RMP- RRT & OBPRM

Key words	Obstacle-Based PRM, RRT, RRT*, kinodynamic RRT, non-holonomic systems, single-query problems
Status	note complete

Rapidly-exploring Random Trees & RRT*

- motivation:
  - **single query problems**: multiple-query PRM expands whole free space —> **resource waste**
  - motion constraints for **non-holonomic robots** need to be included in planning phase.

▼ Process: (straight line graph)

1. select **start** node **randomly**
2. **randomly** select second node: check connectivity between first and second node(linear collision test).
3. **randomly** select 3rd node: **connect** the node to the **closest existing node.**
4. If **collision** with obstacle, place the new node along the connecting line **closest to obstacle.** —> improve path along narrow passages
5. iteratively grow the trees.

—> weighting possible (towards goal), growth limit possible (max. distance to closest node)

▼ When do PRM and RRT work well?

- **high-dimenional** problems
- the free space F is $(\epsilon, \alpha, \beta)$ -expansive, **where ϵ, α, β is large**

- Problem: **not optimal**

▼ Solution: **RRT* — Rewiring graph structure**

- for each new node, **check its log(n) neighbourhood**
- if **shorter path** exists, pick that path

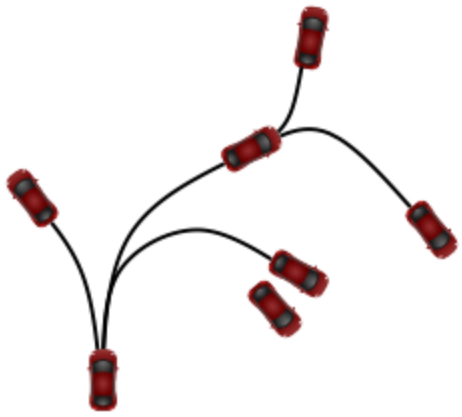
—> shortest path guaranteed. **asymptotically optimal.**

▼ Other improvements:

- accelerate the planner: decompose the free space into multiple expansive components
- increase expansiveness: use geometric transformations
- integration with new planners

Non-holonomic robots: Kinodynamic RRT

- systems with differential constraints
- nodes are not connected with straight lines, but **motion profile of a car —> clothoids**



PRM (mulitple/single-query):

▼ Advantages:

- probabilistic complete
- easy to apply in **higher dimensional** C-space
- support fast queries with enough preprocessing

▼ Disadvantages:

- unlikely to sample nodes in narrow passages
- hard to sample/connect nodes on constraint surfaces

Obstacle-Based PRM

- sample around the obstacle boundaries —> ensure to look into narrow passages.

▼ Process:

1. find a point in the C-obstacle
2. select a random direction, find a free point.
3. **binary search:** find the **boundary point** of obstacle
4. repeat for multiple iterations.

- OBPRM VS. PRM:
  - only sample around obstacles, more nodes through narrow passages
  - less nodes
  - improves connectivty between spaces