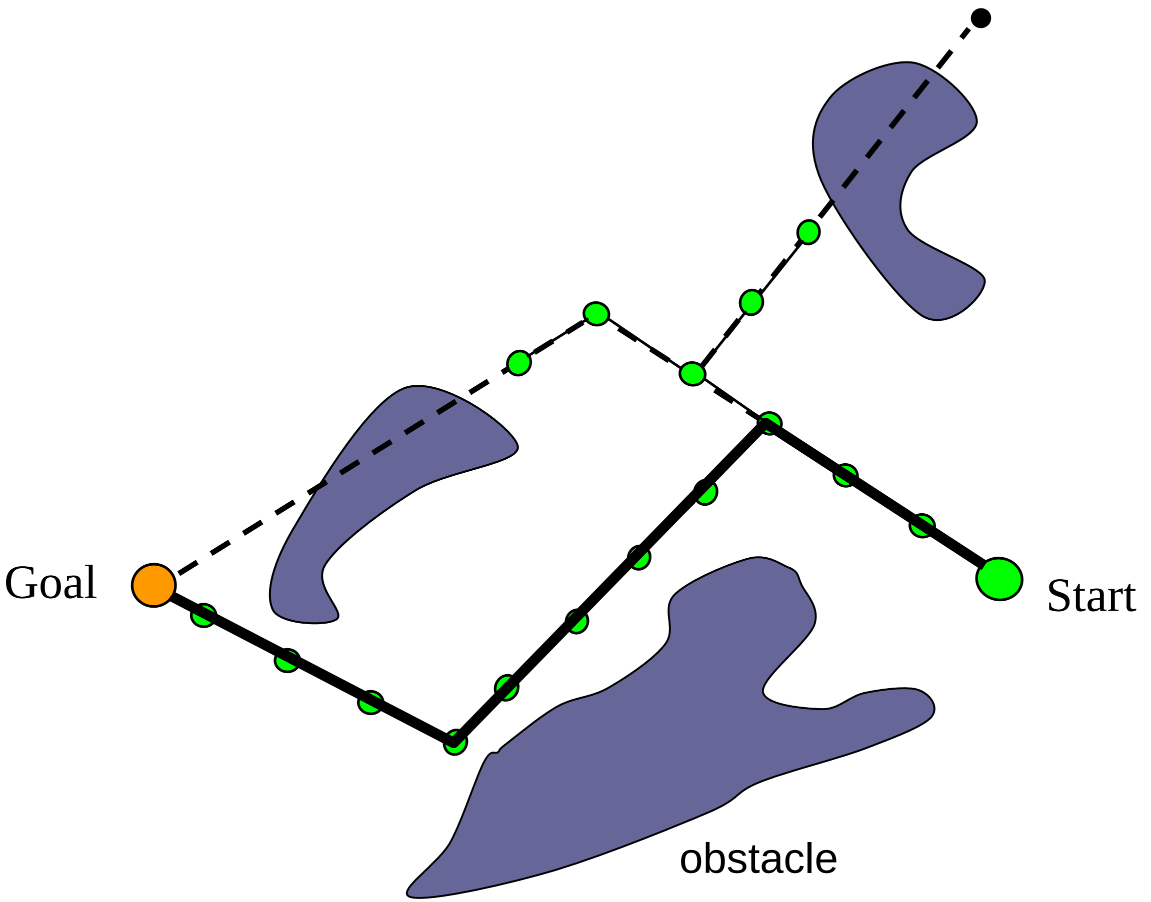
# Naïve Tree Algorithm

```
qnode = qstart
For i = 1 to NumberSamples
        qrand = Sample near qnode
        Add edge e = (qrand , qnode) if collision-free
        qnode = Pick random node of tree
```



# RRT Growing in Empty Space

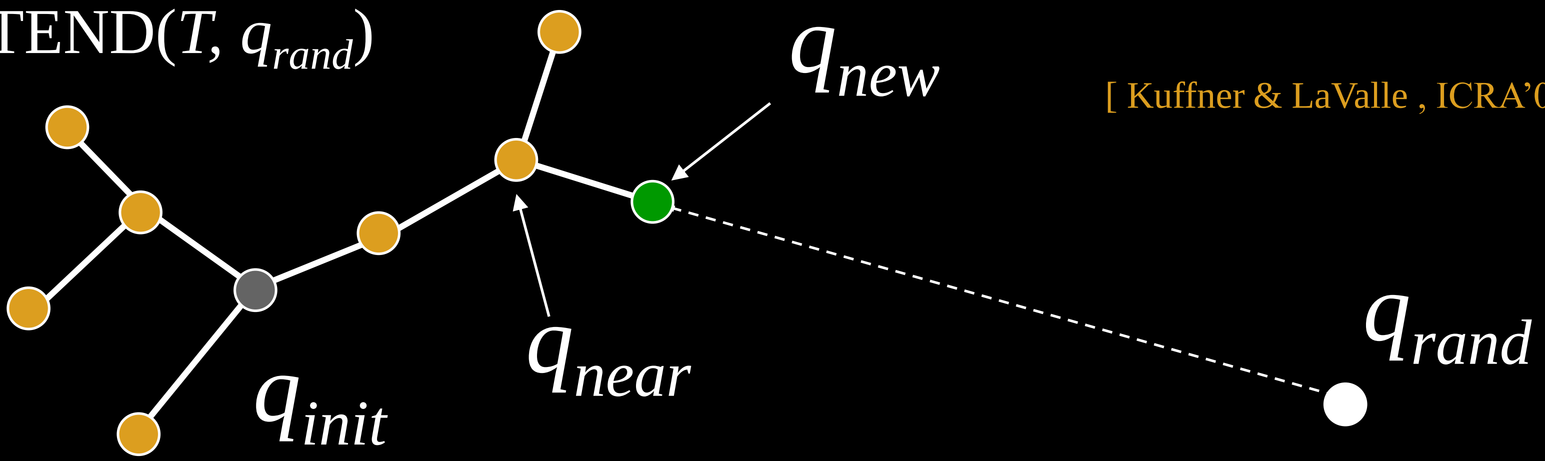
# RRT with obstacles and goal bias



# Path Planning with Rapidly-Exploring Random Trees (RRTs)

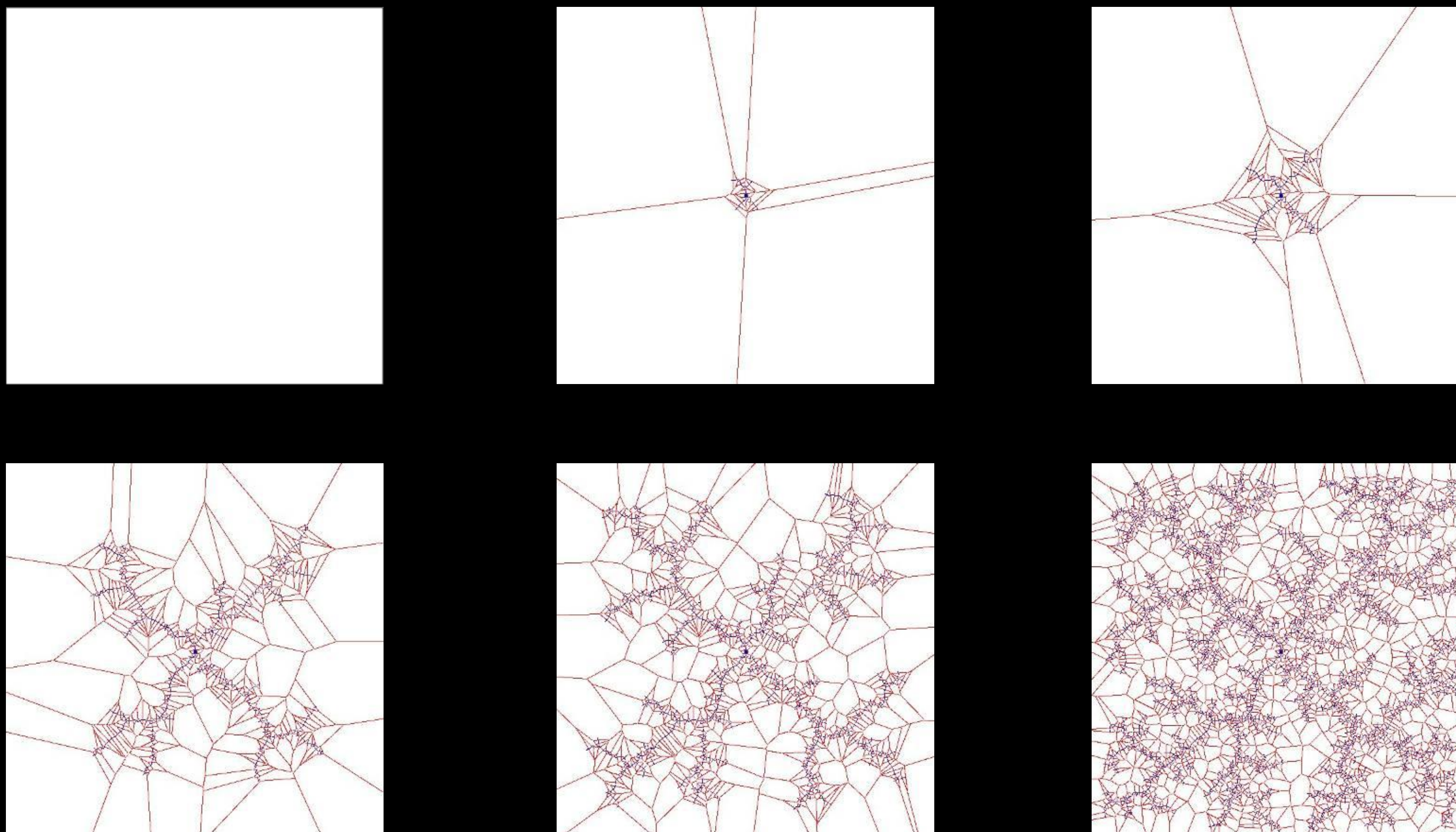
BUILD_RRT ($q_{init}$) {
  $T.init(q_{init})$;
  for $k$ = 1 to K do
    $q_{rand}$ = RANDOM_CONFIG();
    EXTEND($T$, $q_{rand}$)
}

EXTEND($T$, $q_{rand}$)




[ Kuffner & LaValle , ICRA'00]

# RRTs bias exploration toward large Voronoi regions



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

- What about obstacles?

# RRT Goal Biasing

- In “pure” form RRTs are great at filling space, but we need a path!

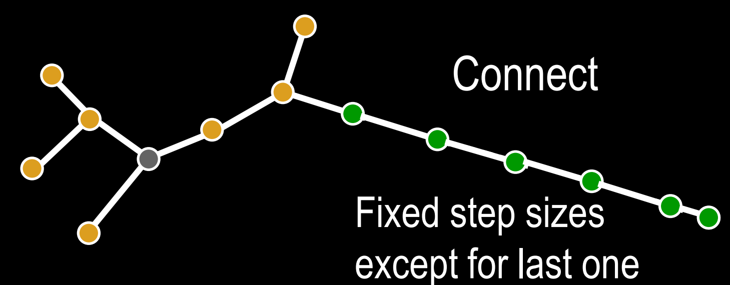
- Need to bias RRTs toward goal to produce a path
  - When generating a random sample, with some probability pick the goal instead of a random node
  - This introduces another parameter
  - James Kuffner’s experience is that 5-10% is the right choice

- 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



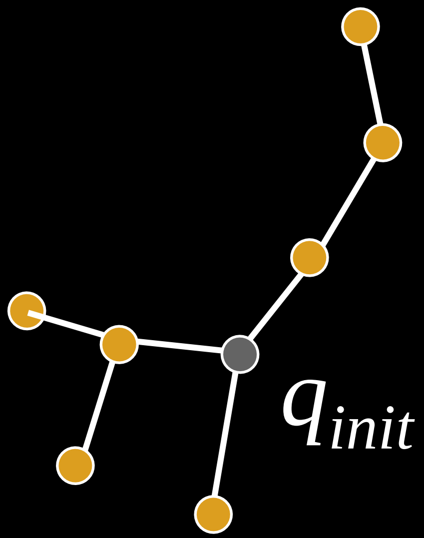

- Note that RRT-Connect is defined differently in some places online!

# BiDirectional RRTs

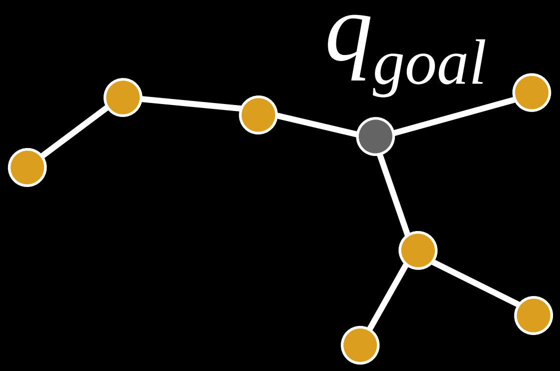
- BiDirectional RRT
  - Grow trees from both start and goal
  - Try to get trees to connect to each other
  - Trees can both use Extend or both use Connect or one use Extend and one Connect

- BiDirectional RRT with Connect for both trees is my favorite, I always try this first
  - This variant has only one parameter; the step size

# Example of BiDirectional RRT

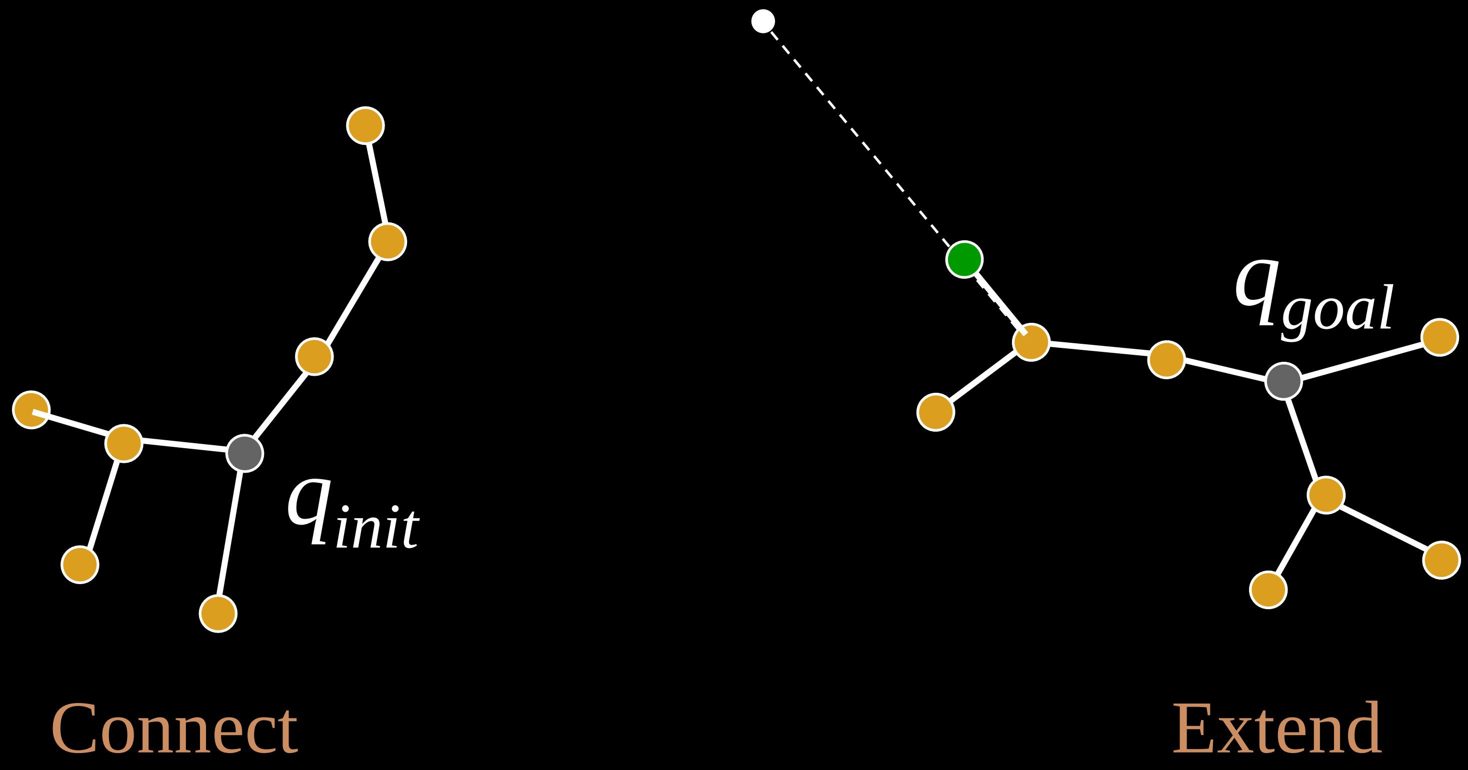



Connect

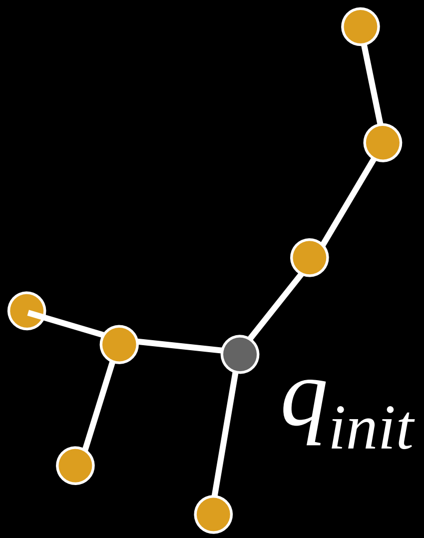



Extend

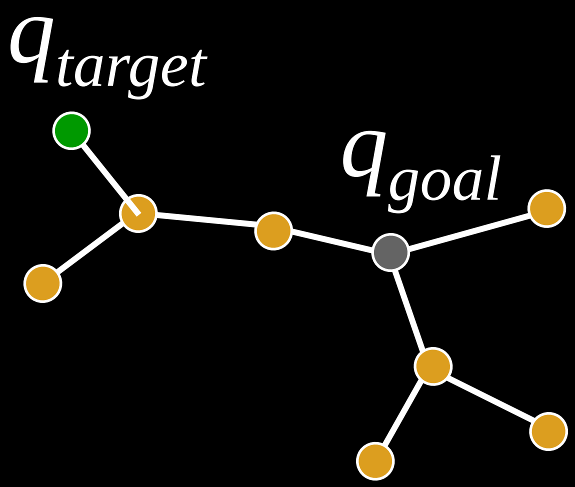
# 1) One tree grown using random target



# 2) New node becomes target for other tree

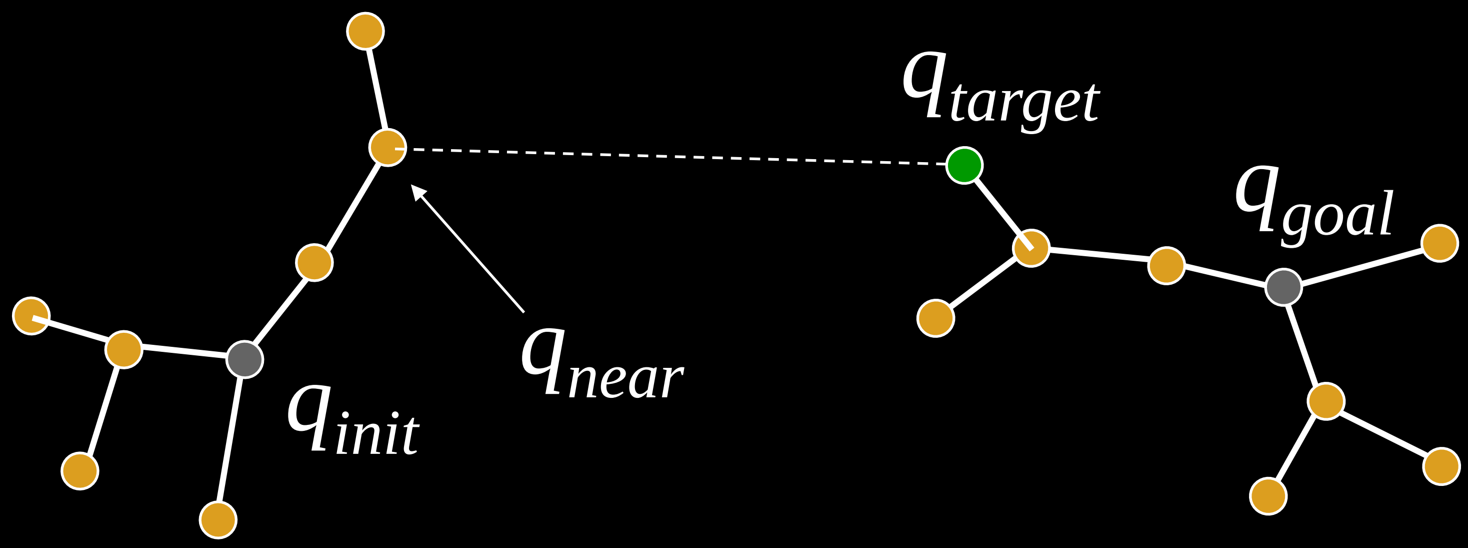



Connect




Extend

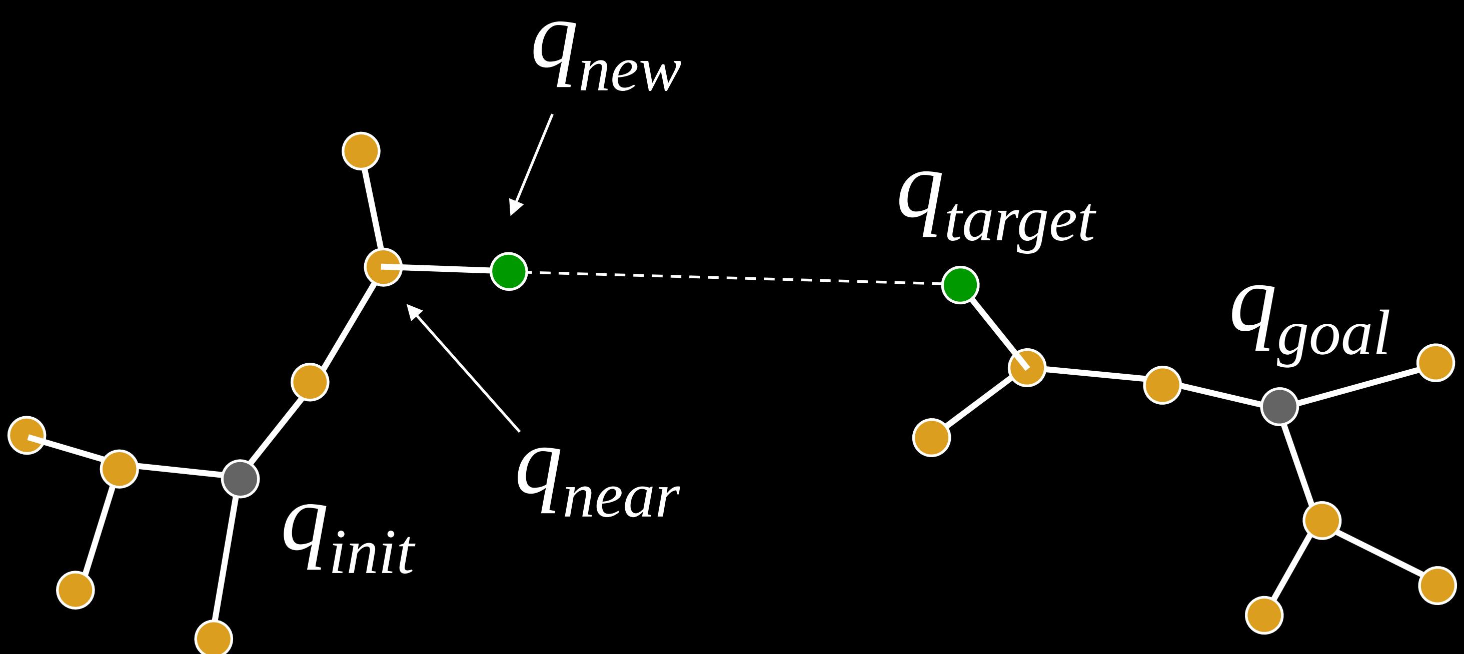
# 3) Calculate node "nearest" to target




Connect

Extend

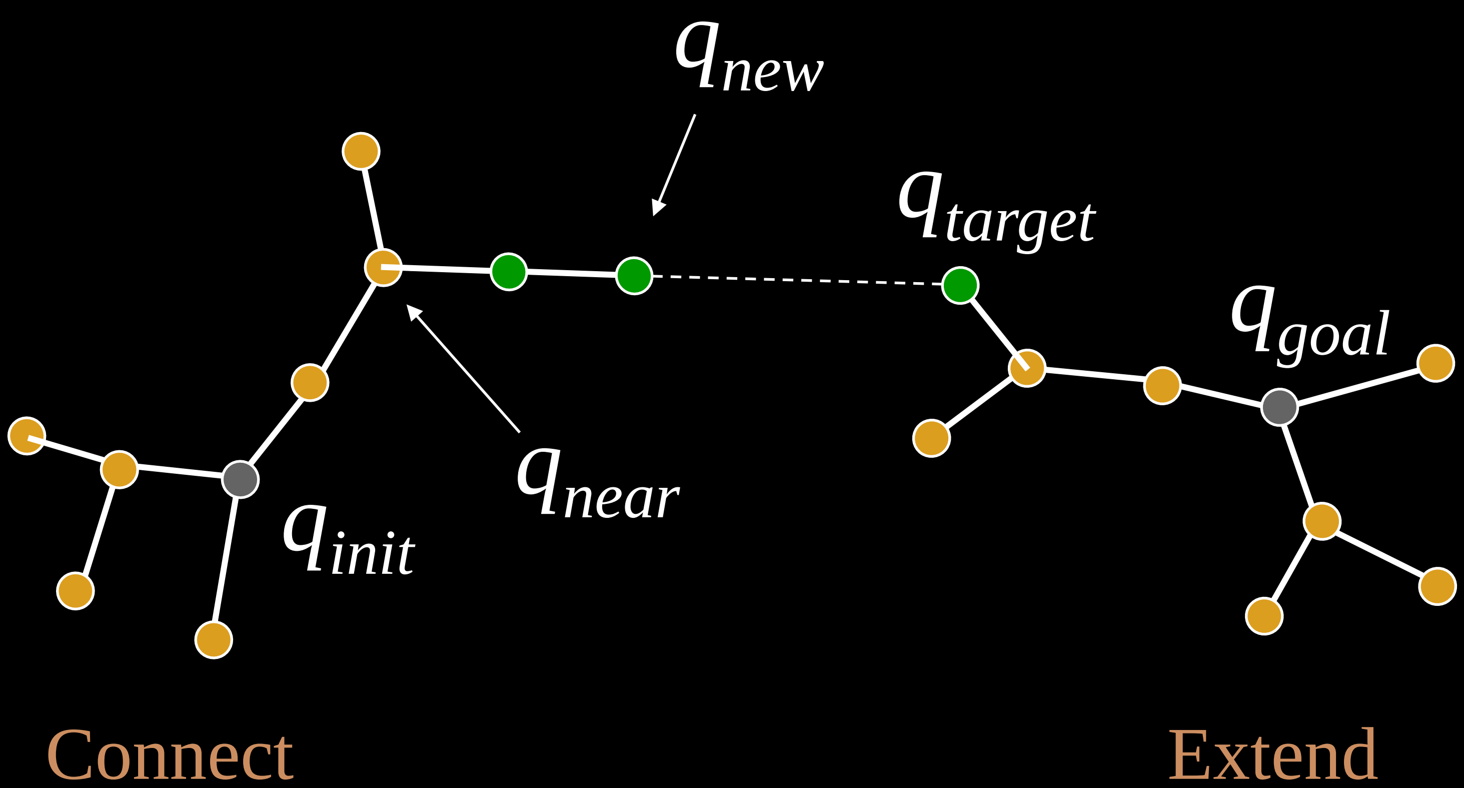
# 4) Try to add new collision-free branch



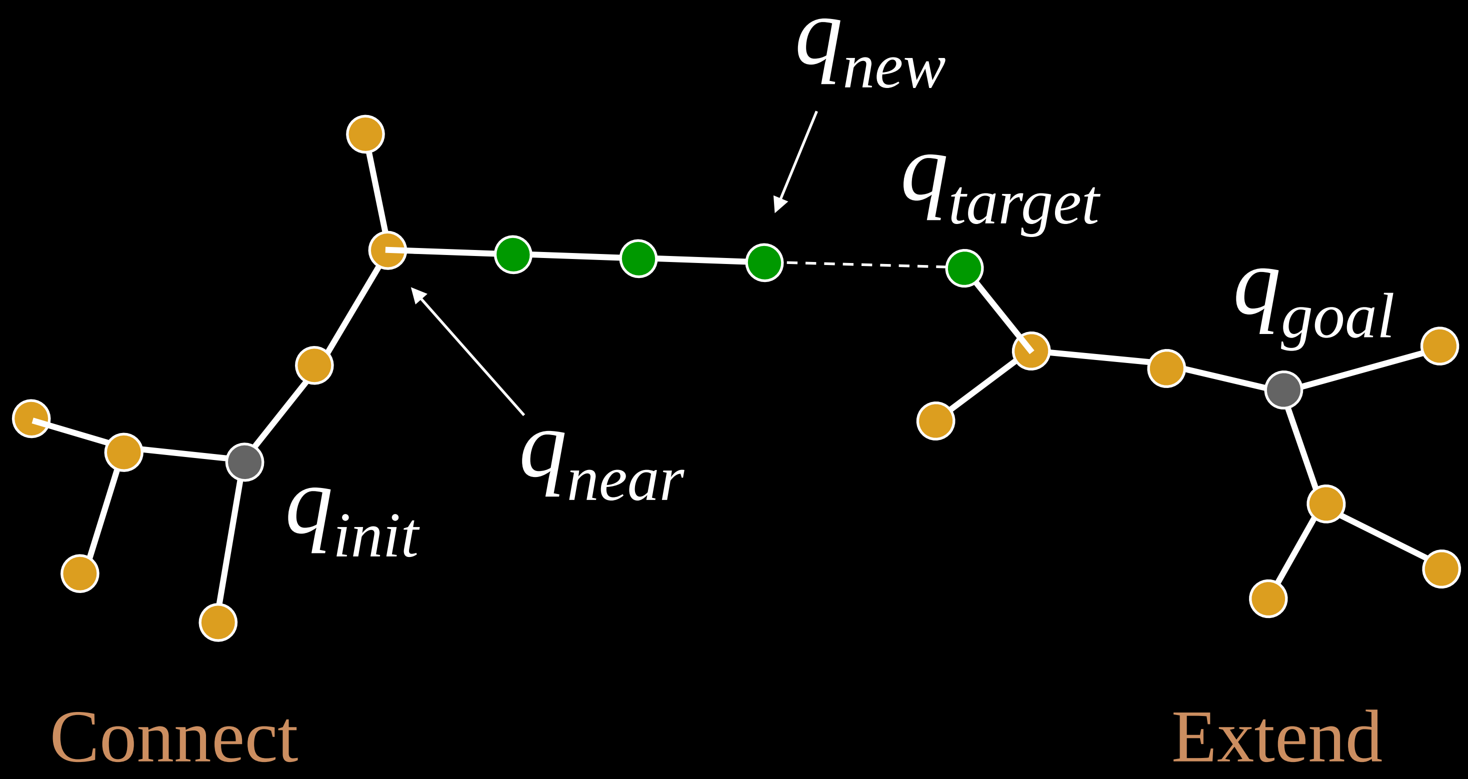

Connect

Extend

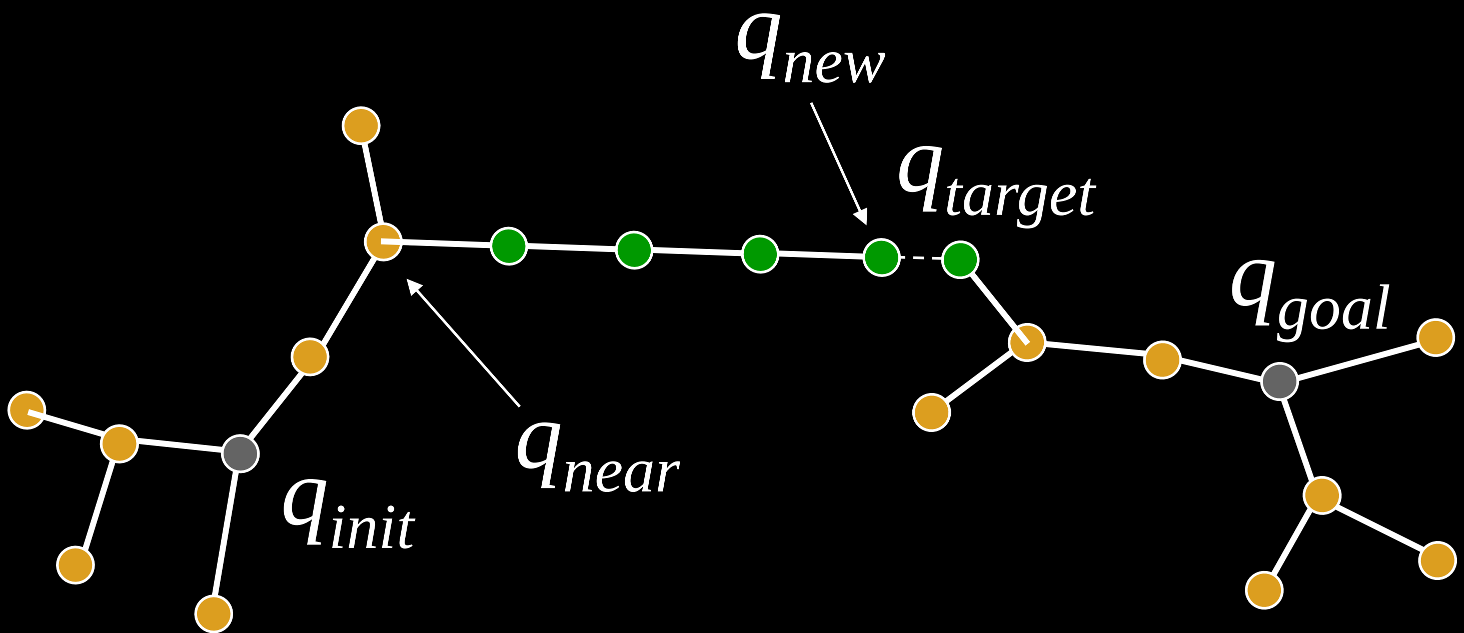
# 5) If successful, keep extending branch



# 5) If successful, keep extending branch



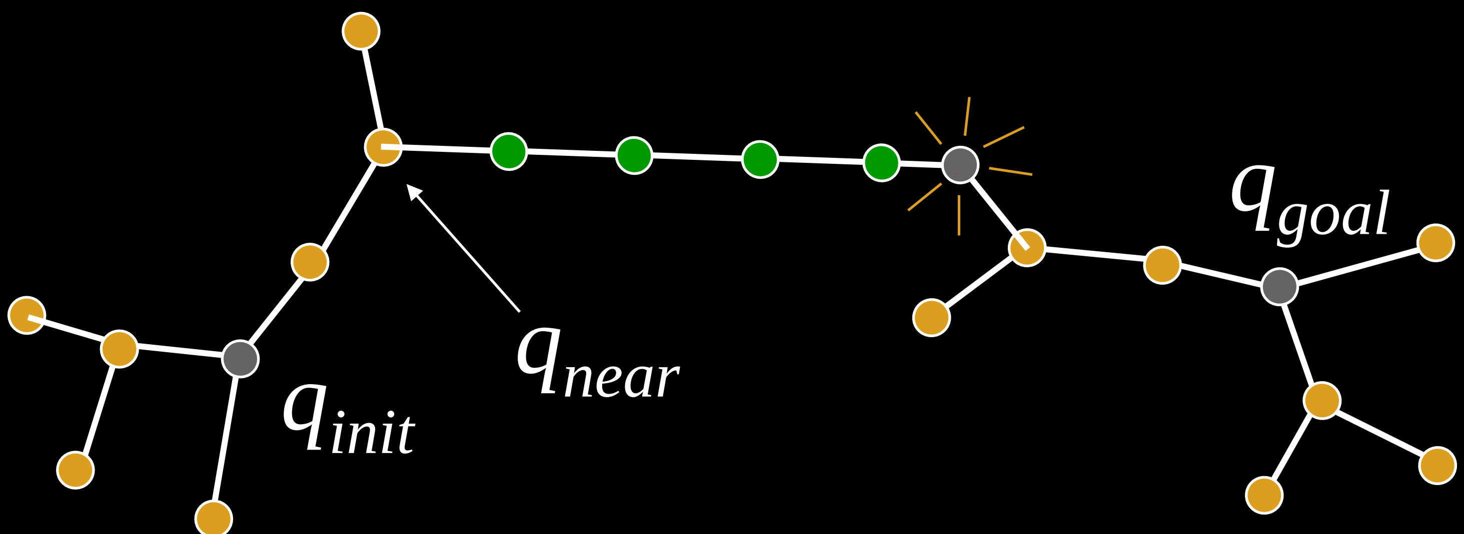
# 5) If successful, keep extending branch




Connect

Extend

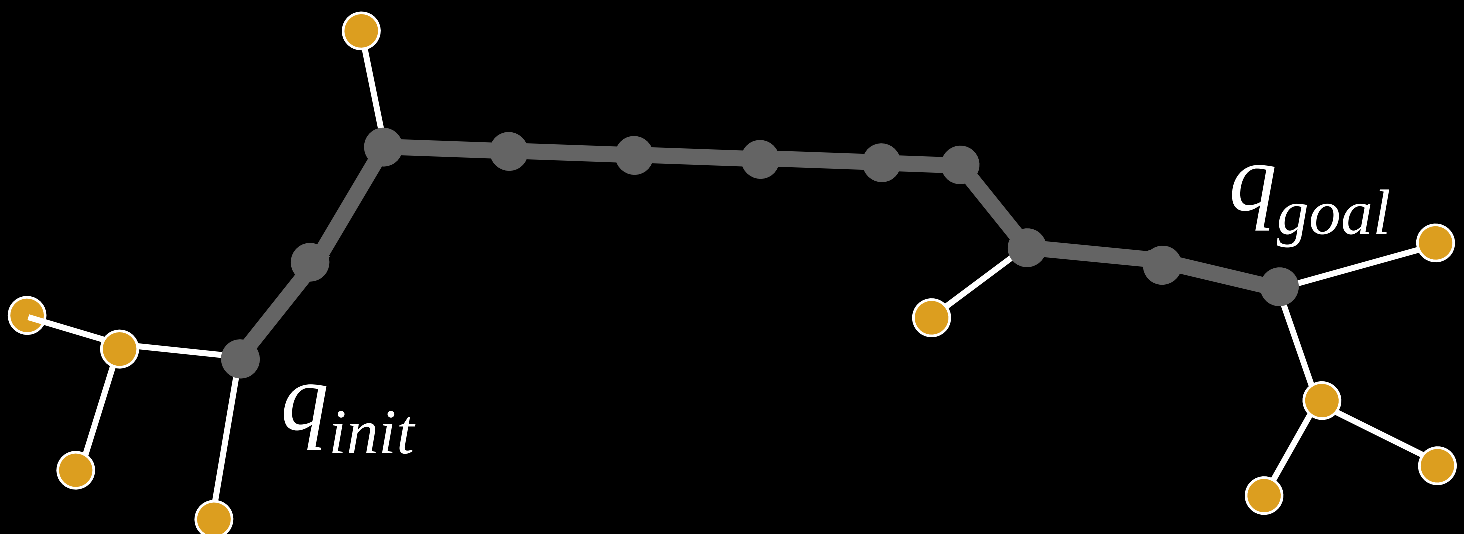
# 6) Path found if branch reaches target




Connect

Extend

# 7) Return path connecting start and goal




Connect

Extend

# Tree Swapping and Balancing

- Usually, swap trees after each iteration
- Some use Tree Balancing:

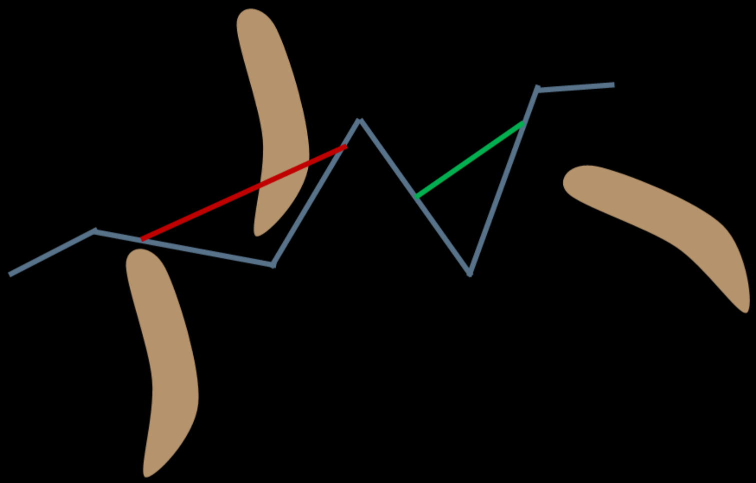
RDT_BALANCED_BIDIRECTIONAL($q_I, q_G$)
$T_a$.init($q_I$); $T_b$.init($q_G$);
**for** $i = 1$ **to** $K$ **do**
  $q_n \leftarrow$ NEAREST($S_a, \alpha(i)$);
  $q_s \leftarrow$ STOPPING-CONFIGURATION($q_n, \alpha(i)$);
  **if** $q_s \neq q_n$ **then**
    $T_a$.add_vertex($q_s$);
    $T_a$.add_edge($q_n, q_s$);
    $q'_n \leftarrow$ NEAREST($S_b, q_s$);
    $q'_s \leftarrow$ STOPPING-CONFIGURATION($q'_n, q_s$);
    **if** $q'_s \neq q'_n$ **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

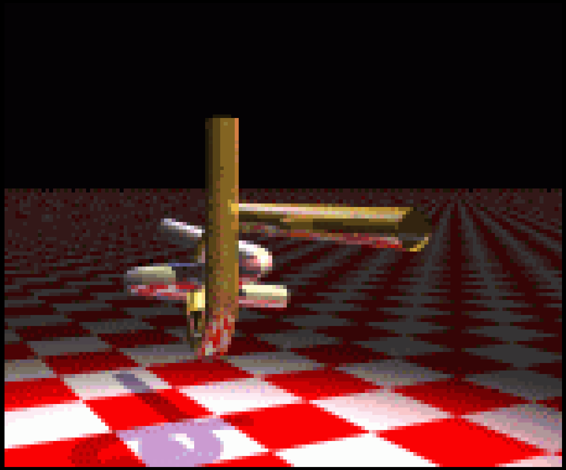
- RRTs produce notoriously bad paths
  - Not surprising since no consideration of path quality

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



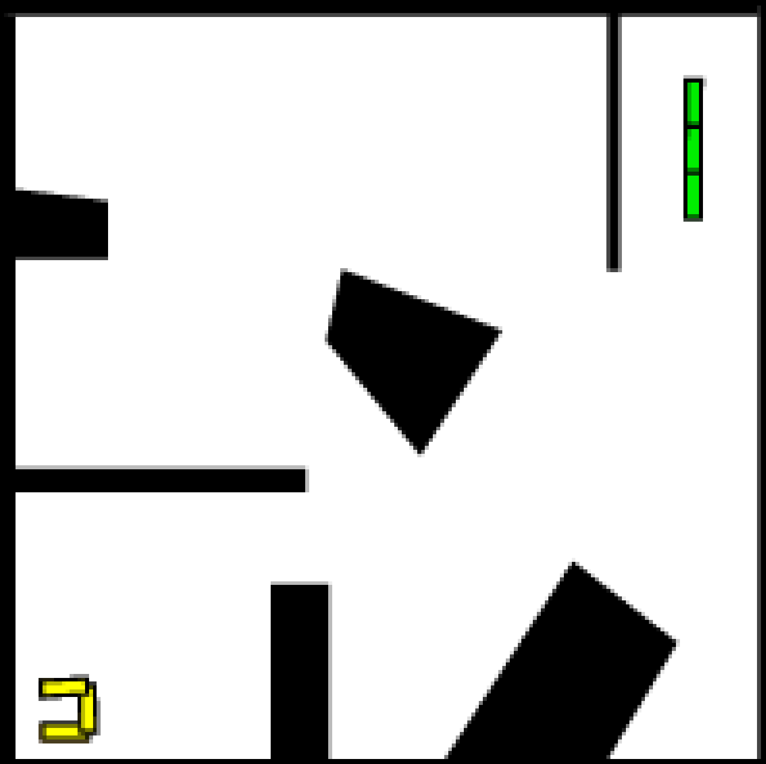
# RRT Examples: The Alpha Puzzle

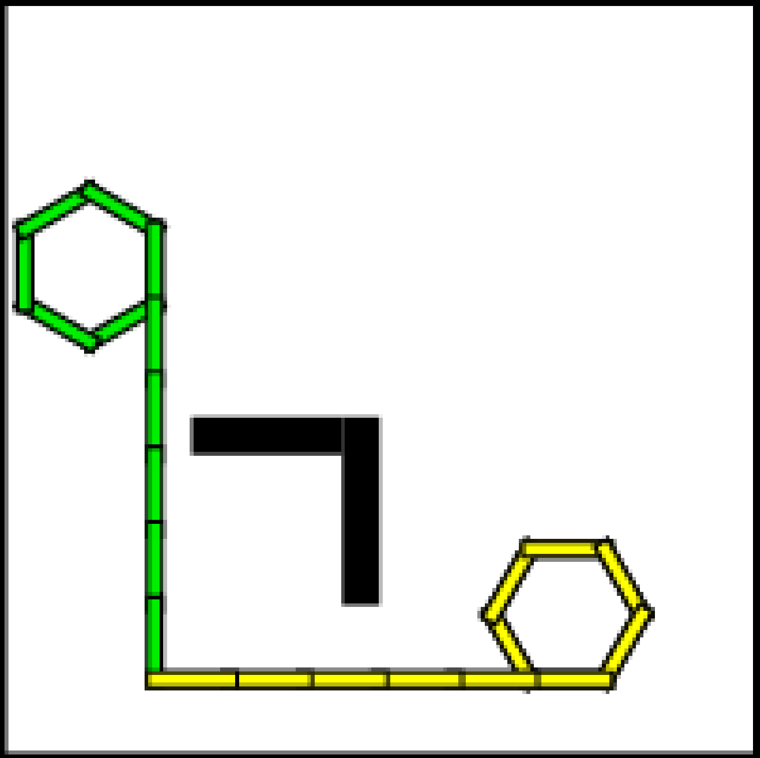
- VERY hard 6DOF motion planning problem (long, winding narrow passage)



- "*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 Examples: Articulated Objects





# RRT Analysis

The limiting 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

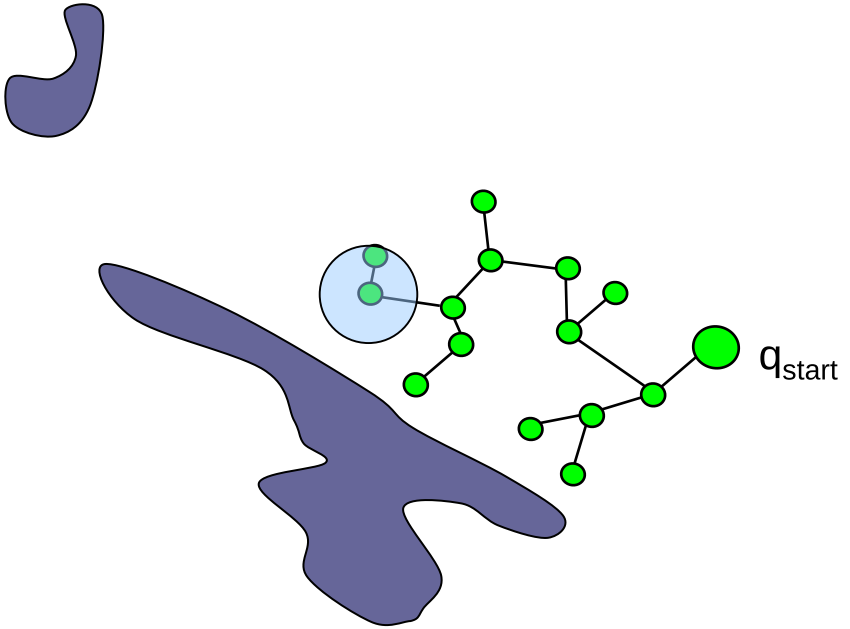
- If using uniform distribution, tree nodes converge to the free space

# 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.

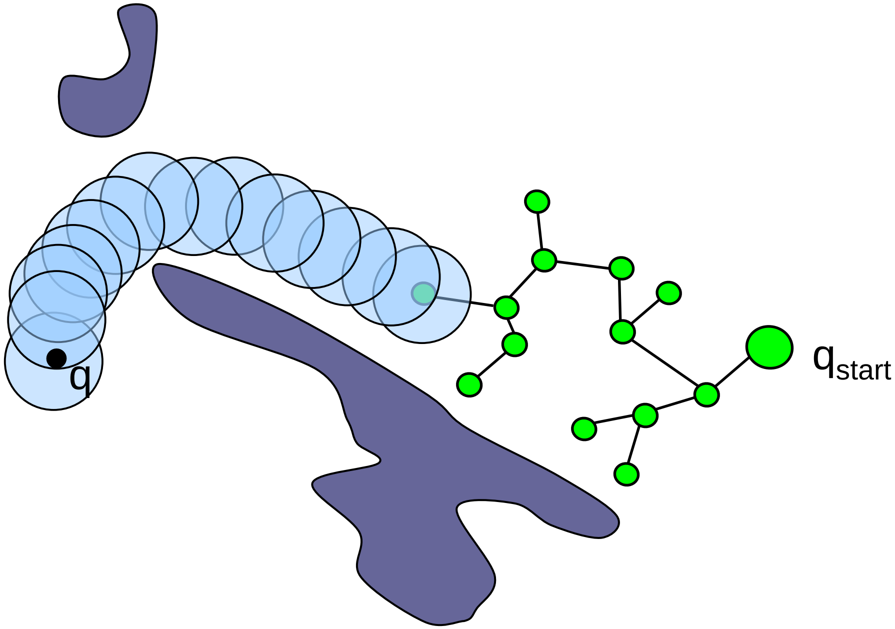
- Will RRT explore the whole space?

# RRT P.C. Proof



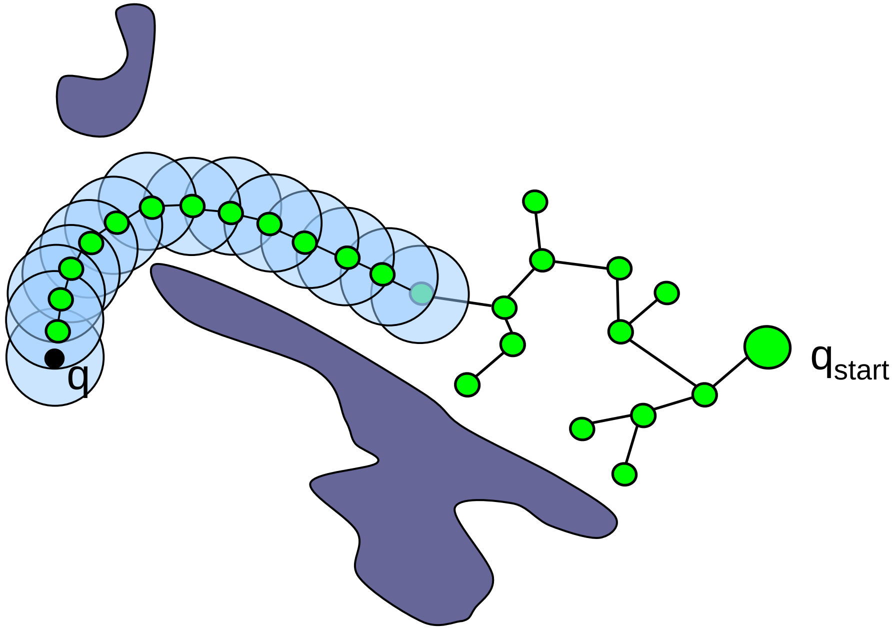

Kuffner and LaValle, ICRA, 2000

# RRT P.C. Proof




Kuffner and LaValle, ICRA, 2000

# RRT P.C. Proof




Kuffner and LaValle, ICRA, 2000

# Probabilistic Completeness

- As the RRT reaches all of $C_{free}$ to $q_{start}$, 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
  - Works even with the wrong distance metric

# Sampling-Based Planning

- The bad:
  - No quality guarantees on paths*
    - 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

*Asymptotically-optimal sampling-based planners can make some guarantees

## Homework

- Read LaValle Ch. 14-14.5
- Read “How to Read a Paper” guide (link on website)
- Make sure you’ve started HW2!