Advanced Robotics

October 2024

1 Introduction

This project focuses on developing an effective motion planning strategy using kinodynamic Rapidly-exploring Random Trees (RRT) in environments cluttered with obstacles. The main challenge is to generate feasible trajectories that respect the system's dynamic constraints while avoiding collisions with obstacles. By leveraging kinodynamic RRT, we aim to create paths that account not only for geometric feasibility but also for the physical limitations of the robot, ensuring smooth, safe navigation through complex environments.

2 Problem Description

Consider a corridor-like environment with n obstacles as in Figure 1. For the first part of the project, we set $n = 1$. A point robot R is located at the origin A and is tasked to navigate this environment to reach point B without colliding with any obstacles. The robot's configuration [1] (location) in the environment is denoted by q and it's velocity is denoted by $\dot{q}$, where both q and $\dot{q}$ are in $\mathbb{R}^2$. For example, the robot at the origin has $q_A = (0, 0)$. The state space of the robot is denoted by $\mathbb{X}$. A robot state $x \in \mathbb{X}$ is defined as $x = \{q, \dot{q}\}$. This state space is divided into collision-free $\mathbb{X}_f$ and obstacle $\mathbb{X}_o$ subsets.

For robot motion, controls $u \in \mathbb{U}$, where $u = (u_x, u_y)$, are applied on the robot in the x-y direction (w.r.t to itself). $\mathbb{U}$ is the control space of the robot.

We need an algorithm that builds a collision-free **motion** plan which takes the point robot R from A to B. A **plan** $p(T)$ of duration T is a sequence of constant controls $\{u_0, \ldots, u_{T-\Delta t}\}$, where each u_t is executed for time Δt. When $p(T)$ is executed it produces a **trajectory** τ. For example, if a plan $p(T)$ is executed from x_0, the trajectory is a sequence of states $\tau(x_0, p(T)) = \{x_0, \cdots, x_T\}$.

In this project, you need to build a motion planning algorithm called Kinodynamic RRT. This is a tree sampling-based algorithm, where we build a tree

[1] This is a symmetric point robot and therefore we do not consider it's orientation (w.r.t the origin) in q.

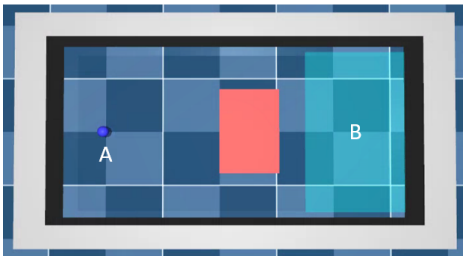

Figure 1: The environment in the Mujoco simulator, where A is the start location and B is the goal region. The blue ball is the point robot R. The red box is the obstacle that we need to avoid.

between A and B by sampling configurations and controls. More details are given in the algorithm below. Refer here for more details on Kinodynamic RRT. Once the tree connects the start and the goal, a **plan** can be extracted from the tree between the start and goal that takes the least amount of time. This algorithm will have 2 termination conditions. First, a boolean goal check that returns true if there is path in tree connecting the start state and the goal state. In other words, if the newly added node x_e is in the goal region. Second, if the planning time exceeds a $T_{\max}$ threshold.

Algorithm 1: Kinodynamic-RRT($\mathbb{X}, \mathbb{U}, x_0, X_G$)

$\mathrm{T} \leftarrow \{x_0\}$;
while *termination condition is not met* **do**
  $x_{\text{rand}} \leftarrow$ `SAMPLE-CONFIGURATION`();
  $x_{\text{near}} \leftarrow$ `NEAREST-NEIGHBOR`$(\mathrm{T}, x_{\text{rand}})$;
  $u_{\text{e}} \leftarrow$ `SAMPLE-CONTROL`();
  $x_{\text{e}} \leftarrow$ `SIMULATE`$(x_{\text{near}}, u_{\text{e}})$;
  if $(x_{near} \rightarrow x_e) \in \mathbb{X}_f$ **then**
    `EXTEND-TREE`$(\mathrm{T}, x_{\text{near}} \rightarrow x_{\text{e}})$;
    if *GOAL-CHECK*(x_e) **then**
      return

For the first part of the project, you will use the MuJoCo simulator (coding language: Python) to create the environment and build motion plans. The XML file attached with the project will help you get started with the environment in mujoco.

1. Setup with the simulator and replicate the environment as shown in Figure 1.
2. Implement the motion planning algorithm.
3. Visualize trees for 5 different trials (5 different seeds) of the algorithm.

4. Assume $T_{\max} = 30$ seconds, Run the 30 trials with different seeds and report the success rates. To measure success, return a motion plan from the algorithm and execute it. Deem it as a failure if we do not find a plan.

5. Repeat the above for $T_{\max} = \{5, 10, 20\}$ and report success rates for each. Also report any observations you have.

References

[1] S. M. LaValle and J. J. Kuffner, *Randomized kinodynamic planning.*