Robotic Motion Planning: RRT's

Robotics Institute 16-735

http://www.cs.cmu.edu/~motion

Howie Choset

http://www.cs.cmu.edu/~choset

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

Next HW Assignment

- Implement a PRM planner for a multi-link (at least four) robot arm. The arm can be a simple planar arm (which will simplify the graphics), or a 3D arm. The arm can be composed of line segments (which will make collision checking easier) rather than finite volume links. All you need to do is write code to detect the intersection between line segments and polygons. If you want, you can use collision checking software that is available on the web.

- How was the previous?

- This is the last one

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

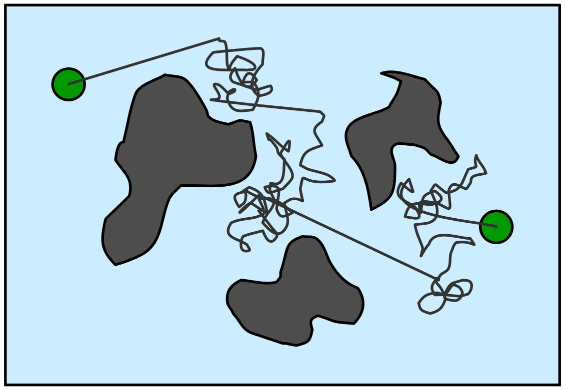
Performance & Implementation Issues
- Metrics and Metric sensitivity
- Nearest neighbors
- Collision Checking
- Choosing appropriate step sizes

High-Dimensional Planning as of 1999

Single-Query:

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

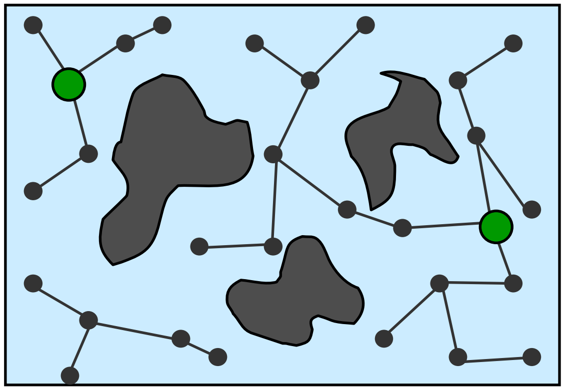
EXAMPLE: Potential-Field



Multiple-Query:

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

EXAMPLE: PRM

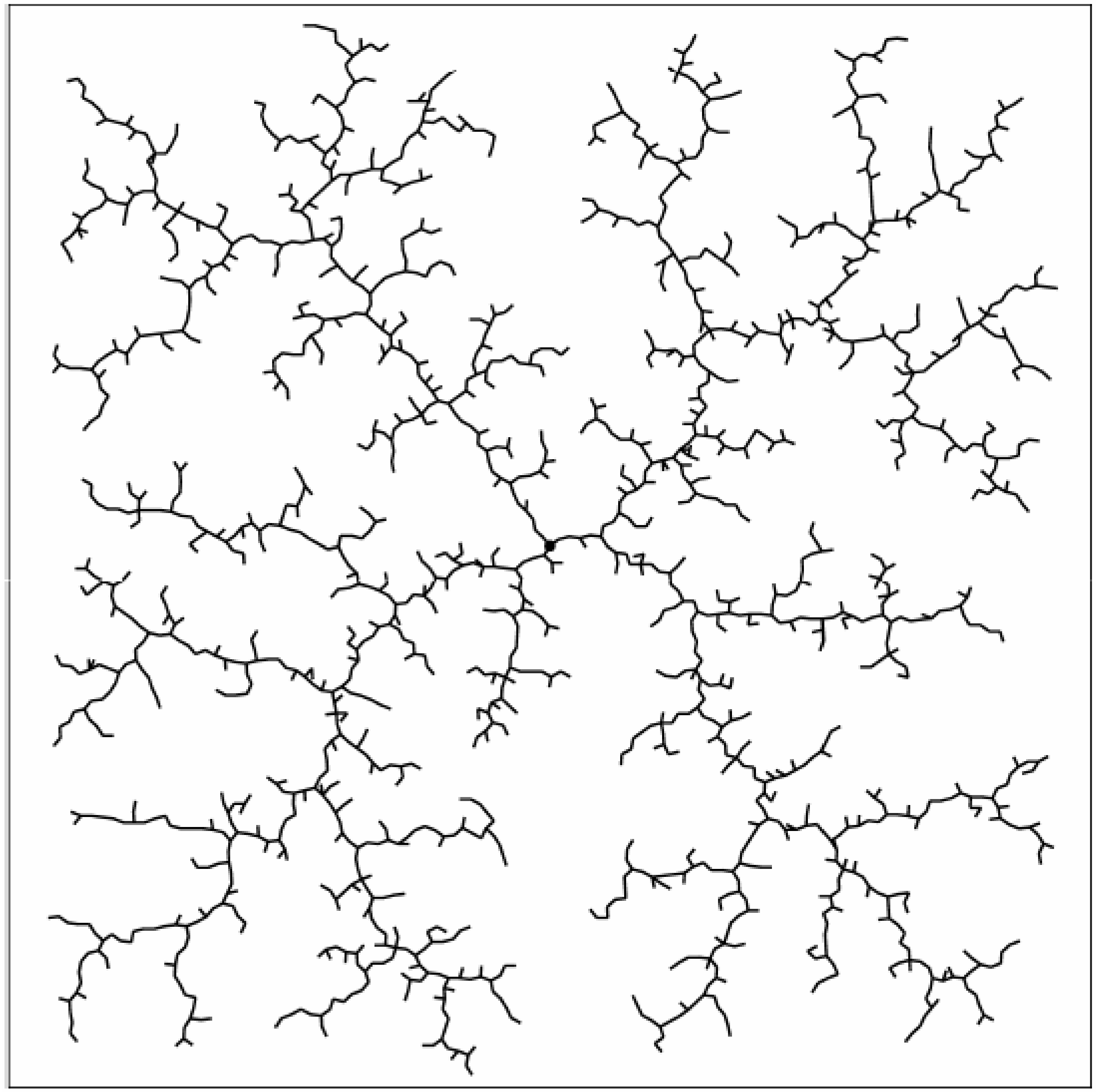


<u>TENSION</u>

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

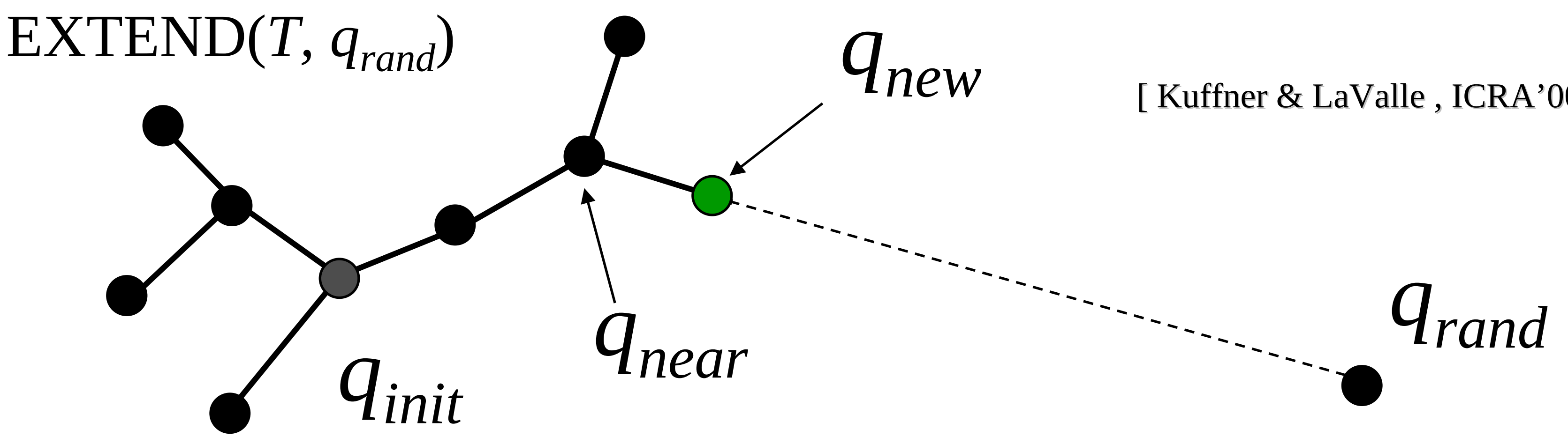
Spreads out like uniformity but need lots of sample to cover space

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})
}



[Kuffner & LaValle , ICRA'00]

Path Planning with RRTs
(Some Details)

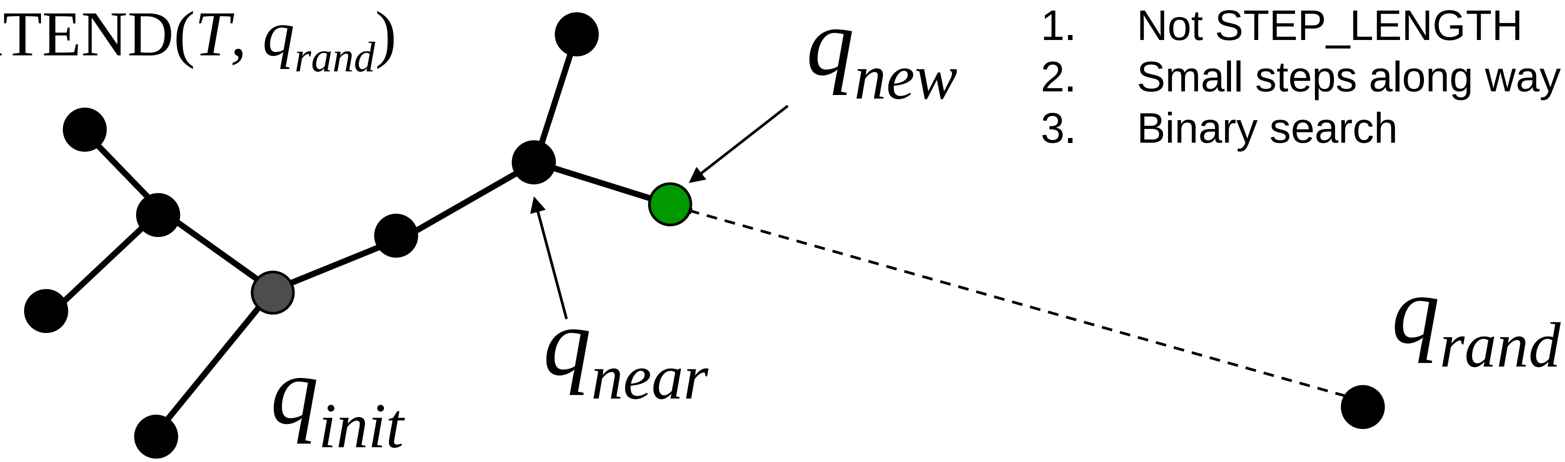
BUILD_RRT (q_{init}) {
  $T.init(q_{init})$;
  for k = 1 to K do
    q_{rand} = RANDOM_CONFIG();
    EXTEND(T, q_{rand})
}

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

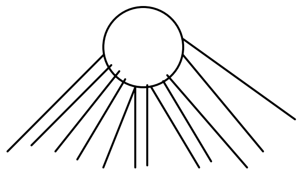
STEP_SIZE
1. Not STEP_LENGTH
2. Small steps along way
3. Binary search

EXTEND(T, q_{rand})

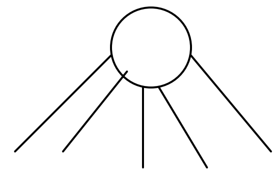


RRT vs. Exhaustive Search

- Discrete

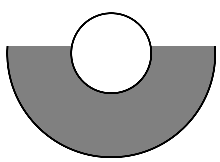


A* may try all edges

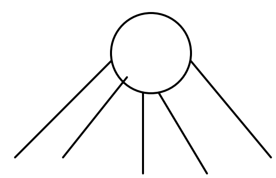


Probabilistically subsample all edges

- Continuous



Continuum of choices



Probabilistically subsample all edges

Naïve Random Tree

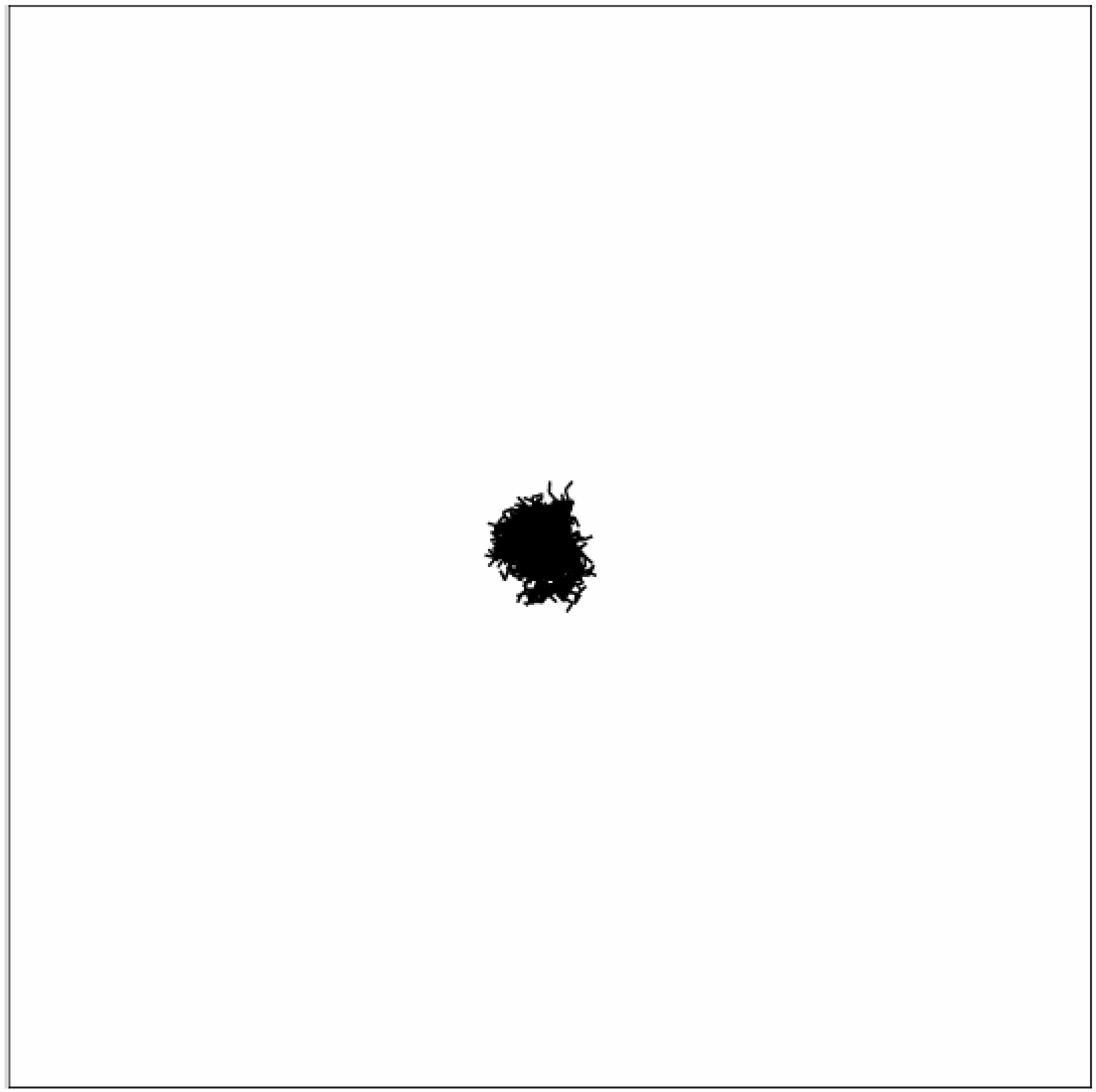
Start with middle

Sample near this node

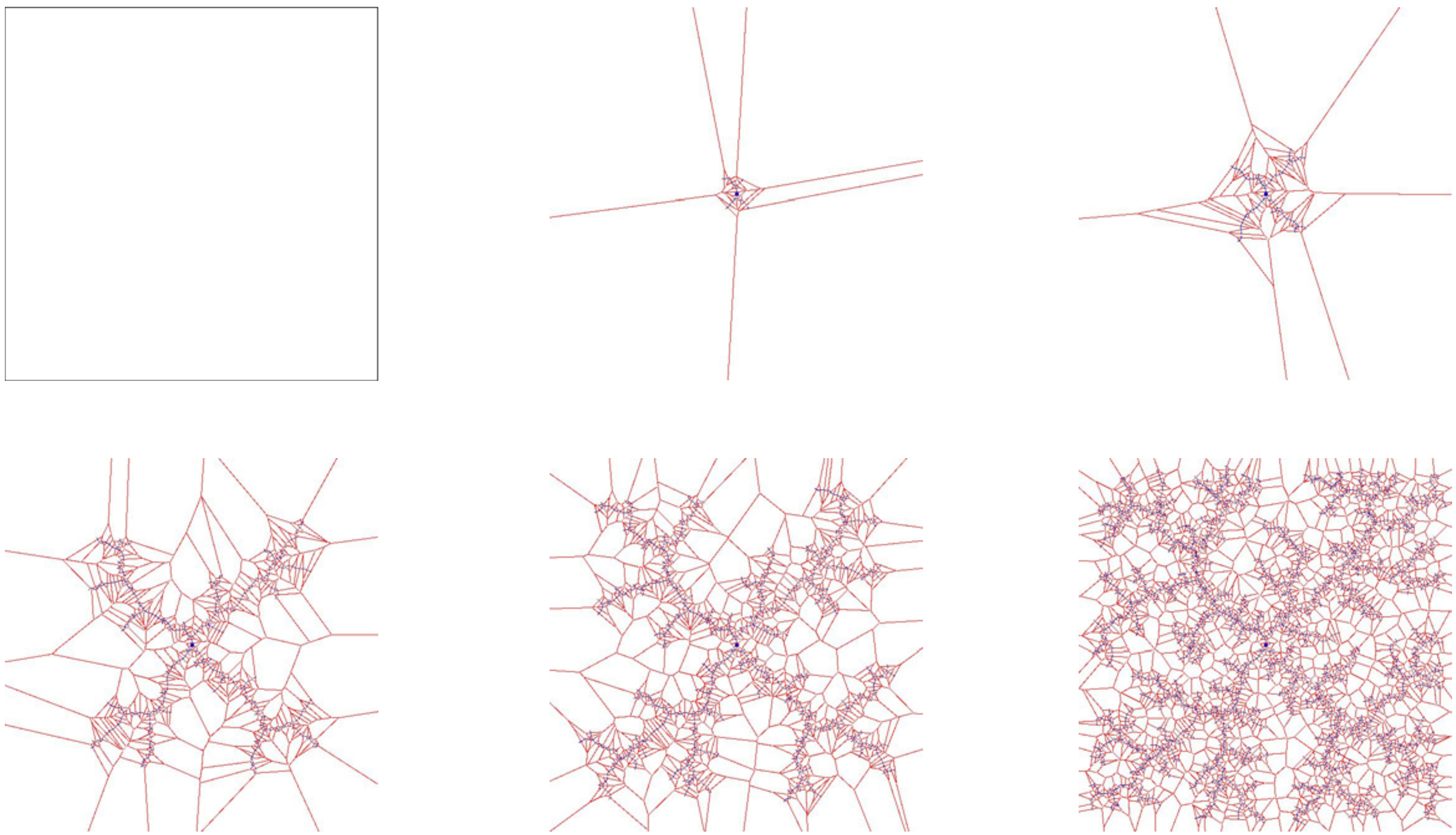
Then pick a node at random in tree

Sample near it

End up Staying in middle



RRTs and
Bias toward large Voronoi regions

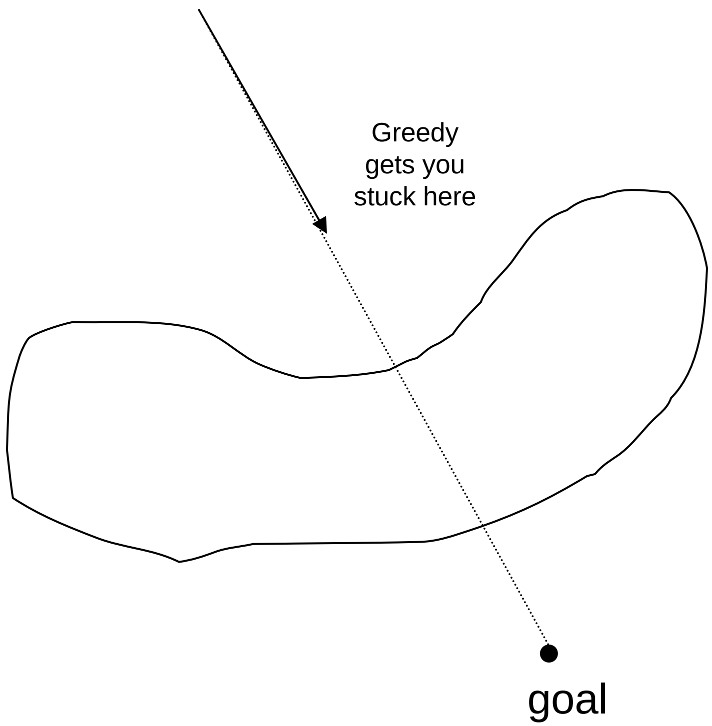


http://msl.cs.uiuc.edu/rrt/gallery.html

Biases

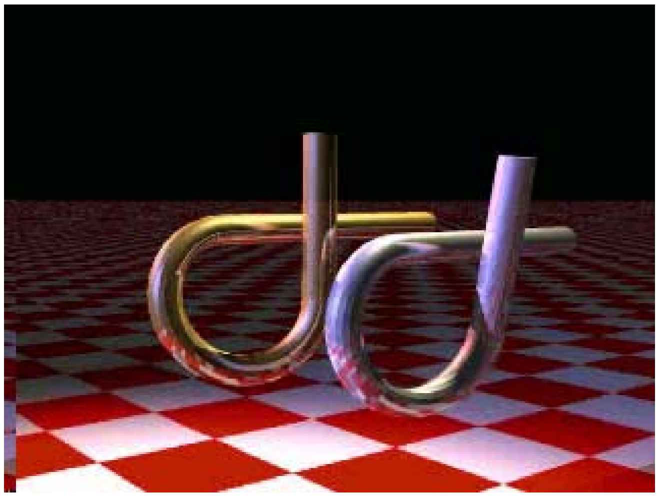
- Bias toward larger spaces
- Bias toward goal
  - When generating a random sample, with some probability pick the goal instead of a random node when expanding
  - This introduces another parameter
  - James' experience is that 5-10% is the right choice
  - If you do this 100%, then this is a RPP

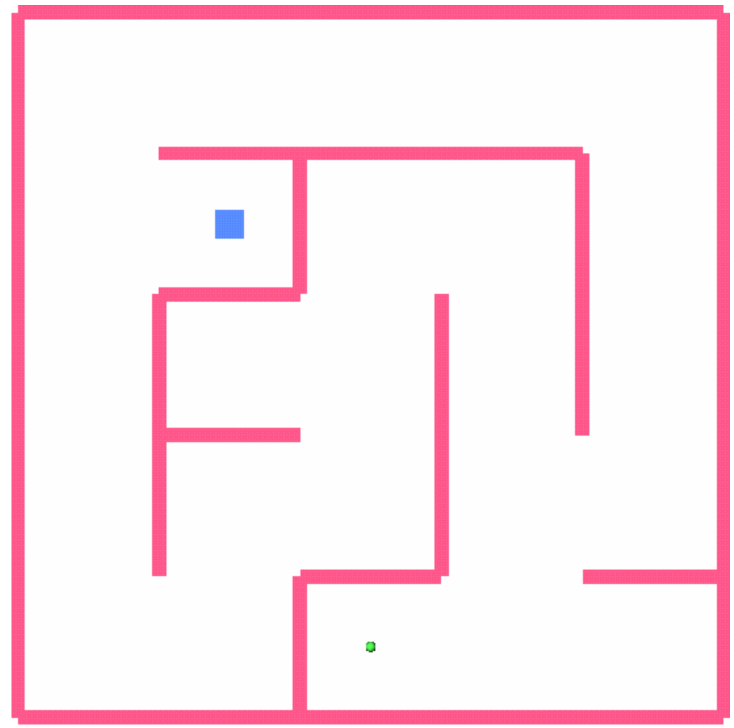
RRT vs. RPP



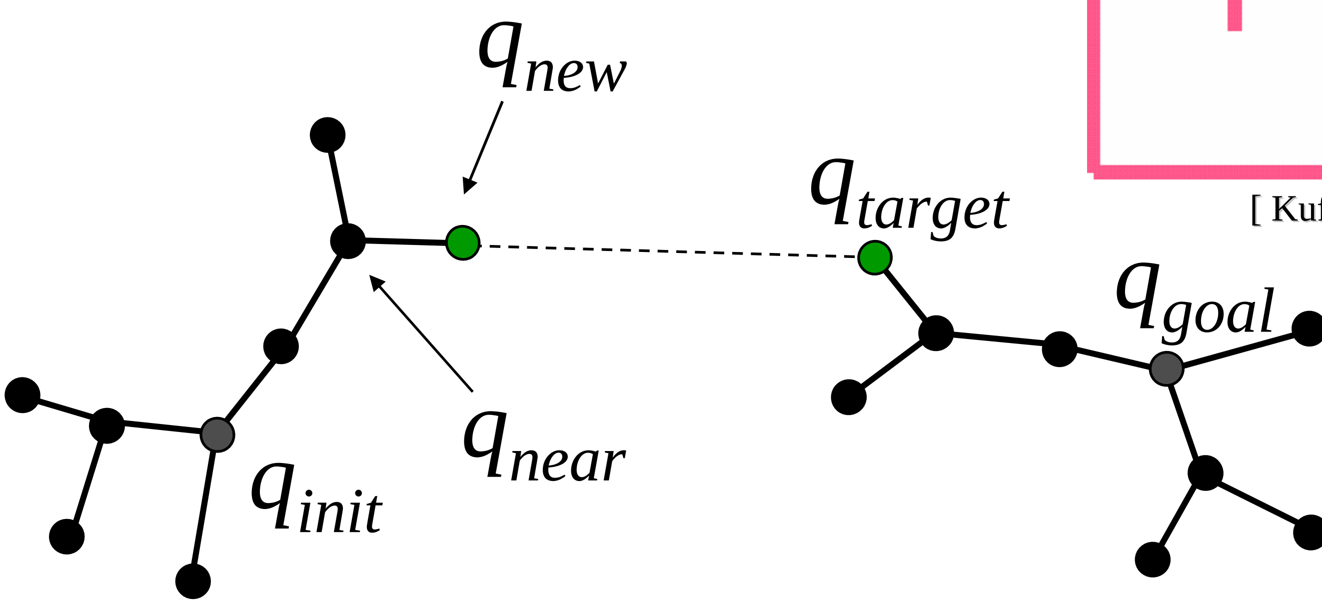
RRT's will pull away and better approximate cost-to-go

Grow two RRTs towards each other

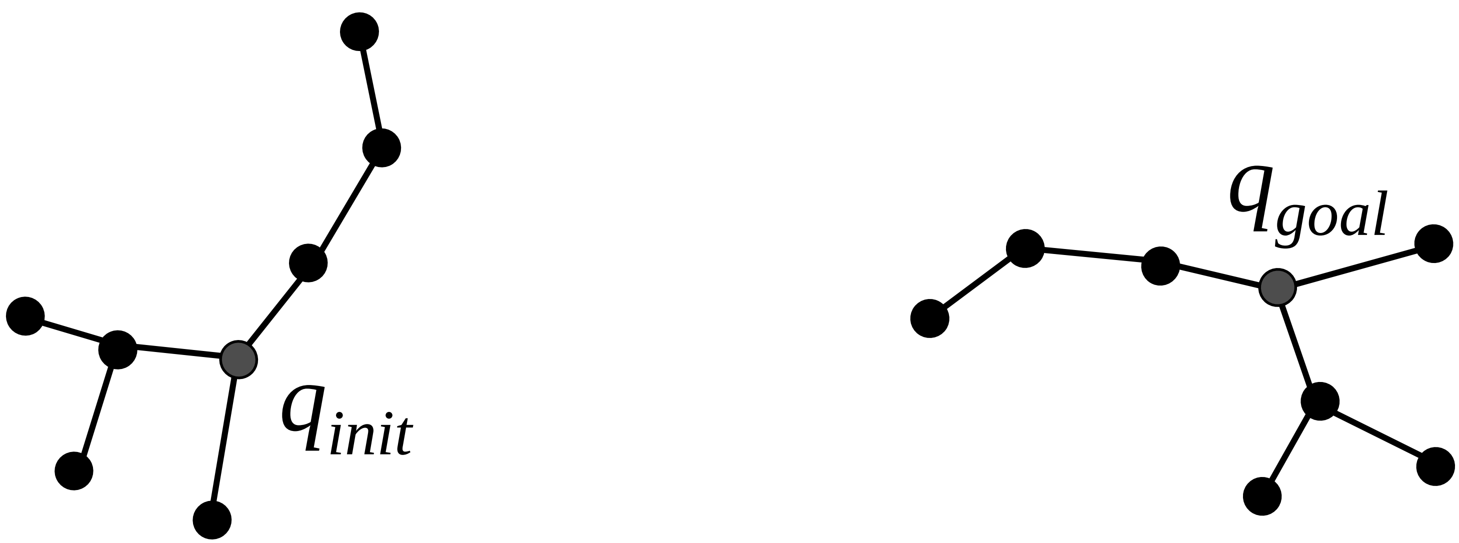




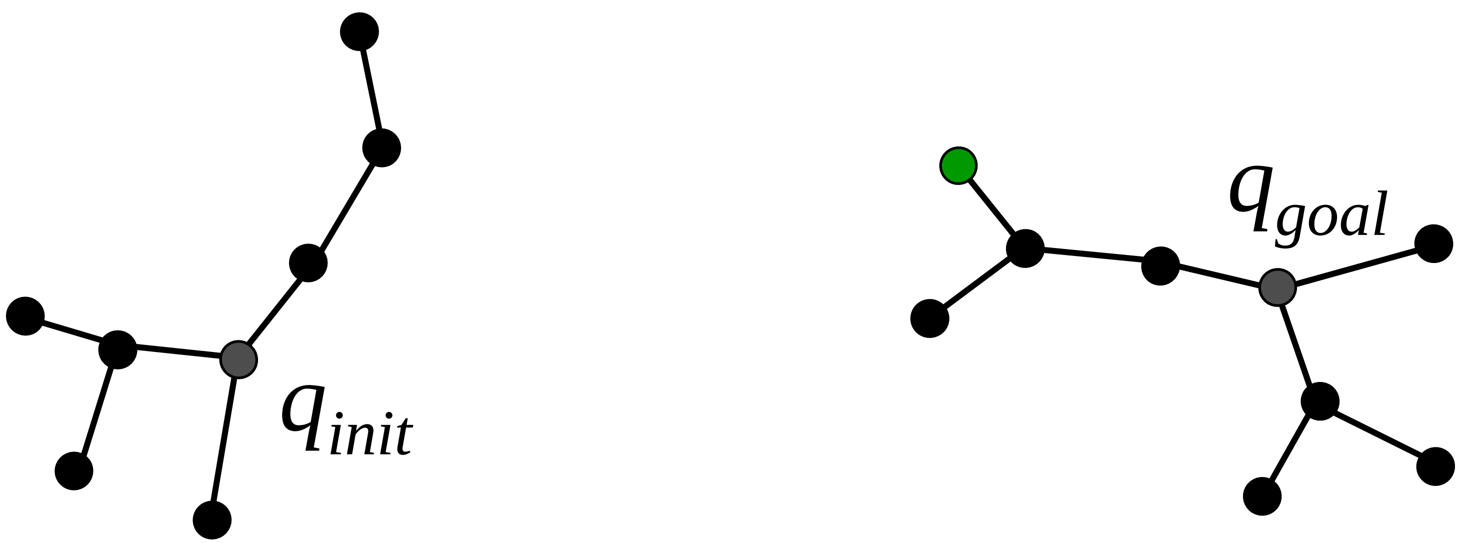
[Kuffner, LaValle ICRA '00]



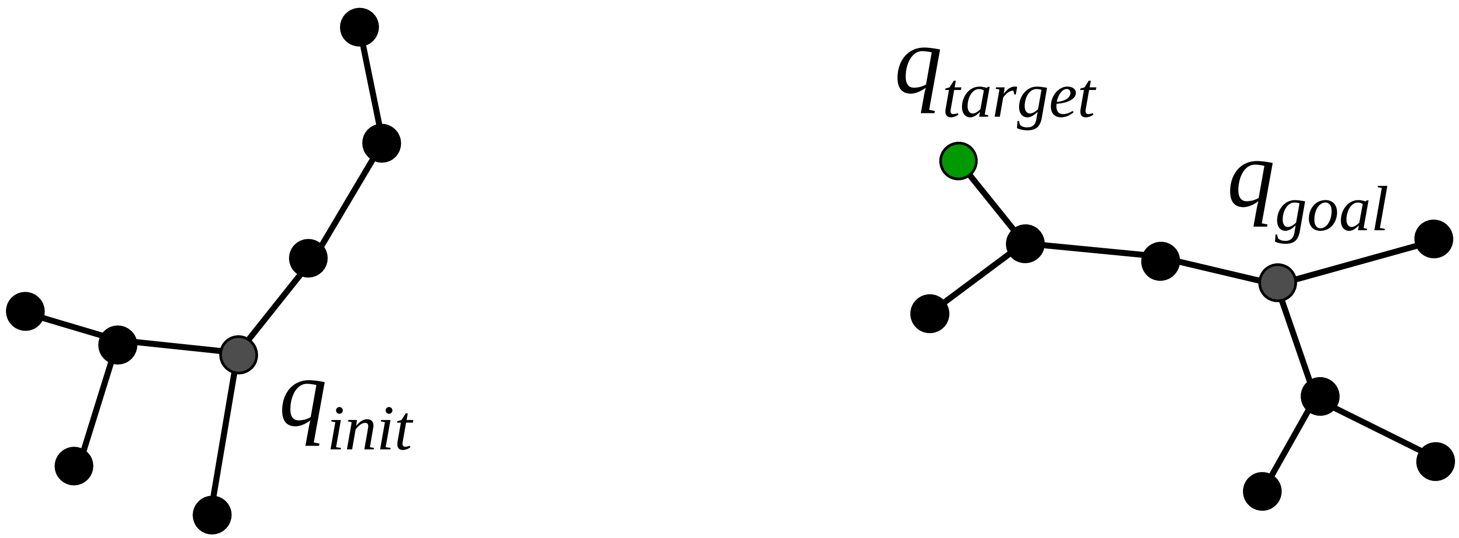
A single RRT-Connect iteration...



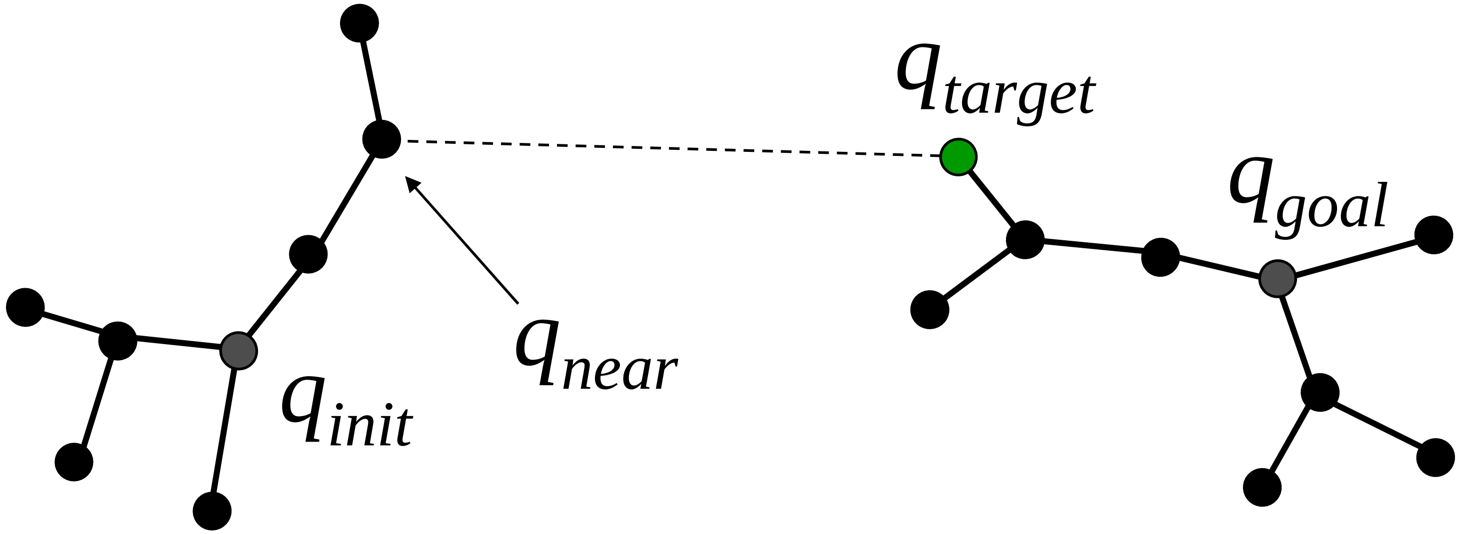
1) One tree grown using random target



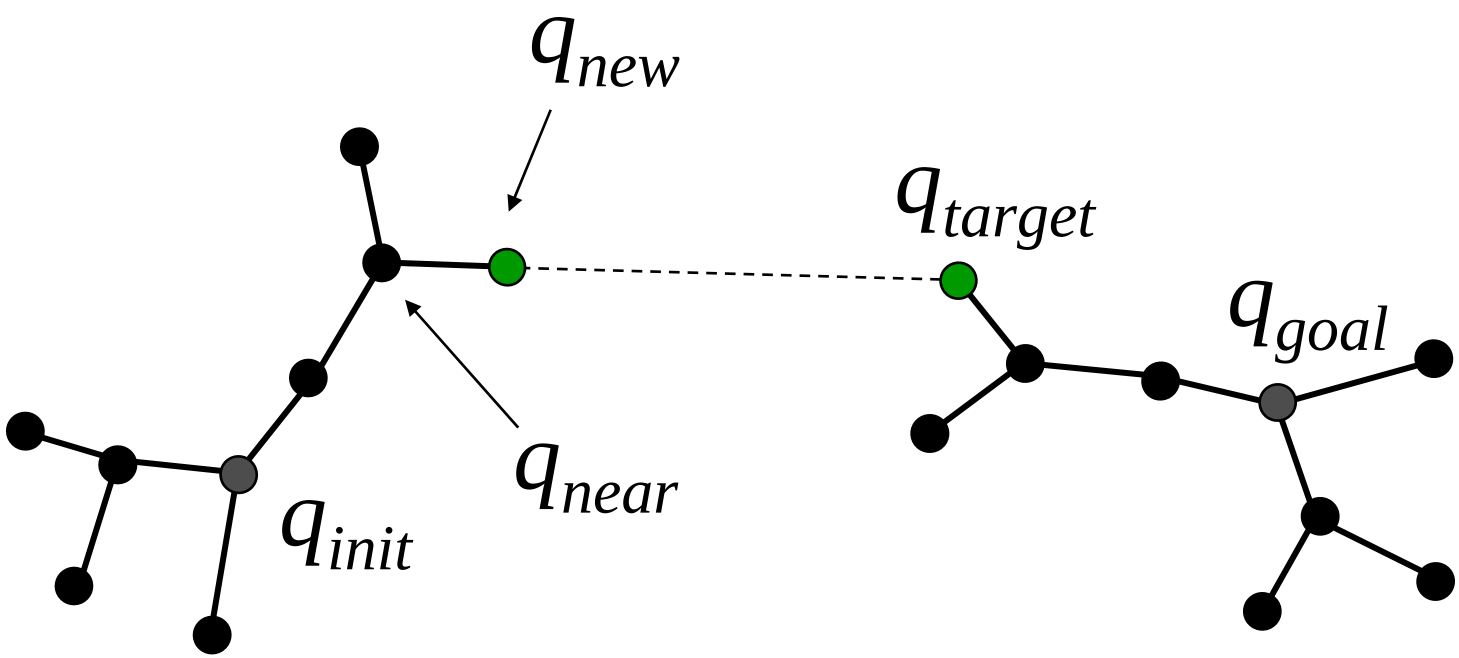
2) New node becomes target for other tree



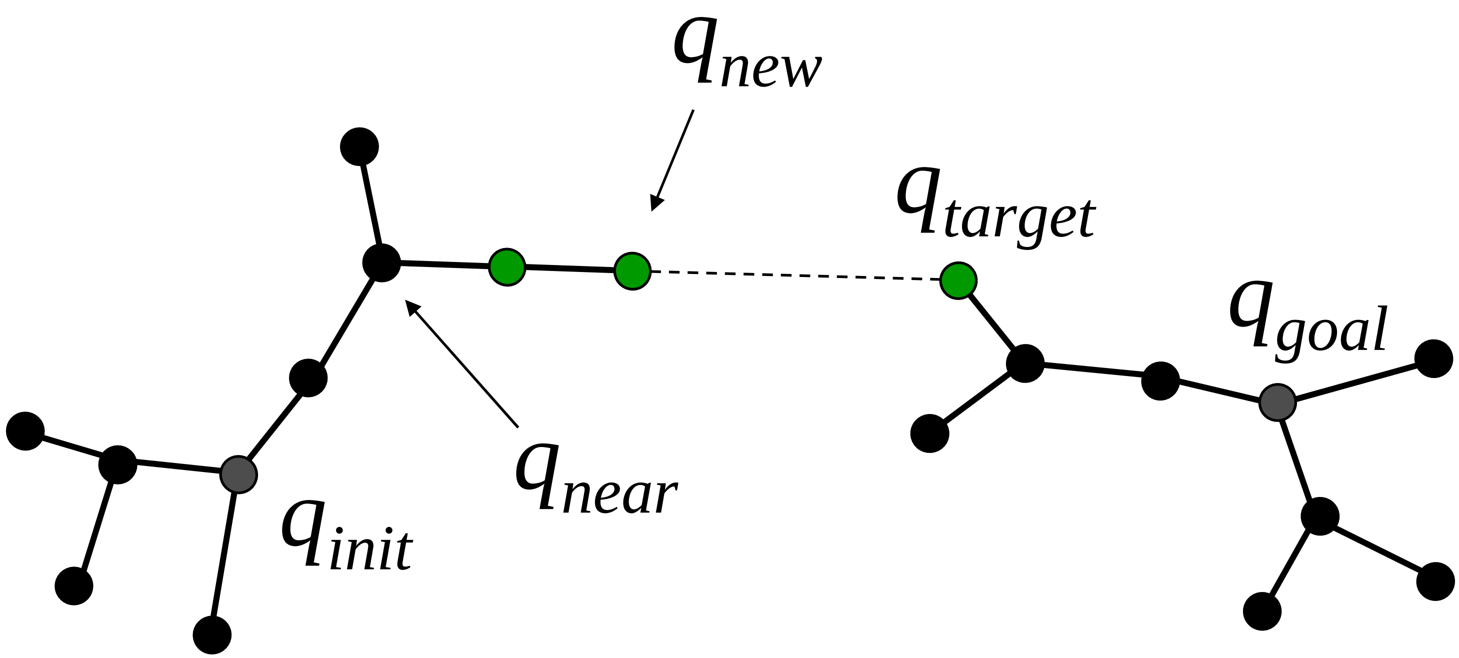
3) Calculate node “nearest” to target



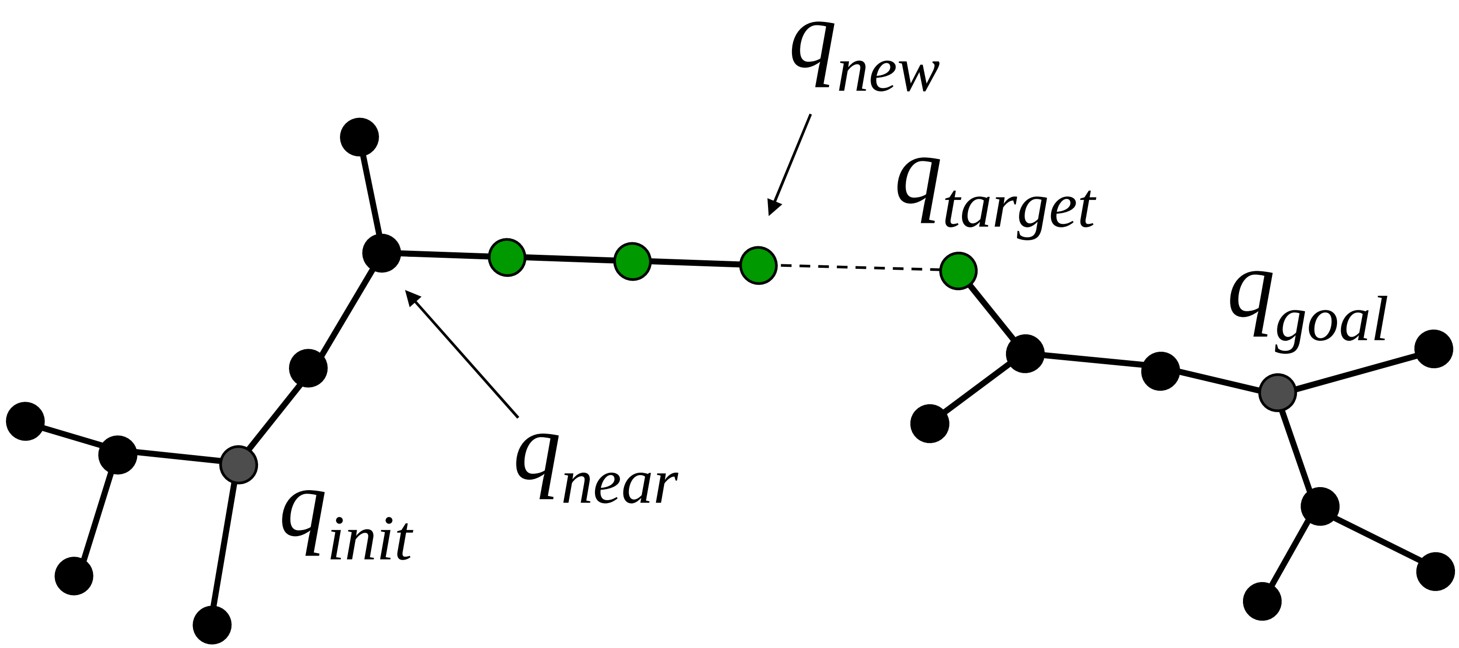
4) Try to add new collision-free branch



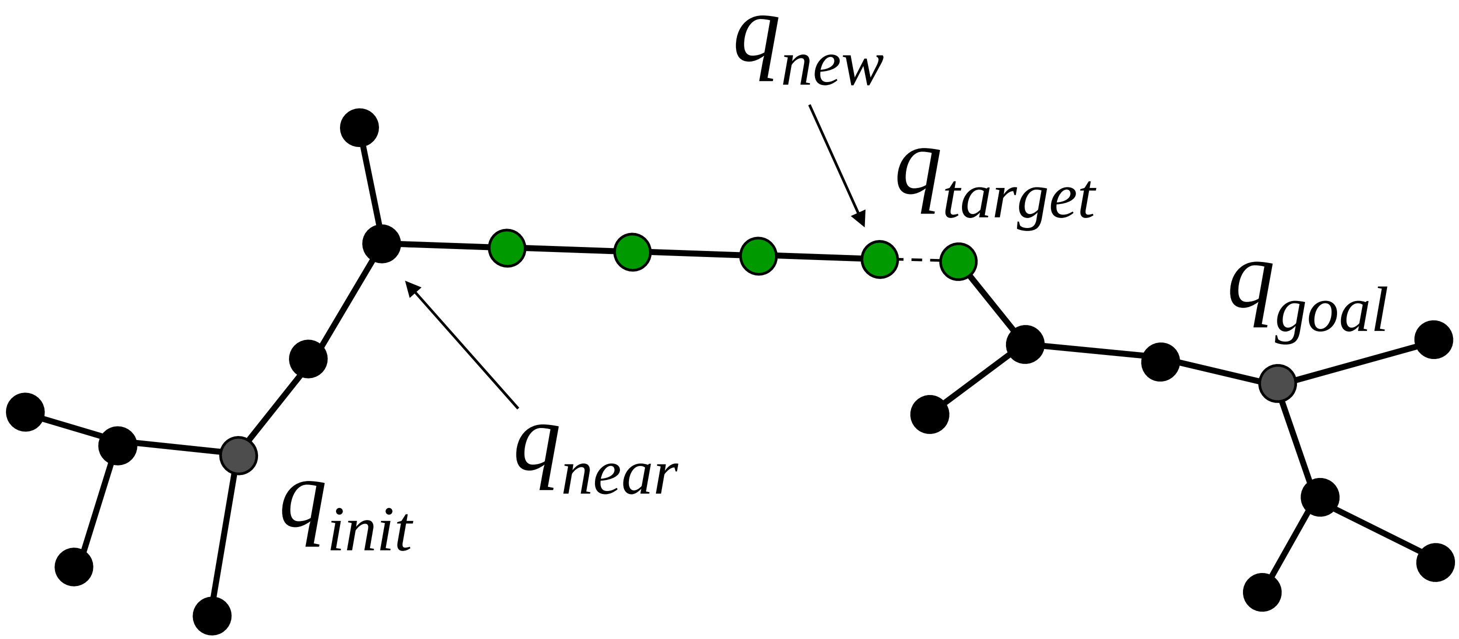
5) If successful, keep extending branch



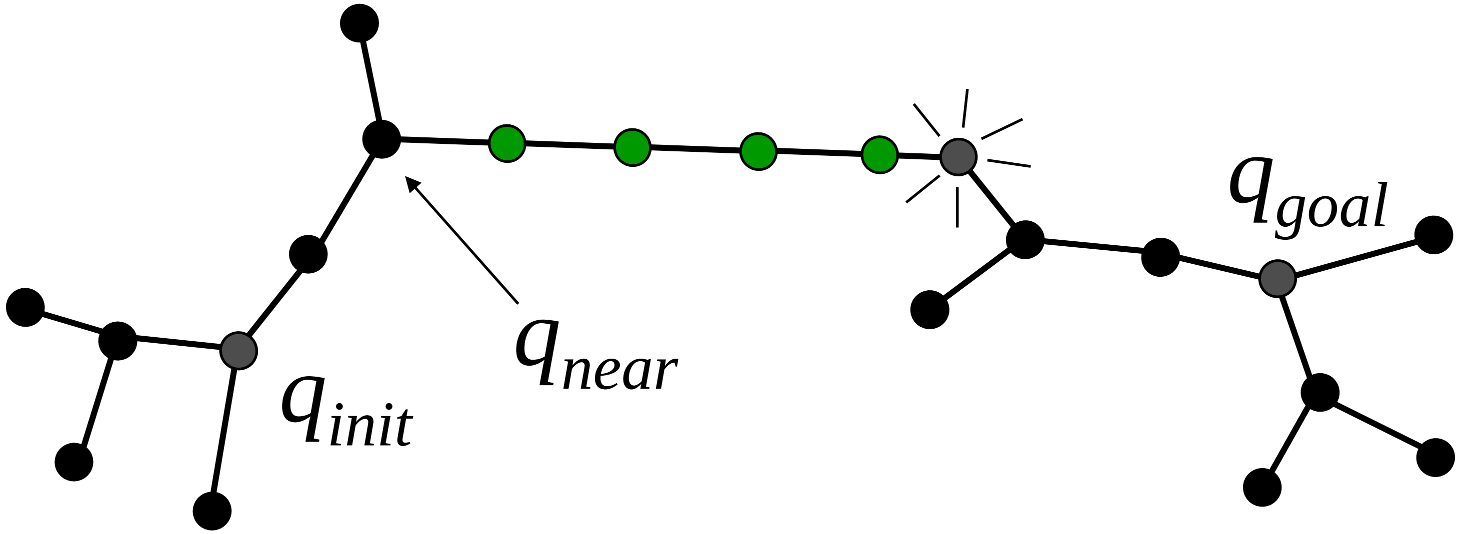
5) If successful, keep extending branch



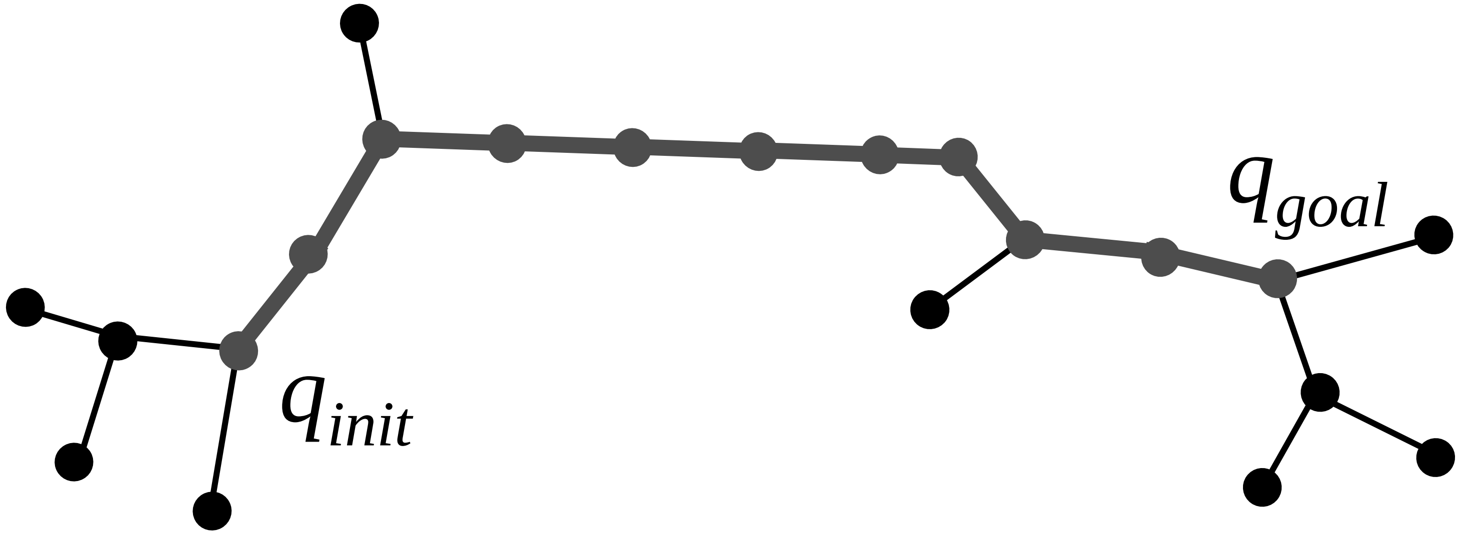
5) If successful, keep extending branch



6) Path found if branch reaches target



7) Return path connecting start and goal

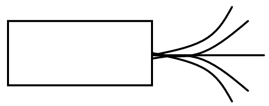


Basic RRT-Connect

```
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;
}
```

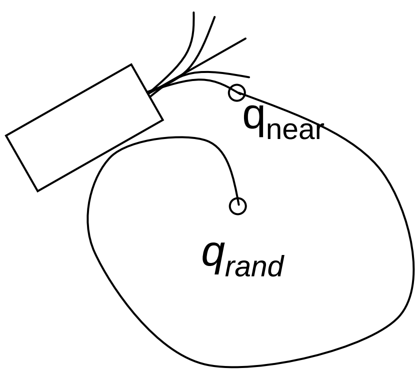
Instead of switching, use T_a as smaller tree. This helped James a lot

q_{near}

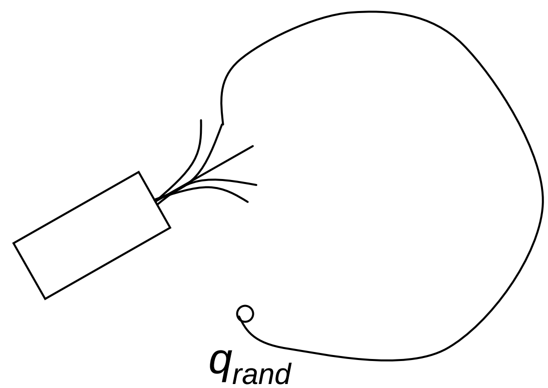


$q' = f(q,u)$ --- use action u from q to arrive at q'

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



Is this the best?

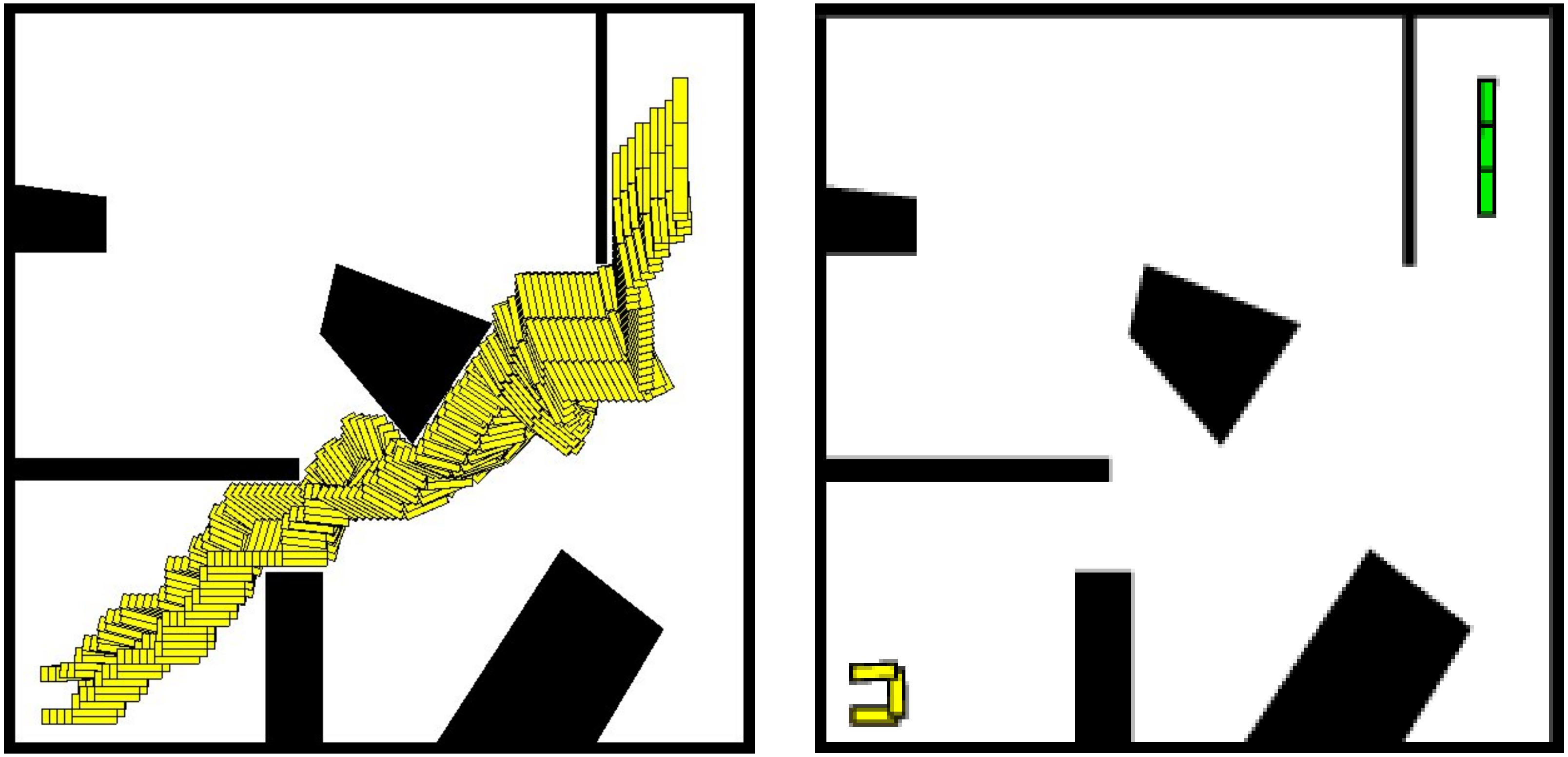


Mixing position and velocity, actually mixing position, rotation and velocity is hard

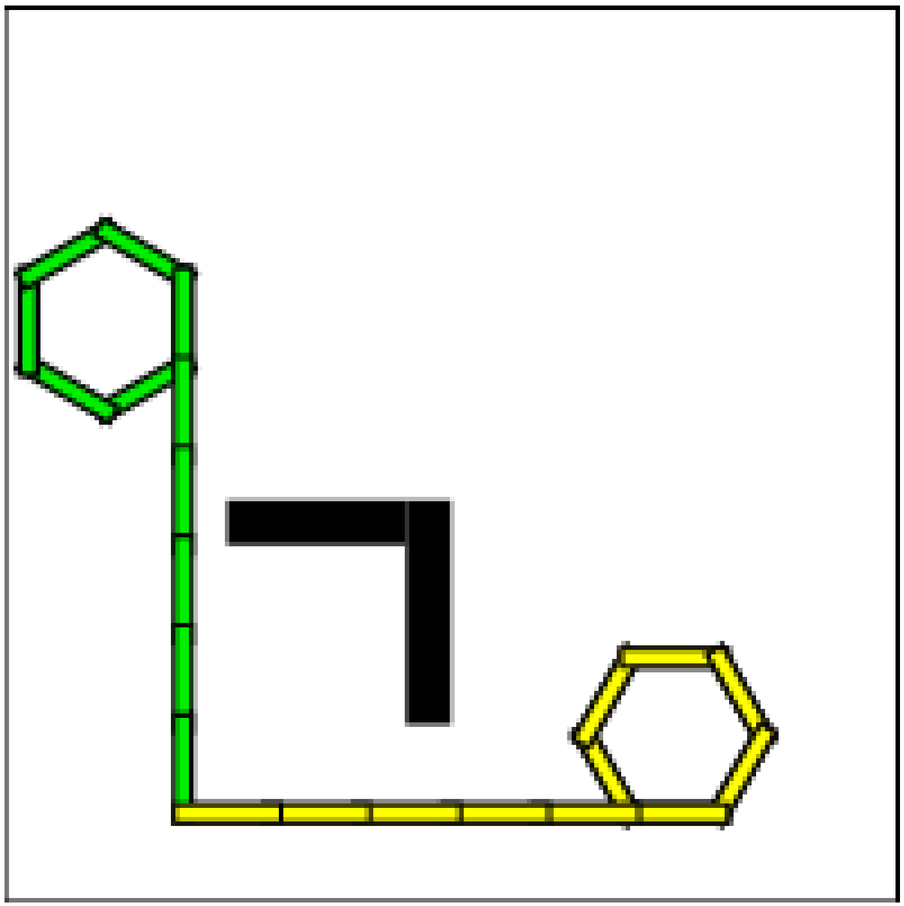
So, what do they do?

- 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

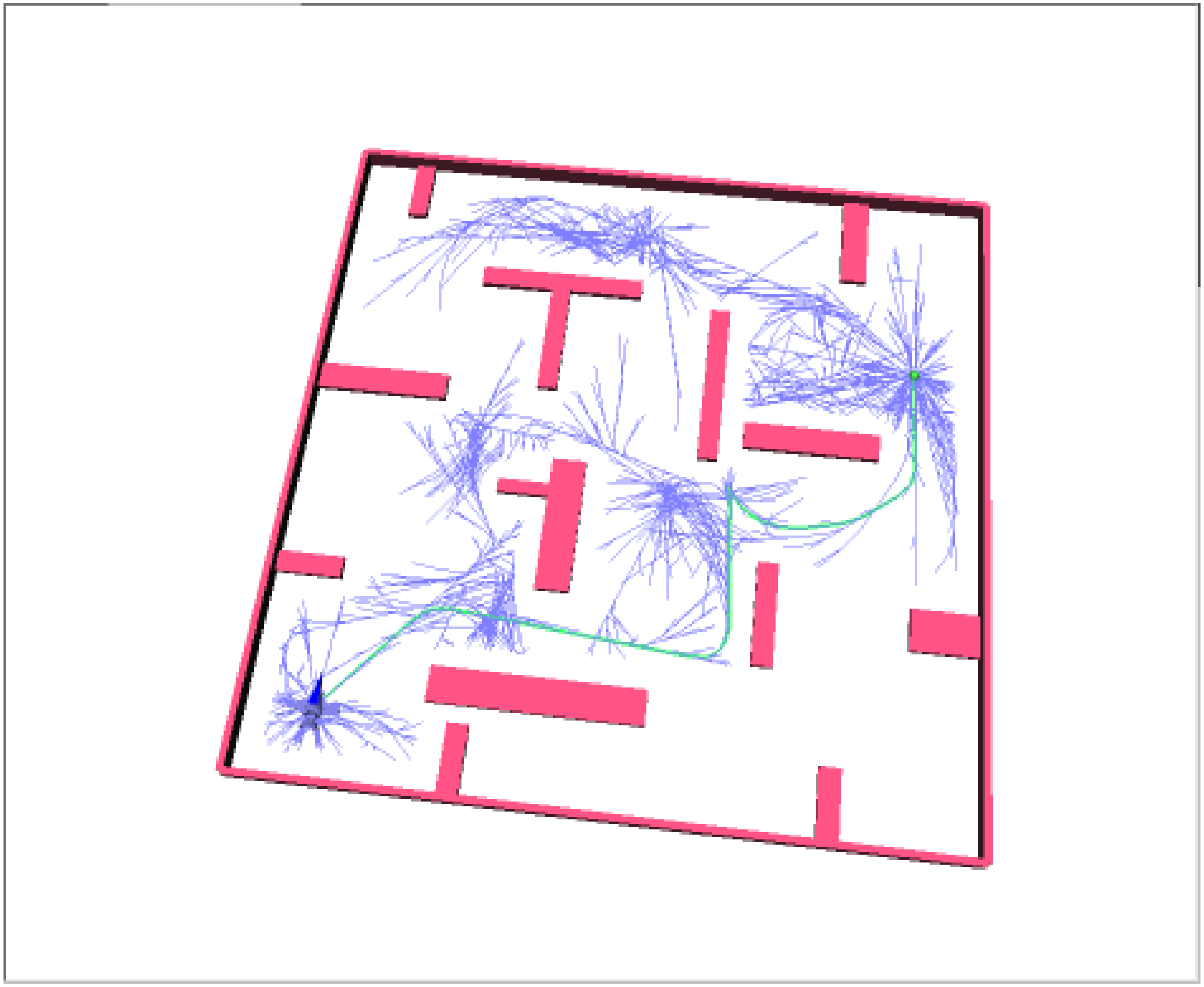
Articulated Robot



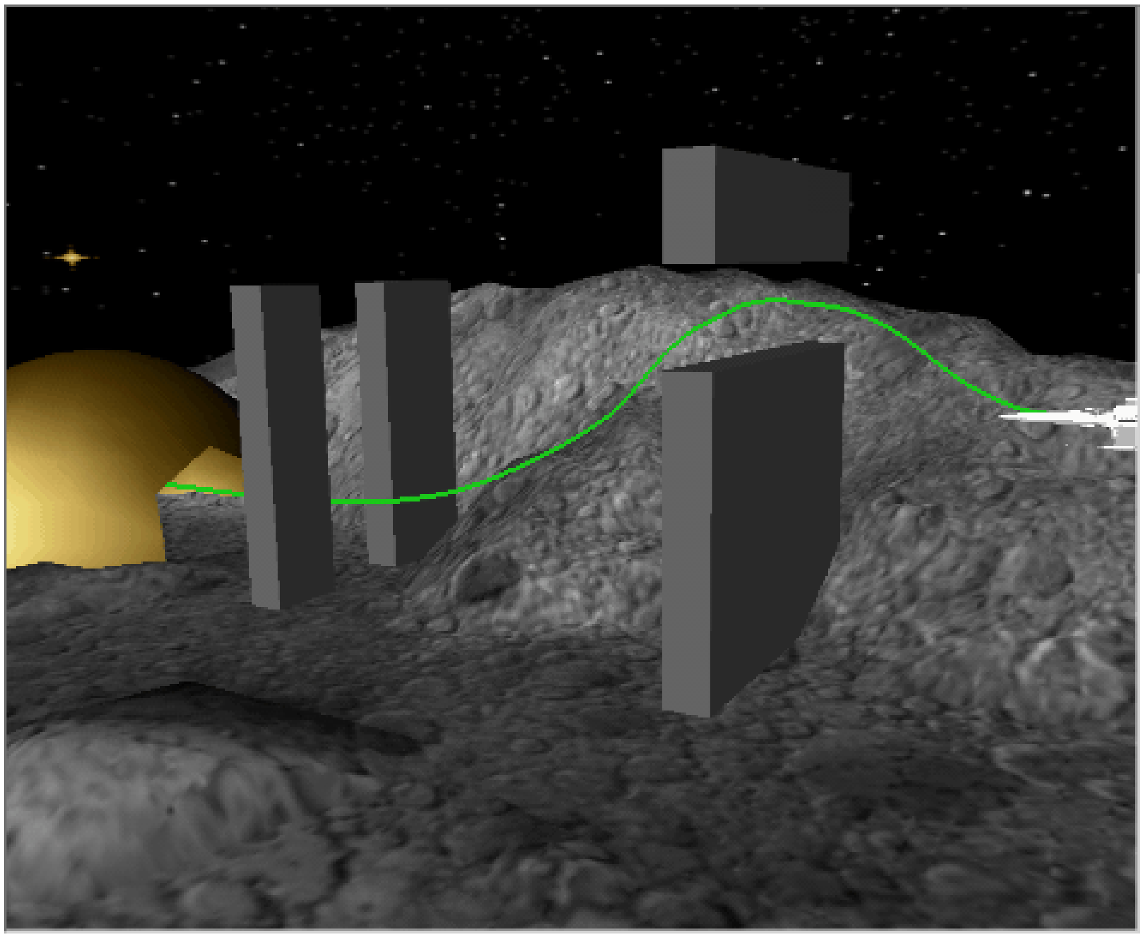
Highly Articulated Robot



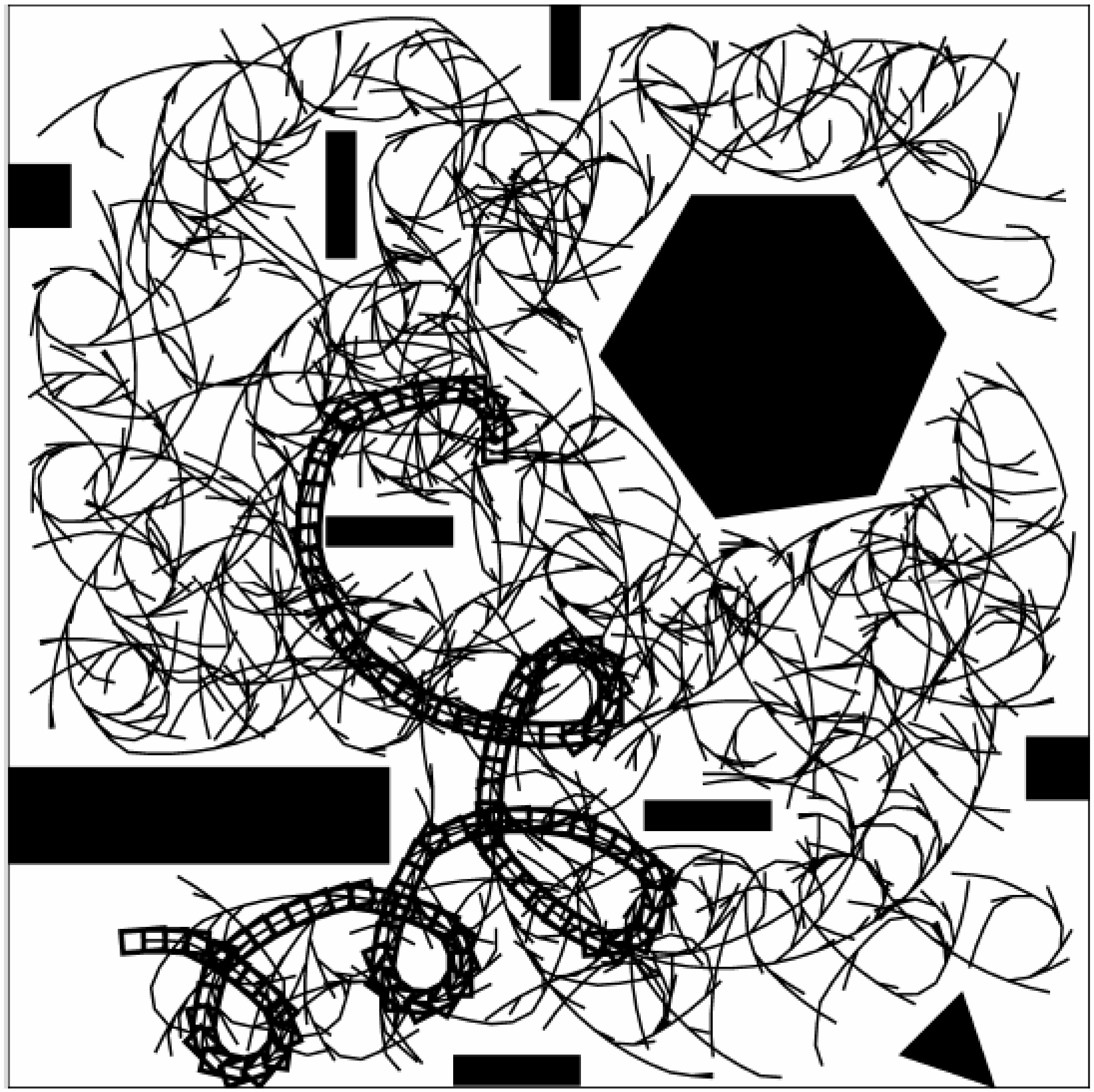
Hovercraft with 2 Thusters



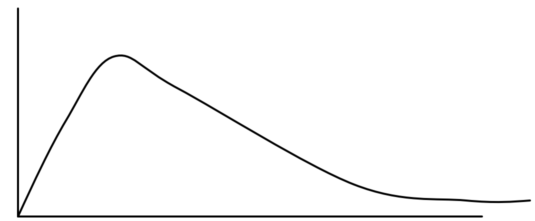
Out of This World Demo



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

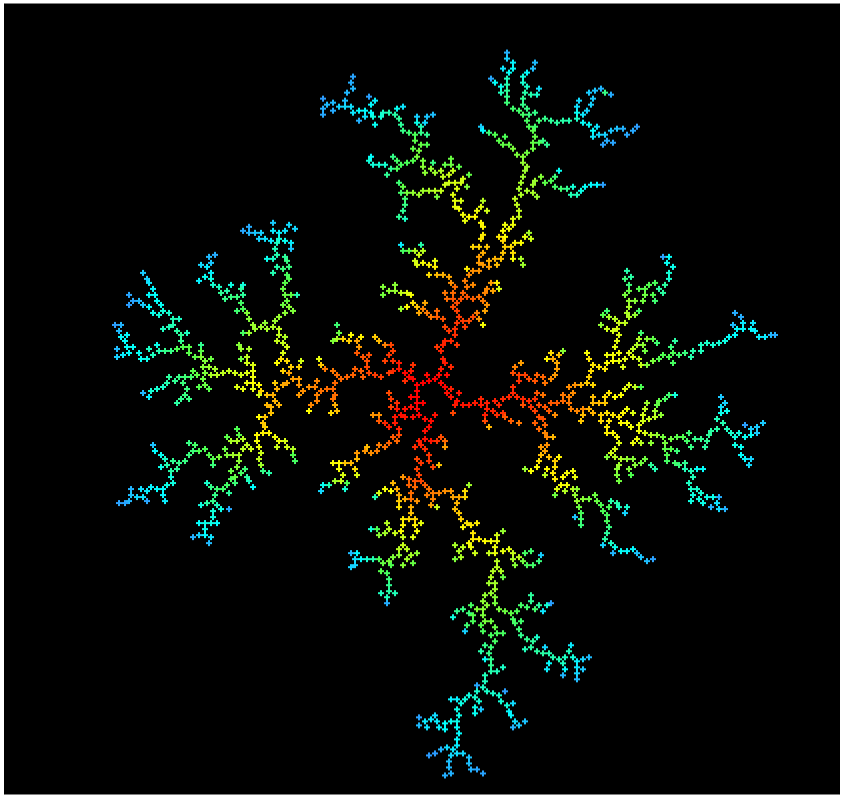
Robotics Applications
- mobile robotics
- manipulation
- humanoids

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

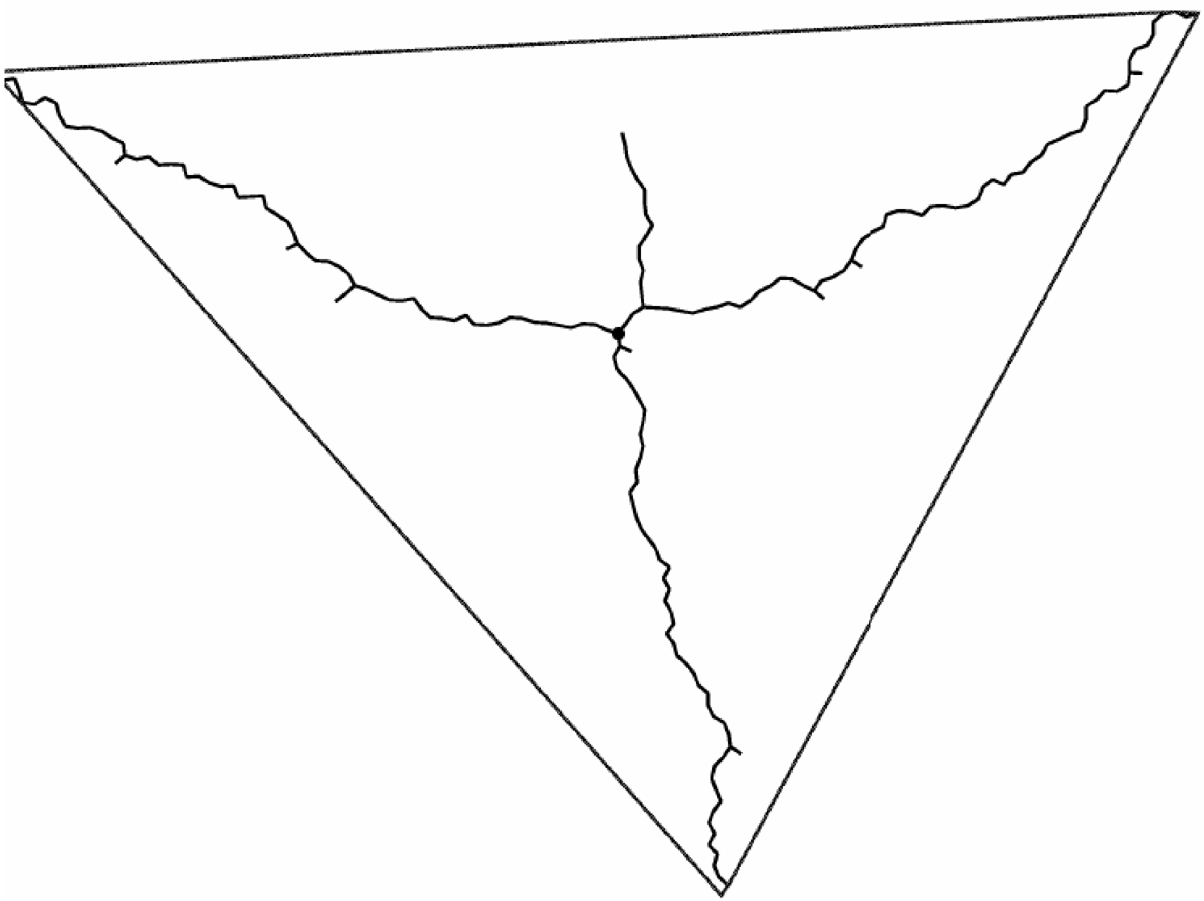
RRT extensions
- discrete planning (STRIPS and Rubik's cube)
- real-time RRTs
- anytime RRTs
- dynamic domain RRTs
- deterministic RRTs
- parallel RRTs
- hybrid RRTs

Diffusion Limited Aggregation

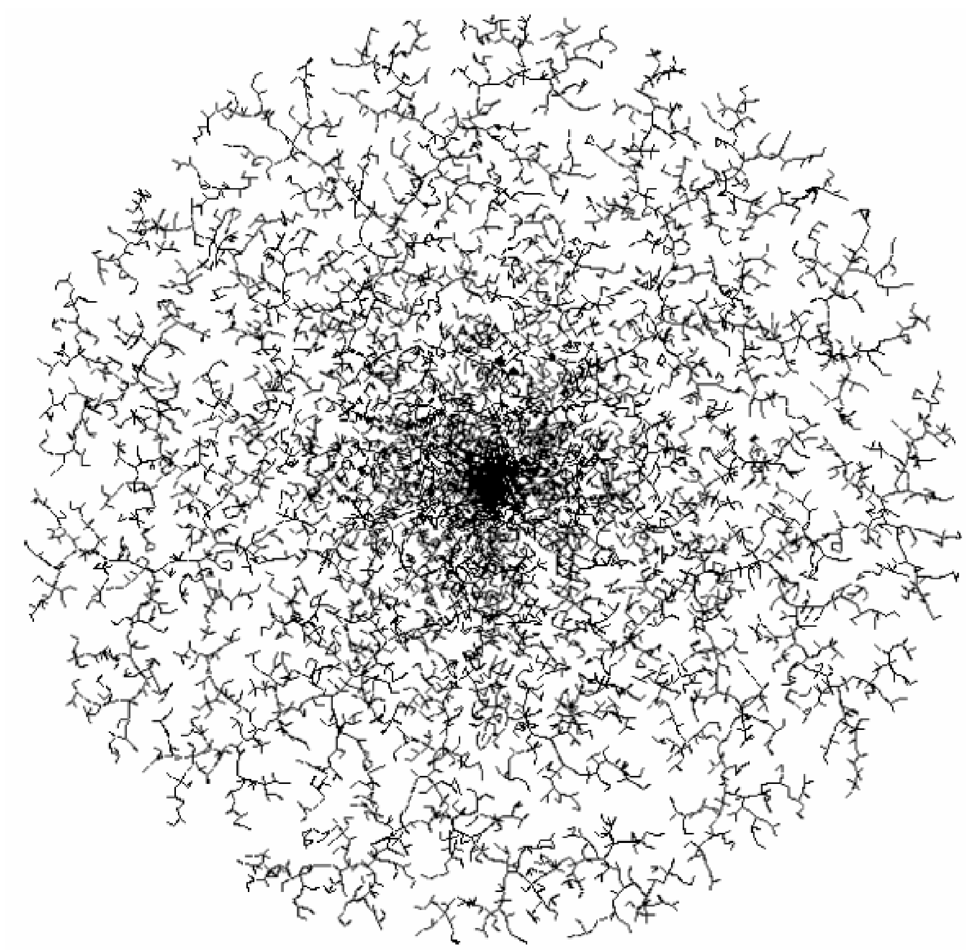
- Often used to model natural physical processes (e.g. snow accumulation, rust, etc.)



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