RRT, Value Iteration and Policy Iteration

Siwei Guo[1]

I. INTRODUCTION

Continuous from last week, other than search based path planning algorithm, another major category is sampling based path planning algorithm. Sampling based path planning has a better performance than search based path planning when the dimension of the space is very large. Since for search based path planning, we don't need to keep the whole discretized configuration space, it will be computationally faster and use less memory. In this project, we will mostly focus on RRT.

We will also discuss (optimal control problem) infinite horizon problems and it's approach.

II. PROBLEM FORMULATION

A. Static Map and Goal

Consider the case that there is a map with a known configuration and goal. We would like to find the optimal path that connects the start with goal without interfere with any obstacles. The problem is the same as shortest path problem. Given a graph G with vertices $x \in \mathcal{X}$, edges and corresponding cost, C_{ij} cost from vertices x_i to node x_j. The objective is to find the shortest path $Q = \{s, x_1, x_2 ..., \tau\} \, with \; x_t \in X$ from a start vertices s to an end vertices τ that minimize its corresponding cost is $J^Q = \sum_{t=1}^{q-1} c_{t,t+1}$. Assume that vertices are not connected to themselves and costs are positive values.

Problem Formulation: Find Q^* such that $Q = \underset{Q \in \mathbb{Q}_{\sim,\tau}}{argmin} J^Q$ with $s \in \mathcal{X}$ and $\tau \in \mathcal{X}$.

B. Optimal Control Problem

Given a pendulum system with known motion model and cost function, we want to find the optimal control so that we can stabilize the inverted pendulum. The state of the system, $\mathbf{x}$ consists of angle of the pendulum, θ, and angular velocity, ω. The continuous-time dynamics is $\mathbf{x}' = d\mathbf{x} + \mathbf{x}$. Define

$$d\mathbf{x} = f(\mathbf{x}, u) * dt + \sigma * dw \tag{1}$$

, where $f(\mathbf{x}, u) = \begin{bmatrix} \omega \\ asin(\theta) - b\omega + u \end{bmatrix}$, where $a > 0$ summarizes the eects of gravity, mass and length of the pendulum, $b > 0$ represents damping and friction, u is the control input and σ is Gaussian motion noise (Brownian motion). Then define the motion model as

$$p_f(\cdot \mid \mathbf{x}_t, \pi_t) = \mathcal{N}(\mathbf{x} + f(\mathbf{x}, u) * dt, \sigma^T \sigma * dt) \tag{2}$$

The stage cost is

$$g(\mathbf{x}, u) = 1 - exp(kcos(\theta) - k) + \frac{r}{2} u^2 \tag{3}$$

, where $k > 0$ determines the shape of the cost and r ¿ 0 scales the control cost. Given these information fo the system, we would like to find a policy that minimize the stage cost for each $\mathbf{x}$.

Problem Formulation: Given motion model $f(\mathbf{x}, u)$ and cost function $g(\mathbf{x}, u)$, find the optimal policy $\pi(\mathbf{x})$ such that the cost of policy π, $J_0^\pi(\mathbf{x}_0) = \mathbb{E}_{x_{1:T}} \left[\sum g(\mathbf{x}_t, \pi_t(\mathbf{x}_t) \mid \mathbf{x}_0\right]$ such that $x_{t+1} \sim p_f(\cdot \mid \mathbf{x}_t, \pi_t)$, $\mathbf{x}_t \in \mathcal{X}$, $\pi_t(x_t) \in \mathcal{U}(\mathbf{x}_t)$.

III. TECHNIAL APPROACH

A. Static Map and Goal

This week, for the motion planning problem, we will use sample based motion planning to solve this problem. The most common sampling based motion plan algorithm is RRT. RRT is a tree that constructed from random sampled nodes in the space. The tree grows until there is a path that connect start and goal. Simple RRT doesn't guarantee to find a path even the number of nodes goes to infinite. The benefit of RRT is that we don't need to store the whole configuration space and we can find a reasonable path in a short amount of computation time even though the path is not optimal. RRT is useful for planning in high deminsonal configuration space, such as plan for robot arm, soft robot. Start from goal, first randomly sample a point x_{rand} in the space. Find the nearest point in the existing nodes x_{near}. Then make a connection between x_{rand} and x_{near} and extend x_{near} a small distance, σ, towards x_{rand} if there is no obstacles. If there are obstacles in the middle, we pick the point that is at the edge of the obstacles to be x_{near}. Repeat this step until the goal is connected. To ensure that the goal can be sampled, for every 100 iteration, set x_{goal} as x_{rand}. If there exist a path between goal and one of the node which is collision free, we can include the goal to the tree and stop planning.

I have also add a heuristic to the sampling process. Instead of sampling from a uniform distribution, x_{rand} is sampled through a gaussian distribution which the mean is the goal, $\sim \mathcal{N}(x_{goal}, \sigma)$. σ determines how spread x_{rand} should be. For the case that there is not many obstacles, adding a heuristic can help find a path in less iterations. However, for the case that there is many obstacles in the map, adding heuristic might cause robot to stuck at a corner.

Algorithm 1 RRT

```
$\tau$.init($x_s$)
for $i = 1...N$ do
    Sample $x_{rand}$
    Extend($\tau$, $x_r and$)
end for
```

Algorithm 2 Extend(τ, $x_r and$)

```
$x_{near} \leftarrow$ NearestNeighbor($\tau$, $x_{rand}$)
$x_{new} \leftarrow$ Steer($x_{near}, x_{rand}$)
if ObsacleFree($x_{near}, x_{rand}$), then then
    $\tau$.add vertex($x_{new}$)
    $\tau$.add edge($x_{new}$, $x_{near}$)
    if $x_{new} = x_{rand}$ then return Reached
    end if
end if
```

B. Optimal Control

First, we will discretize the state space and control space. The pendulum degree is in the range of $[-\pi, \pi]$ and we split into n_1 grids, same for control $u \in [-u_{min}, u_{max}]$ with discretization of n_u grids and angular velocity $\omega \in [-u_{min}, u_{max}]$ with discretization of n_2. Becasue the motion model contains Gaussian motion noise and discretization, for $\mathbf{x}'$ we will pick the points that is within one standard deviation of $\mathcal{N}(\bar{\mathbf{x}}, \sigma^T\sigma * dt)$. For this setup, we need to solve a inifinite-horizon problem. As terminal time T goes to infinity, the original problem, $min \ J_0^\pi(\mathbf{x}_0) = \mathbb{E}_{x_{1:T}}\left[\sum g(\mathbf{x}_t, \pi_t(\mathbf{x}_t) \mid \mathbf{x}_0\right]$ becomes a time invariant cost-to-go problem. Since there is no terminal state, but future reward should weighted less than current rewards. Then we can formulate a dicounted infinite horizon problem which is stationary,

$$J^*(x) = \min_{u\in\mathcal{U}} \quad g(\mathbf{x}, u) + \gamma\mathbb{E}_{x'\sim p_f(\cdot|\mathbf{x},u)}\left[J^*(\mathbf{x}')\right], \forall \mathbf{x} \in \mathcal{X} \tag{4}$$

This is also the Bellman Equation

$$J^*(i) = \min_{u\in\mathcal{U}} \quad g(i, u) + \gamma \sum \left[P_{ij}^u J^*(j)\right], \forall i \in \mathcal{X} \tag{5}$$

This could be solved using policy iteration or value iteration if the policy statisfies follow two assumptions

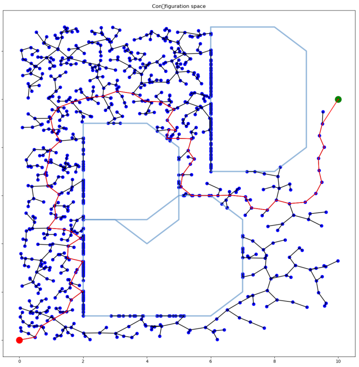
- Proper Stationary Policy: a policy π for which there exists int(m) that $P(x_m = 0 \mid x_0 = i) > 0 \forall i \in \mathcal{X}$
- Proper Policy Assumption: there should exists at least one proper policy π

IV. CONCLUSIONS

A. Static Map and Goal

With RRT and enough time, we are able to find a path for both map. For triangle map, compare to the optimal path, RRT algorithm isn't able to sample points in a very narrow space. The optimal path requires to go through a very narrow corridor. For RRT, it is nearly impossible. Therefore, it didn't find the optimal path. However, RRT finds a suboptimal path with only 2246 sampled points. Since we add a heuristic to the sampling process, more points were sampled at upper right corner which causes the path to go up first instead of go though the bottom which seems to be a better path.

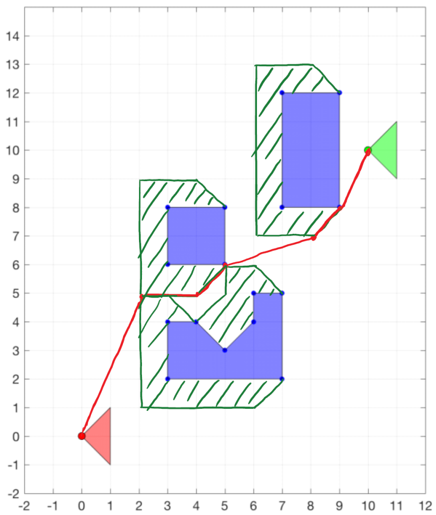
For maze map, the algorithm sampled 121852 points before finding the path which tooks about 4hs. At first, the heuristic for sampling is relatively large. It took even larger amount of points to find a path. Since in the maze case, we don't want the algorithm to be greedy. Instead, we want it find the small exit on each wall. However, even after I remove the heuristic, it still takes a long time to find a path. From fig3 we can see that, sampled points have alomst take over all the space inside the second outermost wall. I also didn't let goal to be x_{rand} in every 100 steps to save time. As long as the robot can escape the maz, the path for the last distance from the exit of maze to the goal can be easily found using heuristic. I chose ϵ to be 0.5. For larger ϵ, it took less time to find the path. However, ϵ cannot be too large, otherwise the x_{near} will frequently collide with obstacles.



(a) Triangle

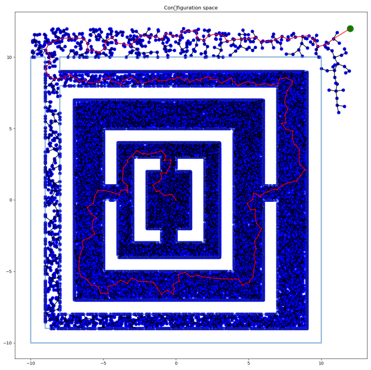
Fig. 1: RRT: Trangle Map. 2246 sampled points. 66 nodes from start to goal

B. Optimal Control



(a) Triangle

Fig. 2: Trangle Map. Optimal Path



(a) Triangle

Fig. 3: RRT: Maze Map. 121852 sampled points. 240 nodes from start to goal