CS 294-115 Algorithmic Human-Robot Interaction **Fall 2017**

Lecture 4: Motion Planning - Part III

Scribe: Jason Zhang

4.1 Sampling-Based Planners

Idea: Represent C_{free} via a sampled "road map."

4.1.1 Probabilistic Road Maps (PRM) [Kavraki '96]

Given:

- A collision checker γ such that

$$\gamma(q) = \begin{cases} 1 & \text{if } q \in C_{\text{obs}} \\ 0 & \text{else} \end{cases}$$

- A simple planner B_s such that

$$B_s(q_1, q_2) \rightarrow \begin{cases} \text{A path if it finds one quickly} \\ \text{Failure otherwise} \end{cases}$$

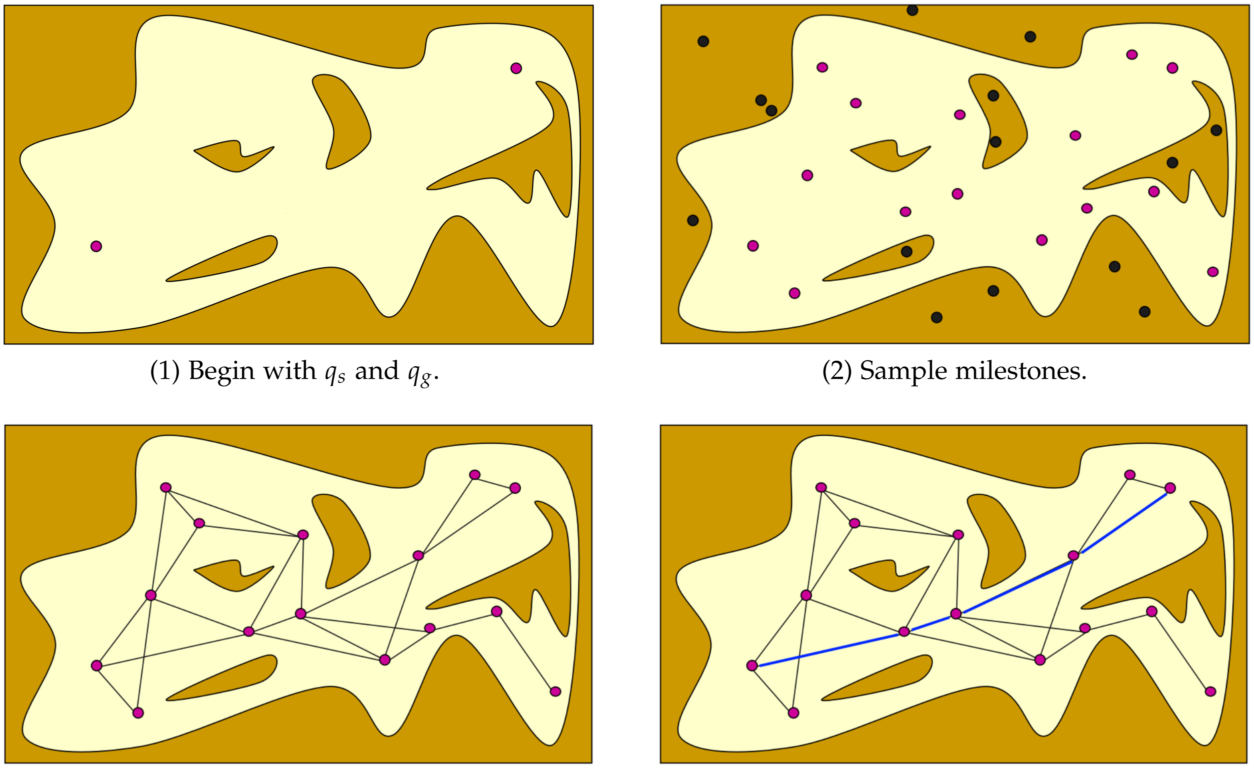
  Note that B_s does not necessarily return a path if one exists. An example of B_s is collision-checking the straight line connecting q_1 and q_2.

Algorithm:

1. Start with the set $\{q_s, q_g\}$.
2. Sample M milestones in C_{free} using rejection sampling to avoid points in C_{obs}.
3. Try to connect all milestones using B_s. Three such options for doing so are:
   (a) Connect all pairs.
   (b) Connect everything in an R-disc, connecting configurations within a distance of R (S-PRM).
   (c) Connect the k-nearest neighbors of each configuration (k-PRM).
4. If q_s and q_g are in the same connected component, find the path using graph search. Otherwise, return to step 2 and sample more milestones.

See Figure 4.1 for a visual example of PRM.

Note: The original PRM algorithm was meant to be a multi-query planner, thus able to deal with multiple queries $\{(q_s, q_g)^{(i)}\}_{i=1}^{N}$ rather than a single query (q_s, q_g). Since C_{obs} usually remains the same, we can build a road map during pre-processing.



(1) Begin with q_s and q_g.

(2) Sample milestones.

(3) Connect milestones.

(4) Attempt to find a path from q_s to q_g.

Figure 4.1: The PRM algorithm in action. The yellow region represents C_{free} while the tan region represents C_{obs}.

4.1.2 Rapidly-Exploring Random Trees (RRT) [Lavalle and Kuffner '01]

From a PRM, we can develop the Bi-directional Rapidly-Exploring Random Tree (Bi-RRT) using three ideas:

Idea 1: $M = 1$. Stop when q_s and q_g are in the same connected component.

Idea 2: Keep track of CC_s and CC_g, the connected components that contain q_s and q_g.

Idea 3: 1-PRM (a k-PRM with k=1). Connect the newest milestone to the closest point in CC_s and CC_g.

4.1.2.1 Pros and Cons

(+) Fast and scalable. Works well even with high DOF.

(+) Does not require overly restrictive assumptions.

(+) Probabilistically complete (i.e. $\lim_{n\to\infty} P(\text{sol}|\exists\text{sol}) = 1$). This is a weak pro as even a planner that randomly samples paths would be probabilistically complete.

(-) Not optimal.

(-) Requires a lot of post-processing.

4.1.3 Shortcutting

As a post-processing step, randomly sample two points to see if they can be connected. Skipping intermediate vertices can help smooth out the path.

4.1.4 Kinodynamic Planning

Kinodynamic motion planning is the problem of finding the optimal path for a robot with motion constraints such as velocities and accelerations from a starting configuration to a goal configuration. In addition to avoiding collisions, a kinodynamic RRT must also satisfy local differential constraints.

When planning in a control space instead of a configuration space, we can attempt multiple sequences of controls and keep the trajectory that results in the closest point to the goal. This allows us to avoid explicitly solving the 2-point boundary value problem. Note that we don't directly sample controls because that would lead to "hairballing."