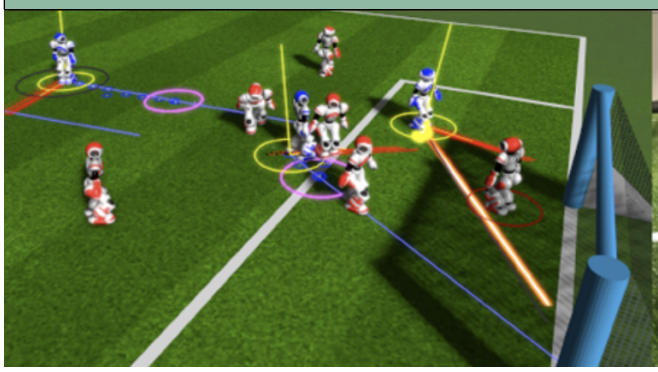
Fall 2024 - CSC398 Autonomous Robots - Assignment 6



Due date: 11/14/2024, 23:59pm. This assignment is worth 20 points.

This assignment is worth 20 points and it is due on or before November 14, 2024. The goal of this exercise is to implement the Rapidly-Exploring Random Tree (RRT) algorithm for path planning in a 2D environment. You will write a Python/C++ program that generates a collision-free path between a start and goal position. Optionally, you will implement a solution using real-time sensor data from a 2D Lidar (`/hsrb/base_scan`).

1. We made some small modifications to the simulation world. `git pull` (from your catkin workspace) to update.
2. (3 points) Explain the following concepts in your own words: What is path planning in robotics? Describe the basic steps of the RRT algorithm. What are the benefits and limitations of using RRT and RRT* for path planning?
3. `cd` into the `src/` directory off your catkin workspace. Clone the new assignment: `git clone` https://classroom.github.com/a/k7cdAta9.You will find a starter code there. Make sure that the scripts are executable.
4. You will use the provided simulation by running `./isaac_hsr_sim_rrt_navigation.sh`.
5. (2 points) You will need to create the following markers:
   - a start point marker (named as (`x_start, y_start`)) at initial 0,0 position.
   - a goal point (marked as (`x_goal, y_goal`)) at the provided position.
   - Your goal points are: (1, 0), (6, 4), (3, 3), (9, 3)
6. Your program should include the following steps:
   - (3 points) Random Point Generation: Randomly generate points within the workspace.
   - (1 point) Nearest Node Selection: Identify the nearest node in the current tree to the newly generated point.
   - (1 point) Tree Expansion: Extend the tree from the nearest node toward the randomly generated point.
   - (2 points) Collision Detection: Ensure the new point does not collide with obstacles. Implement a function that checks whether a path between two points collides with any obstacles. This will require basic geometry to check for intersections with the obstacles in your environment. (You can use map data, inflated map, laser scan, depth/pcl and other information).
   - (1 point) Goal Bias: Introduce a small probability of selecting the goal point directly to speed up convergence.
   - (2 points) Curve smoothing: Smooth the final path for a smooth movement of the robot.
7. (2 points) Use markers to visualize the following: (a) the random nodes generated by RRT, (b) the tree structure as it expands toward the goal, (c) the final path found by the algorithm (if one exists), (d) the robot's traveled path.
8. (2 points) Use `hsrb/cmd_vel` topic to publish velocity commands and move the robot along the generated path. Ensure that the robot can avoid detected obstacles and successfully reaches the goal.
9. (1 point) Create a launch file that runs your path planning script and visualization.
10. Bonus tasks (10 points)
    - (7 points) Dynamic obstacles. Use the real-time sensor data (ex. LaserScan) to avoid dynamic obstacles such as people or random objects.
    - (3 points) Create a solution for approaching an unreachable location. Get as close as possible to that location (ex. you want to get to the position of an object on a table - how would you approach its location without colliding with that table?)

Submission:

1. Add and commit modifications to the provided package to github classroom.
2. Add a .txt or .pdf document containing the following:
    - Answers to questions in question 1.
    - Explanation of your RRT algorithm and how it was implemented.
    - Challenges encountered during the implementation.
    - Screenshots of RVIZ with the generated path for each goal point.
    - Any modifications or optimizations you made to improve the performance of your implementation.