Nonholonomic Planning Assignment 3 Report **Sean Ye, Patrick Grady**
CS 8803 - Adaptive Control and Reinforcement Learning
Prof. Byron Boots Date: 04/15/2019

1 Overview

For this assignment, we chose to use an RRT planner to generate feasible paths for the car to follow. The planner is able to produce paths for all maps given bridges are open but does not converge in a reasonable time to produce plans through the many tight turns on maps 2 and 3.

2 Nonholonomic RRT Planner

The basic RRT algorithm implemented is shown below (1). This algorithm was first described by LaValle in 1998. [**?**]

Algorithm 1 RRT Algorithm

Initialize Tree T
while x_{goal} not reached **do**
  $x_{rand} < -RANDOM\_STATE()$
  $x_{near} < -NEAREST\_NEIGHBOR(x_{rand}, T)$
  $u_{new} < -SELECT\_CONTROL(x_{rand}, x_{near})$
  $x_{new} < -STEER(u_{new}, x_{near})$
  $T.add\_vertex(x_{new})$
  $T.add\_edge(x_{near}, x_{new}, u_{new})$
end while

Several modifications were made to this algorithm to perform better for this project. Each modification with respect to the function is listed here:

2.1 RANDOM_STATE()

Rather than a completely random point chosen for the RRT, we decided to choose the goal point 15 percent of the time. This heuristically biases the tree to keep on trying to explore regions to get to the goal.

3 Results

4 Conclusion and Future Work

The final results of the training are shown in Figure 1 below. I ran 7 episodes which resulted in an agent capable of scoring a million points on average. While still a very good performance, it does not achieve the same scores as Thiery and Scherrer, with their training curve reproduced below in Figure 2 (where the DU line is the same feature set that I used). The time it took to train the agent took about 10 hours.

Attachments

Code can be run with run_sim.m, first displays progress of RRT planner and then runs the path