Lecture 17: *Motion Planning I*

Scribe: *Duong Le, Keyu Han, Baiyu Huang, Seunghee Yoon*

1 Complete Motion Planning

1.1 Cell Decompose

1.2 Visibility Graph

2 Sampling-based Motion Planning

Sampling-based motion planning is not complete: not guarantee to find a solution when one exists. However, as the number of samples goes to infinity, there is a strong guarantee that it can find a solution.
There are 2 types of sample-based planning algorithms:

Multiple Query Algorithm

- Roadmap is built beforehand. Then it can be used multiple times for different queries.
- Query is fast since roadmap is already generated.
- Keeping roadmap all the time is expensive.
- Can only deal with static environment - has to rebuild the roadmap when environment changed.

Single Query Algorithm

- No roadmap is built beforehand. Find a path from start to goal then finish
- Query is slower.
- No saved roadmap - better memory
- Can deal with dynamic environment

2.1 Probabilistic Road Map (PRM)

Probabilistic Roadmap (PRM) is a multi-query algorithm.
There are 2 steps:

1. Preprocessing state: build roadmap
2. Query state: search roadmap given start and goal

2.1.1 Roadmap Construction

Algorithm 6 Roadmap Construction Algorithm

Input:

n : number of nodes to put in the roadmap

k : number of closest neighbors to examine for each configuration

Output:

A roadmap $G = (V, E)$

```
1:  V ← ∅
2:  E ← ∅
3:  while |V| < n do
4:      repeat
5:          q ← a random configuration in 𝒬
6:      until q is collision-free
7:      V ← V ∪ {q}
8:  end while
9:  for all q ∈ V do
10:     N_q ← the k closest neighbors of q chosen from V according to dist
11:     for all q' ∈ N_q do
12:         if (q, q') ∉ E and Δ(q, q') ≠ NIL then
13:             E ← E ∪ {(q, q')}
14:         end if
15:     end for
16: end for
```

Figure 1: Roadmap Construction Algorithm

Let $\Delta(q, q')$ be a function that returns either a collision-free path from q to q' or NIL if it cannot find such a path. There are many algorithms can be used for Δ. For example, if we want to check whether there is a straight-line path from q to q, we can use binary search to check for collision.
Figure 6 shows an algorithm to construct a roadmap:

- Line (1) and (2): Initialize graph $G = (V, E)$ to be empty.
- Line (3) to (8): describes sampling process: keep sampling to get n numbers of collision-free samples.
- Line (9) to end: describes process to build roadmap from n samples: for every sample node $q \in V$, create a set N_q of k closest neighbors. Then from every $q' \in N_q$, whenever function $\Delta(q, q')$ succeeds to compute a collision-free path from q to q', the edge (q, q') is added to E. Figure 2 shows an example of roadmap using Δ as straight-line planner

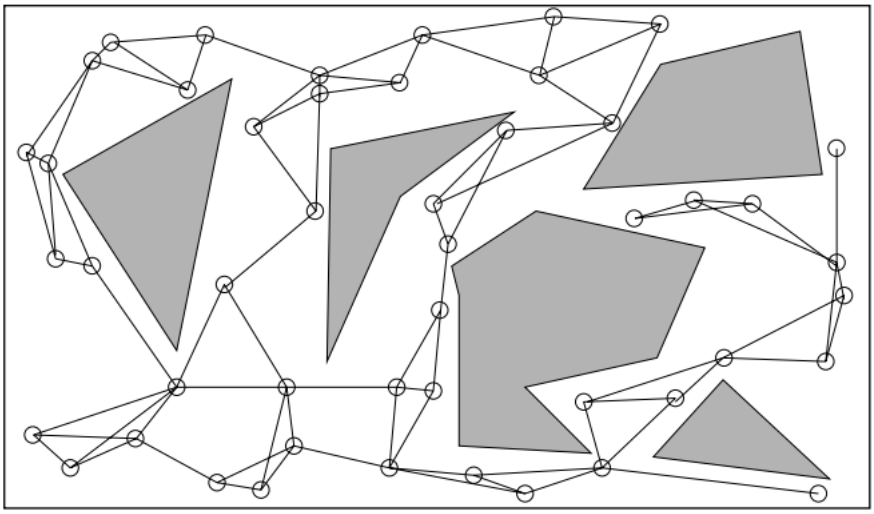

Figure 2: An example of a roadmap

2.1.2 Query

When you have a start point and end point, you connect them to the graph and then using graph searching algorithm to find path from start to end.

2.1.3 Failure

Some reasons to failure:

- Connectivity: disconnected graph

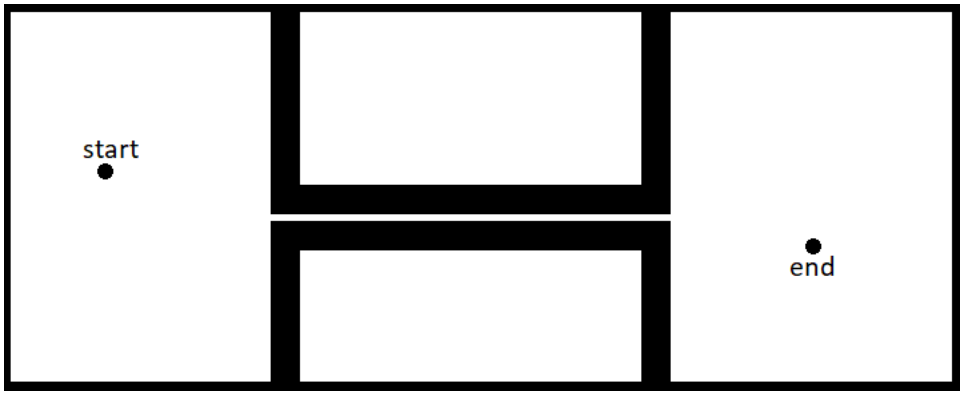

Figure 3: An example when obstacles are very closed to each other

- Obstacles are very closed to the others.

Solution: generate more samples.

But in the case when obstacles are closed to the others, the probability to sample between the obstacles is very unlikely. For example, in figure 3, the chance to sample points in the narrow road between 2 obstacles is very low. If you keep sampling more points, the graph will become so dense and will take a lot of time to find a path between start and end point.
Solution: sample more points toward edges of the obstacles. We can use binary search to find the closest point to the obstacle that's collision-free, and then sample more points near that point. The process of sampling becomes more expensive, but result graph is less dense and faster to compute path.

2.2 Visibility PRM

The roadmap is constructed incrementally by randomly sampling the configuration space and attempting to connect some pairs of collition-free samples by the local method. The visibility roadmaps are build without any explicit computation of the visibility domains.

2.3 Principle

The algorithm that we propose below is general. It allows us to build visibility roadmaps without requiring any explicit computation of the visibility domains. The roadmap is constructed incrementally by randomly sampling the conguration space and attempting to connect some pairs of collision-free samples by the local method. Figure 2 illustrates the principle of the sampling strategy used at each iteration of the algorithm. Randomly

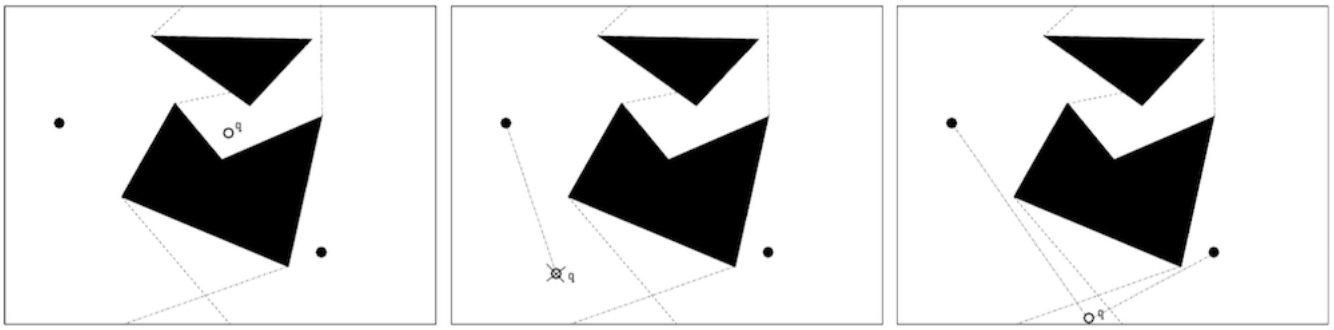

Figure 4: node added as a guard; node rejected; connection node merging two connected components

chosen congurations are checked for collision to generate samples in CS_free; when a free sample is found, it is added to the roadmap either if it does not 'see' any another node of the current roadmap (i.e. it is a new guard) or if it is seen by at least two nodes belonging to two distinct connected components of the roadmap (i.e. it is a connection node). The end of the roadmap's construction is controlled by a termination condition related to the volume of free space currently covered by the roadmap.

2.3.1 Guard and connection node

When a free sample is found, it is added to the roadmap in two cases:

- If it does not "see" another node already in R . This will be a new guard.
- If it is seen at leas by two nodes belonging to two distint connected components of will be a connection node.

2.4 Algorithm

The algorithm, called Visib-PRM, iteratively processes two sets of nodes: Guard and Connection. The nodes of Guard belonging to a same connected component (i.e. connected by nodes of Connection) are gathered in subsets G_i.

At each elementary iteration, the algorithm randomly selects a collision-free configuration q. The main loop processes all the current components G_i of Guard. The algorithm loops over the nodes g in G_i , until it finds a node visible from q. The first time the algorithm succeeds in finding such a visible node g, it memorizes both g and its component G_i, and switches to the next component $G_i + 1$. When q 'sees' another guard g 0 in another component G_j , the algorithm adds q to the Connection set and the component G_j is merged with the memorized G_i. If q is not visible from any component, it is added to the Guard set. The main loop fails to create a new node when q is visible from only one component; in that case q is rejected. Parameter ntry is the number of failures before

$Guard \leftarrow \emptyset$; $Connection \leftarrow \emptyset$; $ntry \leftarrow 0$
while $(ntry < M)$
  Select a random free configuration q
  $g_{vis} \leftarrow \emptyset$; $G_{vis} \leftarrow \emptyset$
  for all $G_i \in Guard$ do
    $found \leftarrow false$
    for all $g \in G_i$ do
      if $q \in Vis(g)$ then
        $found \leftarrow true$
        if $g_{vis} = \emptyset$ then $g_{vis} \leftarrow g$; $G_{vis} \leftarrow G_i$
        else $Connection \leftarrow Connection \cup \{q\}$; Create (g, q) and (q, g_{vis}); Merge G_{vis} and G_i
    until $found = true$
    if $g_{vis} = \emptyset$ then $Guard \leftarrow Guard \cup \{q\}$; $ntry \leftarrow 0$
    else $ntry \leftarrow ntry + 1$
end

Figure 5: Visibility PSM Algorithm

the insertion of a new guard node. 1/ntry gives an estimation of the volume not yet covered by visibility domains. It estimates the fraction between the non-covered volume and the total volume of CS_free. This is a critical parameter which controls the end of the algorithm. Hence, the algorithm stops when ntry becomes greater than a user set value M, which means that the volume of the free space covered by visibility domains becomes probably greater than (1-1/M).

2.5 Failure

The random generation of the guards may produce in some cases guards that will be difficult to connect. This effect is illustrated in Fig. 6 where two guards have been generated near the boundary of the black triangular obstacle. They fully cover CS_free, however the intersection of both visibility domains is 'unfortunately' small. The only way to complete the roadmap is to pick a connection node in the small triangle. Then the algorithm will fail if the parameter M is not sufficiently high.Nevertheless this case is only a side-effect of the algorithm. Indeed, in this example, the probability to select the first two guards with a small intersection domain is very low. Moreover, this undesirable

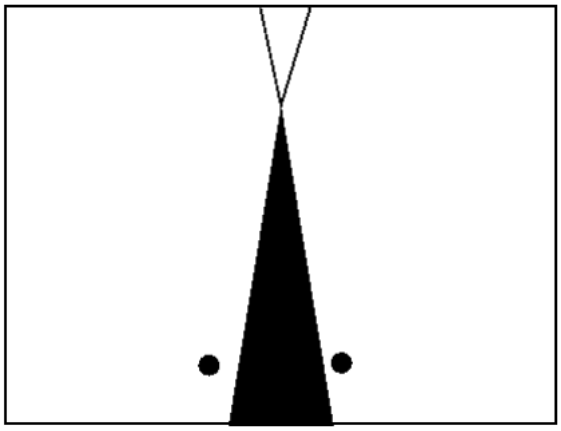

Figure 6: Visibility PSM Failure

effect was never observed in practice in all the examples we experimented on with the algorithm.