Motion Planning

Jana Kosecka
Department of Computer Science

- Discrete planning, graph search, shortest path, A* methods
- Road map methods
- Configuration space

Slides thanks to http://cs.cmu.edu/~motionplanning, Jyh-Ming Lien

1

Probabilistic Methods

- Resort to sampling based methods
- Avoid computing C-obstacles
  - Too difficult to compute efficiently
- Idea: Sacrifice completeness to gain simplicity and efficiency
- Probabilistic Methods
  - Graph based
  - Tree based

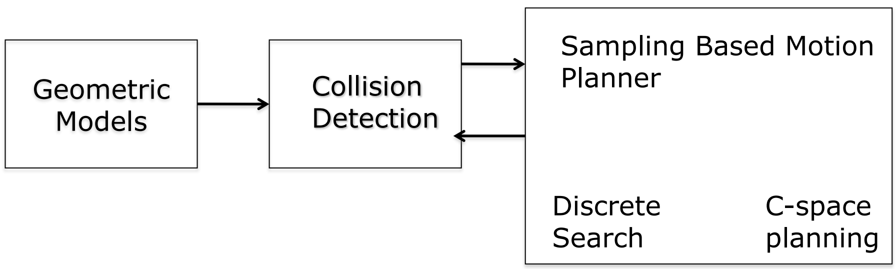
2

Sampling Based Motion Planning

- Recall: Algorithm is considered complete if for any input it correctly reports the path if it exists in finite amount of time
- Sampling based methods cannot achieve completeness
- Deterministic approach which samples densely is called Resolution complete
- Random Sampling Based Methods
  Probabilistically complete with enough samples the probability of finding solution approaches 1

3

Sampling Based Motion Planning

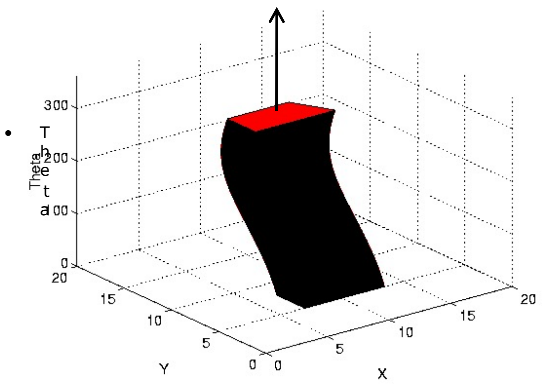


Idea : Generate random configurations
Check whether they are collision free
Connect them using Local planners
Discrete Search: (q0, qG) – single query search until you find qG
Multi-query search: Rapidly Exploring Random Trees

4

Probabilistic Roadmap Method
[Kavraki, Svestka, Latombe,Overmars 1996]

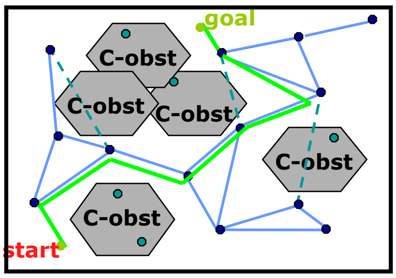
Explicit representation of the configuration space is unknown



6

Probabilistic Roadmap Method

C-space



Roadmap Construction (Pre-processing)

1. Randomly generate robot configurations (nodes
   - discard nodes that are invalid
2. Connect pairs of nodes to form **roadmap**
   - simple, deterministic *local planner* (e.g., straight line)
   - discard paths that are invalid

Query processing

1. Connect *start* and *goal* to roadmap
2. Find path in roadmap between *start* and *goal*
   - regenerate plans for edges in roadmap

7

Probabilistic Roadmap Method

- Important sub-routines
  - Generate random configurations
  - Local planners
  - Distance metrics
  - Selecting k-nearest neighbors (becoming dominant in high dimensional space)
  - Collision detection (>80% computation)

8

Metric in configuration space

- A metric or distance function *d* in a configuration space *C* is a function

  $$d : (q, q') \in C^2 \rightarrow d(q, q') \geq 0$$

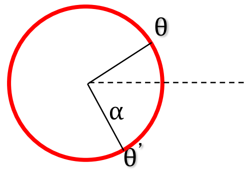
  such that

  - $d(q, q') = 0$ if and only if $q = q'$,
  - $d(q, q') = d(q', q)$
  - $d(q,q') \leq d(q,q'') + d(q'',q')$ aka. Triangle inequality

9

Examples in $R^2 \times S^1$

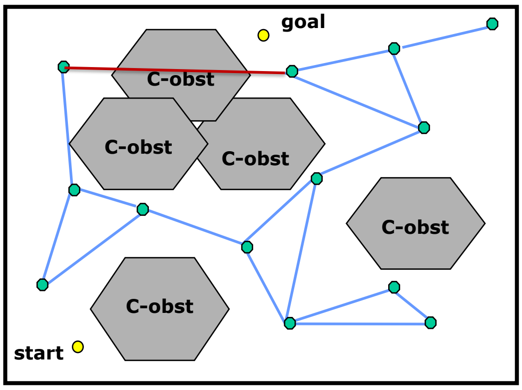
- Consider R^2 x S^1 (plane and circle)
  - $q = (x, y, \theta)$, $q' = (x', y', \theta')$ with $\theta, \theta' \in [0,2\pi)$
    $\alpha = \min \{ |\theta - \theta'|, 2\pi - |\theta - \theta'| \}$
    Distance between two angles



- $d(q, q') = \text{sqrt}((x-x')^2 + (y-y')^2 + \alpha^2))$
- $d(q, q') = \text{sqrt}((x-x')^2 + (y-y')^2 + (\alpha r)^2)$, where r is the maximal distance between a point on the robot and the reference point
- Examples of a distance between two configurations

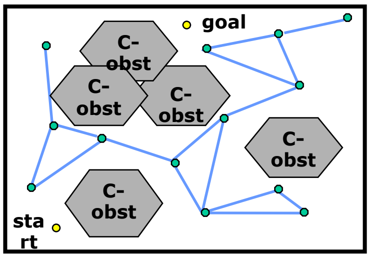
11

Local planner collision checking



12

PRMs: Pros & Cons



PRMs: The Good News

1. PRMs are *probabilistically complete*
2. PRMs apply easily to high-dimensional C-space
3. PRMs support fast queries w/ enough preprocessing

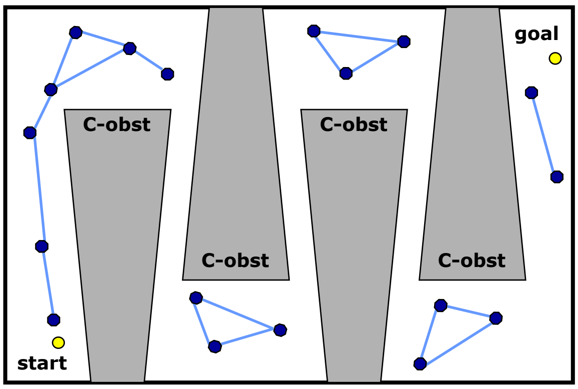
Many success stories where PRMs solve previously unsolved problems

PRMs: The Bad News

1. PRMs don't work as well for some problems:
– unlikely to sample nodes in *narrow passages*
– hard to sample/connect nodes on constraint surfaces

13

Problems with PRMs

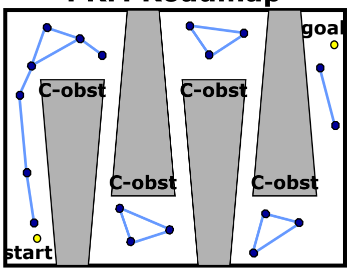


14

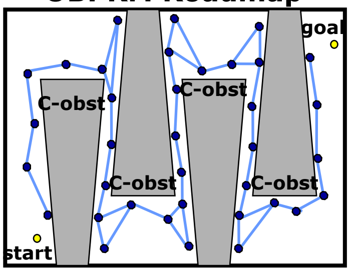
An Obstacle-Based PRM

To Navigate Narrow Passages we must sample in them
• most PRM nodes are where planning is easy (not needed)

PRM Roadmap



OBPRM Roadmap



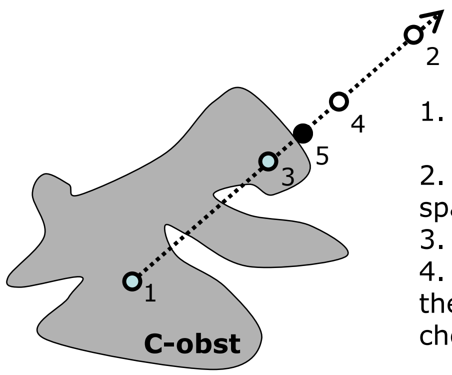
Idea: Can we sample nodes near C-obstacle surfaces?

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

15

Finding Points on C-obstacles

Basic Idea (for workspace obstacle S)

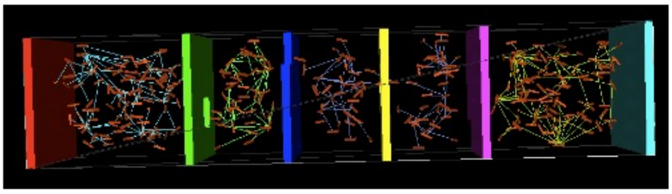


1. Find a point in S's C-obstacle (robot placement colliding with S)
2. Select a random direction in C-space
3. Find a free point in that direction
4. Find boundary point between them using binary search (collision checks)

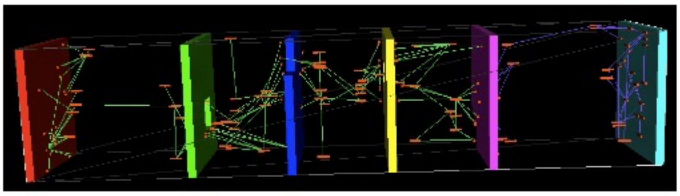
Note: we can use more sophisticated heuristics to try to cover C-obstacle

16

OBPRM



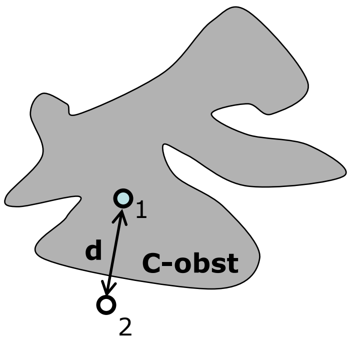
PRM
- 328 nodes
- 4 major CCs



OBPRM
- 161 nodes
- 2 major CCs

17

Gaussian Sampling PRM



1. Find a point in S's C-obstacle (robot placement colliding with S)

2. Find another point that is within distance *d* to the first point, where d is a random variable in a *Gaussian distribution*

3. Keep the second point if it is collision free

Note
- Two paradigms: (1) OBPRM: Fix the samples (2) Gaussian PRM: Filter the samples
- None of these methods can (be proved to) provide guarantee that the samples in the narrow passage will increase!

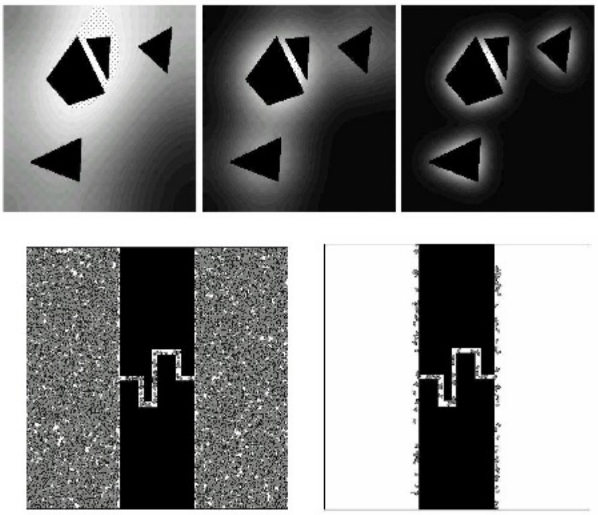
18

Related Work (selected)

- **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]
- Bridge test [Hsu et al 03]
- 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]
- Using workspace clues

19

Gaussians



20

Collision Detection

- Treated as black box - takes most of the computation
- In 2D convex robot and obstacle, there exist linear time collision detection algorithms
- Construct polygonal C_{obst}
- Define a logical predicate which indicates whether configuration is free or not
- Hierarchical Methods or Incremental Methods
- Section 5.3.4 Motion Planning Book

22

Collision Detection

- For more complex non-convex bodies – Hierarchical methods (create bounded regions – to avoid checking bodies which are far apart)
- Have a quick way of computing whether two regions intersect (Bound. Regions: spheres, axis aligned boxes), the composite bounding regions are represented by trees
- Incremental Methods

e.g. compute closest point distance at each iteration assuming that the robot does not move too much. Can be efficiently computed for 2D convex polygons (computing vertex-vertex, edge-vertex and edge-edge distances).

23

Collision Detection

- Previous methods:
- Check for collision free configuration
- Check for collision free path segment
- Consider that the path between two configurations is a straight line, parameterized by [0,1], sample the interval and check each sample whether its collision free

- For more details on alternative sampling strategies (Section 5.3.4 Motion Planning Book)
- There exist algorithms with guarantees – trickier to implement

24

Issues

- How do we determine a random free configuration
- We would like to sample nodes uniformly from C_{free}
- Draw each of the coordinates from the interval of corresponding DOF (use uniform probability per interval)
- For each sample check for collision between the robot and obstacles and robot itself

- If collision free add to V otherwise discard
- Collision detection and sampling – large topics
- Suitable for high-dimensional spaces

- Resulting path look very jerky, question how to follow them with smooth trajectories

25

Probabilistic Roadmaps

- Construct the road map for the entire configuration space

26

Planning in high dimensional spaces

- Single query planning $q_{initial}$ and q_{goal} are given once – no pre-computation (greedy search technique can take a long time)
- Multiple query planning – spreads out uniformly, requires lot of samples to cover the space
- Next incremental sampling and search methods that yields good performance without parameters tunning. Idea gradually construct search tree, such that it densely covers the space

27

Rapidly Exploring Random Trees

- Single query model – given start and goal q find a path
- Analogy with the discrete search algorithms
- Samples are states, edges are paths connected them (as opposed to actions previously)
- Graphs are undirected (Tree) Ingredients

1. Initialize the graph
2. Select vertex for expansion
3. Generate set of new vertices
4. For some new vertices run a local planner and check whether its collision free
5. If yes insert an edge to the graph
6. Keep on going until termination condition is satisfied

28

Incremental Search and Sample

- Why not just discretizing configuration space ?
- For high dimensions large number of states can be wasted exploring various cavities of the C-space
- For low dim spaces grid points themselves can serve as roadmap points (need to be checked for collisions etc)

- How to choose a resolution of the discretization (start coarse , iteratively refine)
- Another option – abandon discretization and work with continuous problem (like randomized potential fields) or RRT's

29

Rapidly Exploring Random Trees

30

Rapidly-Exploring Random Tree (RRT)

- Tree Based single shot planners – compute the respresentation of C_{free} for single start and goal

- RRTs: Rapidly-exploring Random Trees

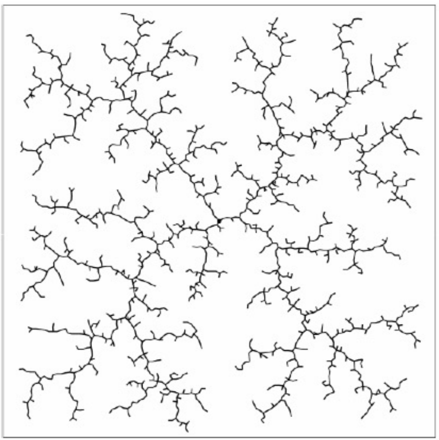
  Rapidly-exploring random trees: Progress and prospects. S. M. LaValle and J. J. Kuffner. In *Proceedings Workshop on the Algorithmic Foundations of Robotics*, 2000.)

  Incrementally builds the roadmap tree

- Extends to more advanced planning techniques
  - –Integrates the control inputs to ensure that the kinodynamic constraints are satisfied

31

Rapidly-Exploring Random Trees



Idea: Incrementaly construct the search tree, that improves with resolution
Previous incremental search methods could spend long time exploring nodes inside unimportant cavities

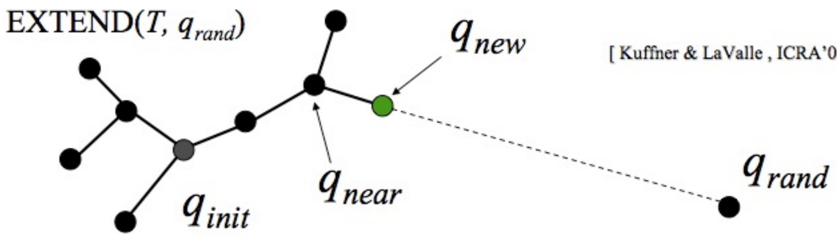
32

RRT's

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

Random sample connects to the nearest node so far

If the nearest point lies on an edge, the edge is split in two



[Kuffner & LaValle , ICRA'00]

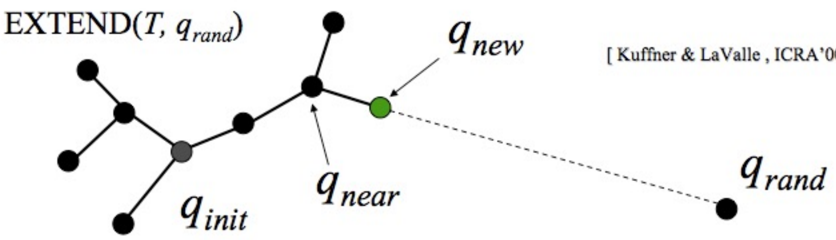
33

RRT’s

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

Details:
Step length: how far to sample
Sample just at the end point
Sample all along, small steps

Extend returns the new edge



[Kuffner & LaValle , ICRA’00]

34

RRT pseudo-code details

- Add start node to the tree
- Repeat n times
  - Generate random configuration x
  - If x is in free space using CollisionCheck

    find y, the closes node in the tree to the configuration x
    if dist(x,y) > delta – check if x is too far away from y
    find a configuration z that is along the path from x to y
    such that dist(z, y) <= delta
    x = z;
    if (LocalPlanner(x,y) – check if you can get from x to y
    if yes add x to the graph

35

Naïve Random Tree

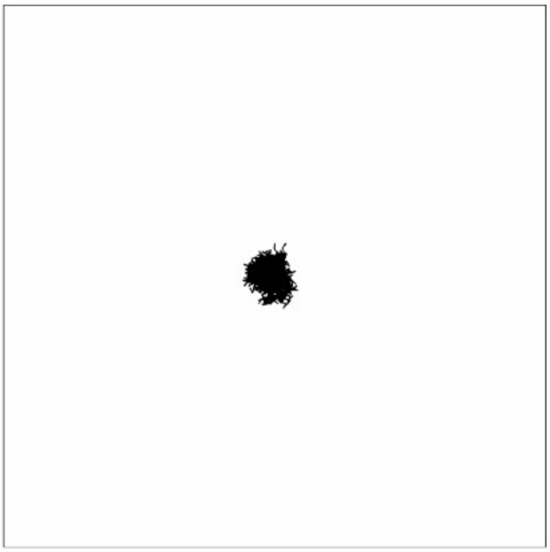
Start with middle

Sample near this node

Then pick a node at random in tree

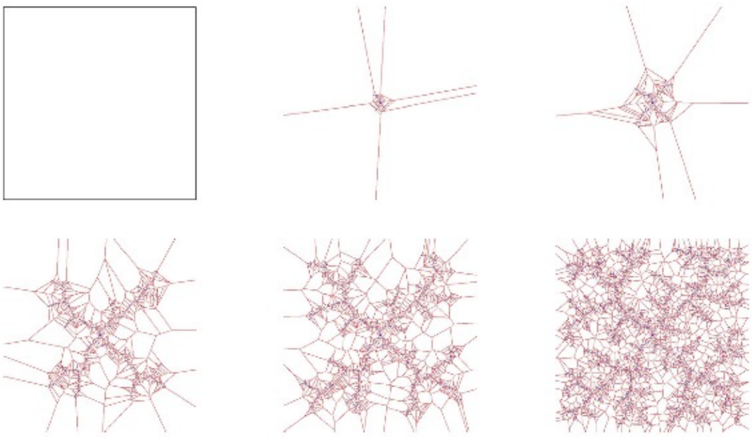
Sample near it

End up Staying in middle



36

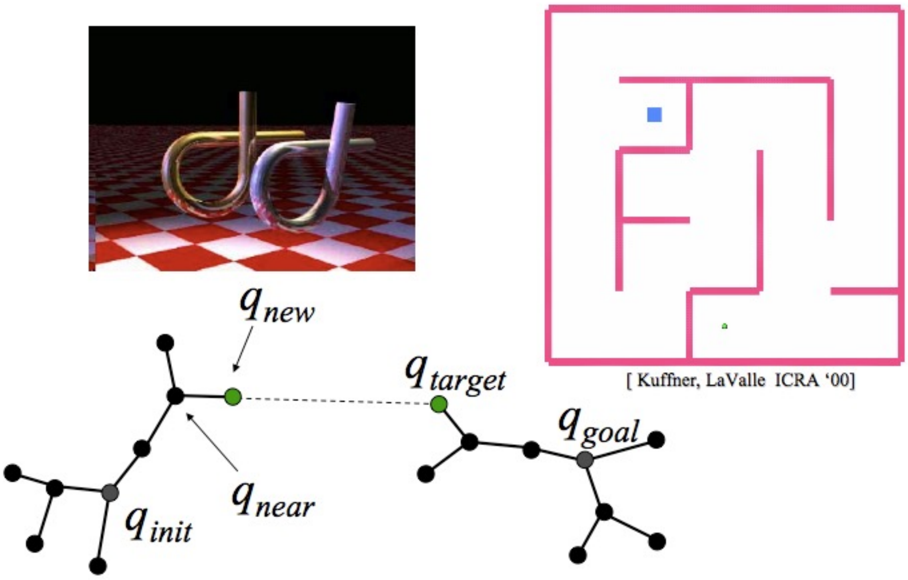
RRT's are biased towards large Voronoi cells



The nodes most likely to be closest to a randomly chosen point in state space are those with the largest Voronoi regions. The largest Voronoi regions belong to nodes along the frontier of the tree, so these frontier nodes are automatically favored when choosing which node to expand.

37

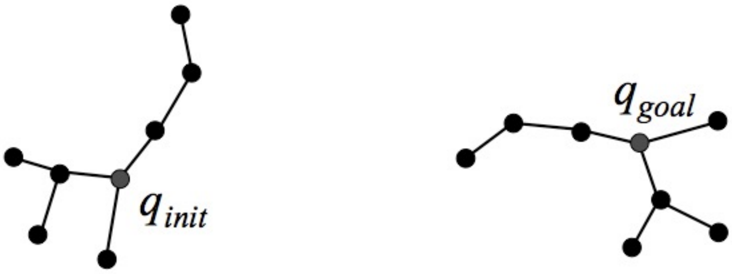
Grow two RRT's together



38

Two RRT's

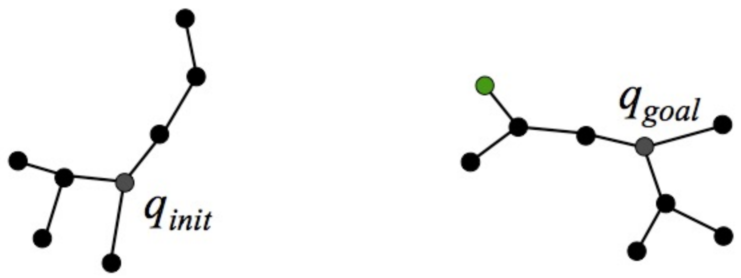
A single RRT-Connect iteration...



39

Two RRT's

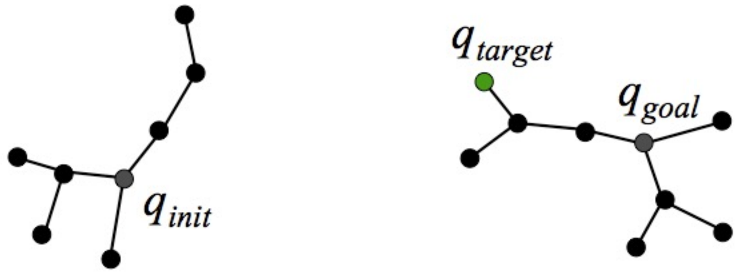
1) One tree grown using random target



40

Two RRT's

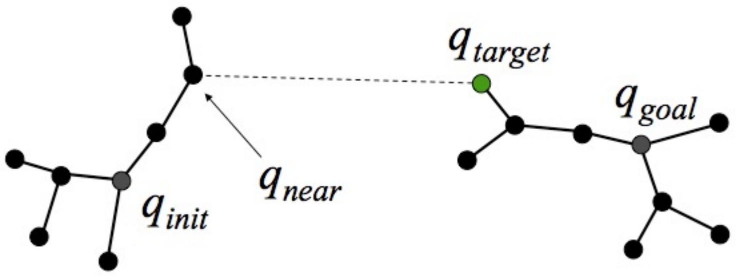
2) New node becomes target for other tree



41

Two RRT's

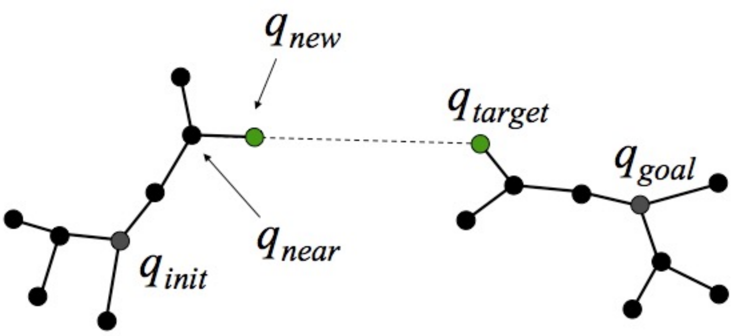
3) Calculate node "nearest" to target



42

Two RRT's

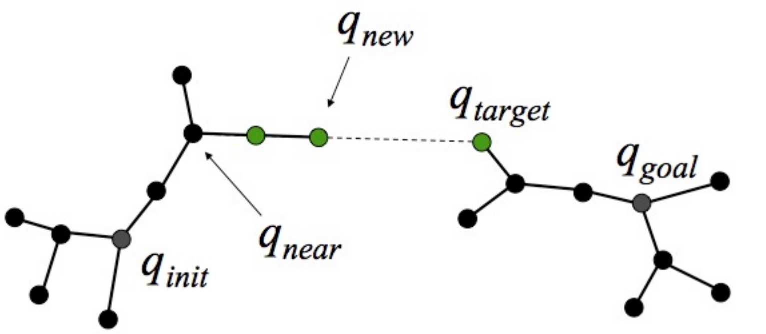
4) Try to add new collision-free branch



43

Two RRT's

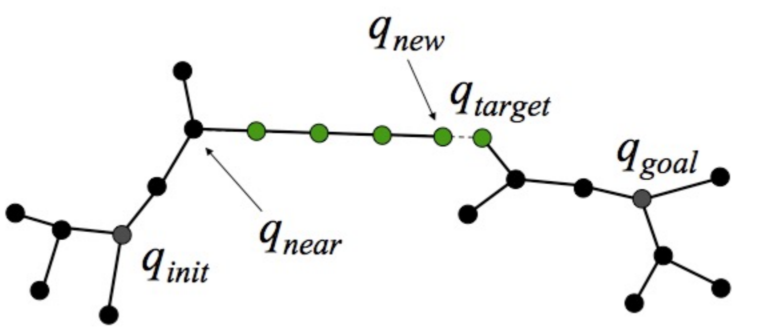
5) If successful, keep extending branch



44

Two RRT's

5) If successful, keep extending branch



45

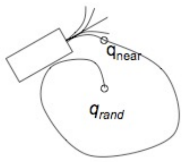
Taking actions into account

Instead of moving in a straight line for some distance, take into account kinematic constraints

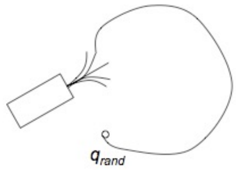
q_{near}

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

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



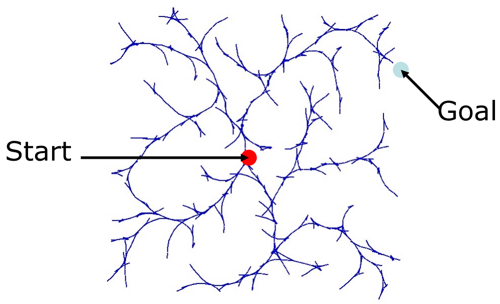
Is this the best?



46

How it Works

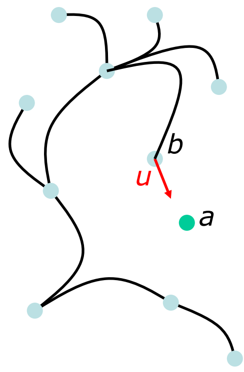
- Build a rapidly-exploring random tree in state space (*X*), starting at s_{start}
- Stop when tree gets sufficiently close to s_{goal}



47

Building an RRT

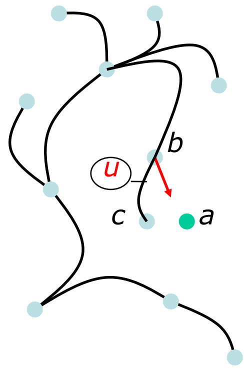
- To extend an RRT:
  - Pick a random point *a* in *X*
  - Find *b*, the node of the tree closest to *a*
  - Find control inputs *u* to steer the robot from *b* to *a*



48

Building an RRT

- To extend an RRT (cont.)
  - Apply control inputs *u* for time δ, so robot reaches *c*
  - If no collisions occur in getting from *a* to *c*, add *c* to RRT and record *u* with new edge



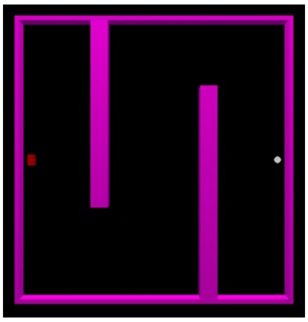
49

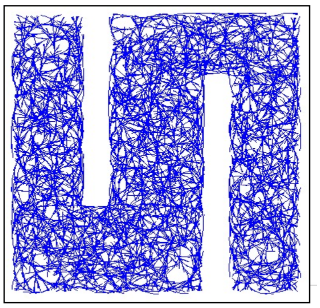
Executing the Path

- Once the RRT reaches s_{goal}
  - Backtrack along tree to identify edges that lead from s_{start} to s_{goal}
  - Drive robot using control inputs stored along edges in the tree

50

Problem of Simple RRT Planner



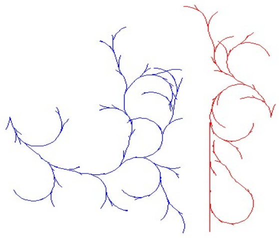


- Problem: ordinary RRT explores X uniformly
  → slow convergence
- Solution: bias distribution towards the goal – once in a while choose goal as new random configuration (5-10%)
- If goal is choosen 100% time then it is randomized potential planner

51

Bidirectional Planners

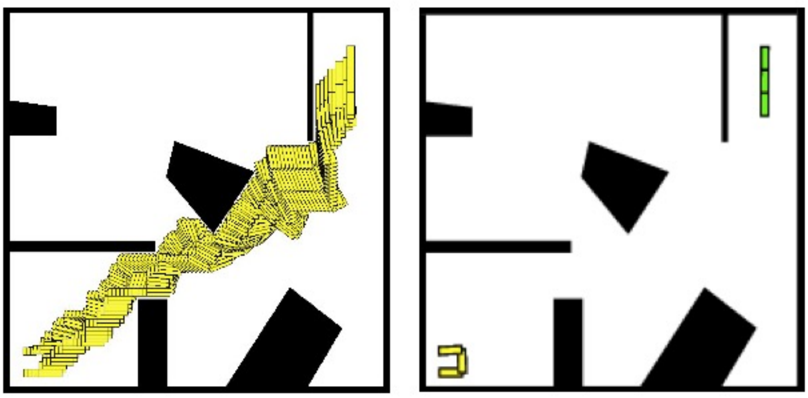
- Build two RRTs, from start and goal state



- Complication: need to connect two RRTs
  - local planner will not work (dynamic constraints)
  - **bias** the distribution, so that the trees meet

52

Articulated Robot example



54

RRT's

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

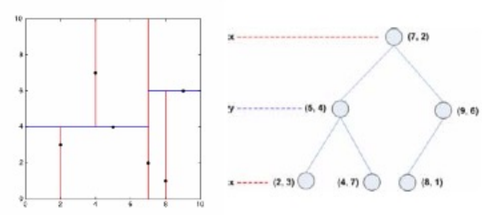
- Issues/problems
- Metric sensitivity
- Nearest neighbour efficiency
- Optimal sampling strategy
- Balance between greedy search and exploration

- Applications in mobile robotics, manipulation, humanoids, biology, drug design, areo-space, animation
- Extensions – real-time RRT's, anytime RRT's dynamic domains RRT'sm deterministic RRTs, hybrid RRT's

55

Efficient nearest neighbour algorithms

- How to find NN in high dimensional spaces



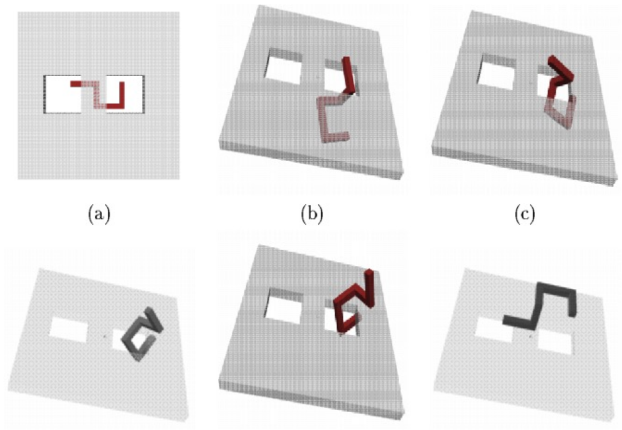
- KD trees – recursively choose a plane P that splits the set

  evenly in a coordinate direction
- Store P at the node
- Apply to children sets Sl and Sr
- Requires O(dn) storage

- Various hashing strategies

56

Computed example



(a) (b) (c)

(d) (f) (g)

63

Conclusion

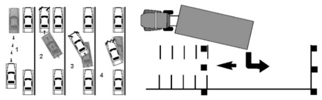
- Motion planning is difficult (intractable)
- Roadmap methods
  - Probabilistic Motion Planners

We will return to planning when considering partial information, dynamically changing worlds, uncertainty

64

What is not covered?

- Other types of motion planning
  - With constraints
    - Close-chain constraint
    - Nonholonomic constraint
    - Differential constraints
  - Manipulate planning
  - Assembly planning
  - Planning with uncertainty
  - Planning for multiple robots, dynamic env
  - Planning for highly articulated objects
  - Planning for deformable objects
  - …

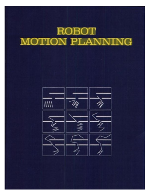


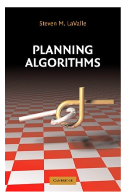
Little Seiko

65

Additional Readings

- **Gross motion planning—a survey**, Y. K. Hwang and N. Ahuja, ACM Computing Surveys, 1992 (survey paper)
- **Robot Motion Planning**. J.C. Latombe. Kluwer Academic Publishers, Boston, MA, 1991.
- **Motion Planning: A Journey of Robots, Molecules, Digital Actors, and Other Artifacts**. Jean-Claude Latombe, IJRR, 1999 (survey paper)
- **Planning Algorithms**, Steven LaValle, 2006, Cambridge University Pres, (Free download at http://planning.cs.uiuc.edu/)





66

Examples

- Road Map methods and behavior based strategies
- Homing https://parasol.tamu.edu/dsmft/movies/flocking_Homing_web.mpg
- Flocking, Goal Search

- https://parasol.tamu.edu/dsmft/movies/flocking_GoalSearch_web.mpg
- https://parasol.tamu.edu/dsmft/movies/flocking_RBFlock_narrow_homing.mpeg

67