Roadmap-Based Planning

• **Step 1: Preprocessing**
  Build a connected roadmap which is accessible from any point in $C_{\text{free}}$

• **Step 2: Query**
  Given a start and a goal state, connect them to the roadmap and search for shortest path
Roadmap-Based Planning

• Great for multi-query planning

• Expensive preprocessing for single-query!
  • Ensure connectivity
  • Ensure coverage
  • Different ways of constructing quality roadmaps
  • Collision checking to remove roadmap states in self-collision
Ideas?

• **Termination Condition**
  • Start and Goal states in the same connected component

• **Idea 1**
  • Set $M = 1$
  • Iteratively sample a point and connect to the graph $[kNN, r\text{-disk}]$
  • Search the graph but multiple islands form.

• **Idea 2**
  • Keep track of roadmap $G_s$
  • Connect to $G_s [kNN, r\text{-disk}]$
  • When the single connected component engulfs goal, search over the graph.

• **Idea 3**
  • Make the graph a tree!

• Anything more?
Rapidly-Exploring Random Trees (RRT)

- Effective for single-query planning in high dim.
- No preprocessing step!
  - Begin with start configuration
  - Build tree towards goal configuration
  - Terminate
- RRT-Connect, RRT*
- Kinodynamic Planning
RRT: Algorithm

1 \text{T.init}(x_{\text{init}})

2 For i = 1 to N:
3 \text{sample} random configuration \( x_{\text{rand}} \) in \( C_{\text{free}} \)
4 Find \textbf{nearest} milestone \( x_{\text{nearest}} \) in \( T \)
5 \textbf{Extend} \( x_{\text{nearest}} \) towards \( x_{\text{rand}} \)
6 If extended to \textbf{near goal}, terminate with success

7 Terminate with failure
RRT: Algorithm
RRT: Extend

Extends the tree by a small step towards $x_{\text{rand}}$
RRT: Implementation Details

• **Goal Biasing**
  • Sample $x_{\text{rand}}$ uniformly from $C_{\text{free}}$ with probability $1 - p_{\text{bias}}$ and uniformly from $x_{\text{goal}}$ with probability $p_{\text{bias}}$
  • Rule of thumb: Use $p_{\text{bias}}$ of 0.05

• **Connection Strategy**
  • Captured with the step size. How?
RRT: Postprocessing

- Paths produced by RRT can be arbitrarily bad
- Often characterized by unnecessary turns
RRT: Postprocessing

- Often a time-consuming step
- Can be non-trivial in kinodynamic planning
RRT: Postprocessing

Figure adapted from [Kuffner, LaValle00]
RRT: Postprocessing

Figure adapted from [Kuffner, LaValle00]
RRT: Postprocessing

Figure adapted from [Kuffner, LaValle00]
RRT-Connect

• How do we speed up RRT?
• Grow trees from both start and goal!
RRT-Connect
RRT-Connect
RRT-Connect

• Which tree do we extend?

• Issue: Connection between the two trees.
  • Can the connection always be exact?
Theoretical Properties of RRT

• Rapid Exploration

• Probabilistically Complete

• (Low) Quality of Solution
  • Non-Zero probability of non-optimal solution even as number of samples goes to infinity.
RRT*
RRT*

\(x_{\text{init}}\)

\(x_{\text{nearest}}\)

\(x_{\text{rand}}\)
RRT*
RRT*
RRT*
RRT*
RRT*
RRT*
RRT*
RRT*
RRT*
RRT*
RRT*
RRT*
RRT*
RRT*
RRT*
RRT*
RRT*
RRT vs RRT*

RRT* is asymptotically optimal
RRT: Regions of Improvement

• Improving the quality of RRT solution
  • Sampling schemes (Alternative EST, WIS, Learned Samplers)
  • Postprocessing techniques (Shortcutting)
  • Changing the connection scheme (kNN, r-disc)
  • Use heuristics to bias sampling (hRRT, typically slower than RRT)

• Improving the convergence rate of RRT
  • Lazy computation (LazyRRG* much faster than RRT*)
  • Bi-directional search (Bi-RRT / RRTConnect)
  • Bounded sub optimality for speed (LBT-RRT)
RRT Summary

**Setting:** Single-query motion planning

**Common approach:** Sampling-based \( (\text{RRTs}) \)

**Optimize:** Path-length

- **RRT** [LaValle Kuffner01] — Fast, not optimal
- **RRG, RRT\*** [Karaman Frazzoli 11] — Slower, asymptotically optimal
RRT in Action
RRT or PRM?

- Single Query vs Multi Query
- Preprocessing vs Postprocessing
- Difficulty of the problem: Amount of free space in C-Space

What other practical considerations?