Robot Motion Planning

Bug 2

Probabilistic Roadmaps
New leave point condition: $d < d(H_j, \text{Target})$
Bug 2 Algorithm

1. From point $L_{j-1}$ move along M-line until:
   a. Target is reached. Stop
   b. An obstacle is hit at $H_j$. Goto 2

2. Turn left and follow the boundary until:
   a. Target is reached. Stop
   b. M-line met at distance $d$ from target such that:
      \[ d < \text{dist}(H_j, q_{\text{target}}) \]
      Define $L_j$, set $j=j+1$, and goto 1.
   c. If we return to $H_j$ without meeting the M-line, stop. The target is trapped.
A Hard Example for Bug 2
Some results

- $N_i =$ number of intersections of $B_i$ with M-line

- **Lemma:** in Bug2, any segment of the boundary is passed at most $N_i$ times

- **Theorem:** the length of the path verifies

  $$P = D + \sum \frac{n_i p_i}{2}$$
Bug + Noncontact Sensors

- Suppose the robot is provided with info within a disk of radius $r$
- E.g.: vision, infra-red, ultrasonic
- The robot reconstructs at each point, the path that would generate with no info
- Result: smooth version of previous paths
Bug + Non Contact Sensor
Probabilistic Roadmaps

- Path planning algorithm are expensive
- If #DOFs’ is large, deterministic algorithms are not effective
- Solution: use a probabilistic roadmap planner (PRM).
- PRM are complete in a probabilistic sense
- An heuristic planner: potential fields.
  Disadvantage: local minima. Solution: randomize to escape local minima
PRM: Description

- Roadmap = undirected graph \( R = (N, E) \)
- \( N \) : (nodes) set of selected configurations in \( C_{\text{free}} \)
- \( E \) : (edges) collection of simple paths. The Local Paths
- Local paths are computed by the fast but not powerful local planner
- Idea: connect \( q_{\text{init}} \) and \( q_{\text{goal}} \) with \( q_{\text{init}}' \) and \( q_{\text{goal}}' \) in \( N \)
- Search \( R \) for a path
Preprocessing Phase

Three stages

1. **Roadmap construction.** Objectives:
   a) Obtain reasonable connected graph
   b) Be sure “difficult regions contain a few nodes

2. **Roadmap expansion.** Objectives:
   Improve graph connectivity by selecting nodes of $R$ which lie in (heuristic) difficult regions and adding nodes there

3. **Roadmap Component reduction.** Optional. Attempts to simplify the graph
Roadmap Construction

- Initially $R$ is empty
- Repeatedly, generate & add a free configuration to $R$
- For every new $q$, select some nodes in $R$ and try to connect them to $q$ using local planner
- On success, add the edge $(q, q')$ to $E$

$\Delta: C_{\text{free}} \times C_{\text{free}} \rightarrow \{0,1\}$ is a local function returned by local planner

$D$: pseudo metric in $C_{\text{free}}$
Create random configurations
Update Neighboring Nodes’ Edges
End of Construction Step

Connected nodes
Expansion Step
End of Expansion Step
Select start and goal
Connect Start and Goal to Roadmap
Find the Path from Start to Goal
Roadmap construction algorithm

1. \( N \leftarrow \emptyset \quad E \leftarrow \emptyset \)
2. Loop
3. \( q \leftarrow \) randomly selected free configuration
4. \( N_q \) a set of candidate neighbors of \( q \) from \( N \)
5. \( N = N \cup \{q\} \)
6. \( \forall q' \in N_q \) in order of increasing \( D(q, q') \), do
7. If \( q, q' \) are not in the same connected comp. and \( \Delta(q, q') = 1 \), then \( E \leftarrow E \cup \{(q, q')\} \)
8. Update \( R \)'s connected components
Creation of Random Configurations

• Nodes of \( R = \) uniform random sampling of \( C_{\text{free}} \)

• Obtain \( q \) by drawing each of its coords. From allowed interval for the corresponding DOF

• Check \( q \) for collisions

• If \( q \) is collision-free, add it to \( N \). Otherwise discard
The Local Planner

• Deterministic vs. Non-deterministic planner

• Powerful planner = slower planner

• Empirical results: use deterministic and simple but very fast planner

• Example: connect two configurations by a straight line in configuration space

• Check for collisions by bisection
• Choice of $N_q$ is important. Local planner is used for all $q$’s in $N_q$

• $N_q = \{q' \in N / \text{MaxDist} \geq D(q, q')\}$

• Skip nodes in the same connected component

• Heuristics: bound the size of $N_q$ by some $K$. This gives independence on the size of $R$
Distance Function

• \( D \) is used for constructing & sorting \( N_q \)
• Hopefully, \( D \) reflect the chance that the local planner will fail
• Simple selection that works in practice:

\[
D(q, q') = \max_{x \in \text{Robot}} \| x(q) - x(q') \| \]
Roadmap Expansion

• $N$ should be a fair enough covering of $C_{\text{free}}$

• $R$ often consists of a few large components and several small ones

• $\text{Expansion} = \text{add nodes to form a large component with as many nodes as possible}$

• Try to cover the “difficult” parts of $C_{\text{free}}$

• Define weight $w(q)$. Large $w \Rightarrow q$ is in a difficult region

• Add $M$ nodes to the collection