ME5413: Autonomous Mobile Robotics

Homework 3: Planning

AY2023/24-Sem 2

Due: 23:59 Sunday, 31 March 2024

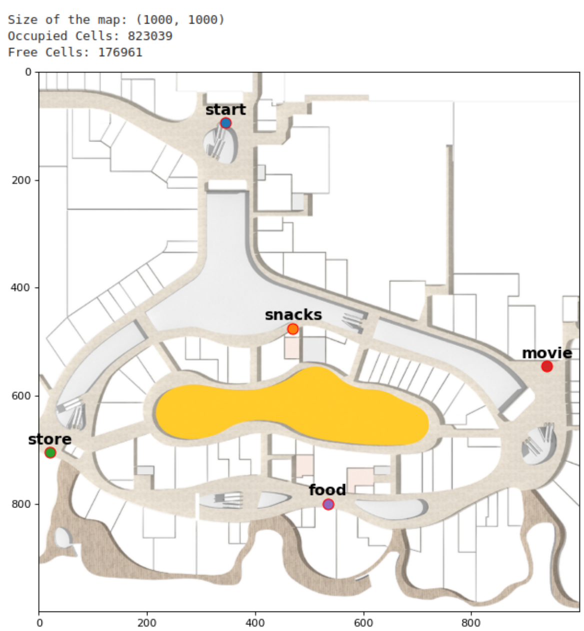
(Late submissions will be penalized)

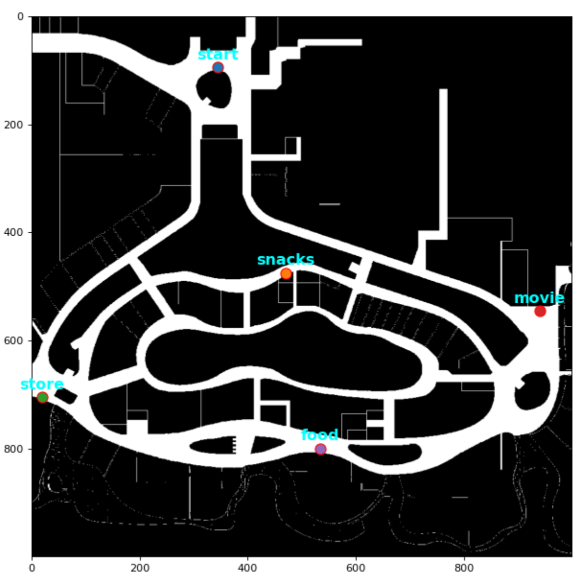
Introduction

The aim of this homework is to let you gain experience with several basic planning algorithms and their applications. After implementing and testing those algorithms, you should be able to appreciate the real-world challenges in planning and the pros and cons of each algorithm.

Task 1: Global Planning

For both Task 1&2 of your planning homework, you will be using a map of the VivoCity level 2, as shown below:





The left picture is the original floor plan, while the right one is a grayscale version to be used by your planning algorithms. The whole map has a size of 1000 x 1000 pixels (grid cells), with each cell representing a 0.2m x 0.2m square area. Each grid cell belongs to one of two possible states: Free (value `**255**`) or Occupied (value `**0**`). You yourself, as a human, are expected to have a circular footprint of no less than **0.3m** radius.

There are five given key locations on the map you wish to visit: `**start**`, `**snacks**`, `**store**`, `**movie**`, and `**food**`. Your mission is to plan a path/trajectory between each pair of a start and end points, display your plan on the map and count the total travel distance of your plan in meters. Finally, use the distances you calculated, find the most efficient route for yourself, that is currently standing at the level 2 escalator (the start), to visit all the four locations and return to the start.

In this task, you are required to implement an A* planning algorithm.

In your algorithm, you should:

- Use 8-connected neighbors, with cost values 0.2m or 0.282m
- Compute the total distance traveled to reach the goal
- You may pick 1-2 ideas from below and try:
    - Try different heuristic functions (also design your own)
    - Try switch the start and goal positions
    - Try search from both ends
    - Degenerate the A* algorithm to Dijkstra's Algorithm
    - Degenerate the A* algorithm to Greedy Best First Search Algorithm
    - Implement the Hybrid-A* algorithm
    - Implement the RRT family

The output of your algorithm should include:

- Your planned path
- The total travelled distance in meters
- All the cells visited by your algorithm
- The total run time of your algorithm
- Other critical performance indicators

In your report, you should:

- Document the implementation details of your algorithm, and the improvements you made to the original algorithm (if any)
- Compare the difference between algorithms/different settings of the same algorithm
- Describe the difficulties you encountered, and your solutions
- Identify the shortcomings of the methods you used, and suggest improvements
- Summarize the shortest distances between each pair of locations in a table:

From / To	start	snacks	store	movie	food
start	0.0				
snacks		0.0			
store			0.0		

movie				0.0	
food					0.0

Task 2: The "Travelling Shopper" Problem

You are now at the VivoCity level 2 escalator (the start), and you wish to visit all the four locations and then return to the escalator. Based on the distance table you obtained in **Task 1**, find the optimal route for you to visit all stores and come back to the start location.

1. Describe how you want to model this problem and propose a few methods you think that can solve this problem.
2. Apply at least two of the methods on this problem, and document your implementation details
3. Compare the solutions computed by the two methods, state your observations/findings
4. Show the final shortest route on the map, and the total distance.

Task 3 (Bonus): Path Tracking

Control your robot to follow the given figure 8 track using any algorithm you like

- ☐ Replace our template code with your own algorithms in the:
  - `src/me5413_world/include/me5413_world/path_tracker_node.hpp`
  - `src/me5413_world/src/path_tracker_node.cpp`
  - `src/me5413_world/cfg/path_tracker.cfg`
- ☐ Test your algorithms on the track & Report your tracking accuracy using the given metrics:
  - RMSE position
  - RMSE heading
  - RMSE speed
- ☐ Some suggestions:
  - You may try algorithms like Pure Persuit, LQR, MPC, etc.
  - You may vary the shape & size of the track by changing the `track_A_axis` and `track_B_axis` lengths (in meters), and report the changes in your algorithms' performance.

Submitting your completed Homework Assignment:

Code: For **Task 1&2**, all code should be implemented and submitted in the Jupyter Notebook with all your outputs displayed. For **Task 3**, all code should be implemented in your repository and submitted as a GitHub repository with link provided. Do note that you should practice good code styles in your own code, including proper naming conventions, informative documentations, etc. (please refer to the Google Python Style Guide, Google C++ Style Guide, C++ Core Guidelines, ROS C++ Style Guide)

Report: You are expected to summarize your observations, assumptions, and your own implementation details in a 5-page report (there is no page limit for appendices if you wish to attach more of your results, but limited to equations/algorithms/tables/figures only, no text paragraphs).

Submission: Generate a non-password-protected zip file of this folder and upload it to CANVAS – under Assignment 3. We will use the latest version, regardless of uploaded by whom. Name of Zipfile: \`HomeworkGroupNumber_Homework3.zip\` (e.g., \`43_Homework3.zip\` - for group 43)

1. Report details:
   a. Homework Group number
   b. Matric numbers of group members (e.g. number starting with A0...)
2. Code and results:
   - Task 1&2: in Jupyter Notebook (\`/src/Task_1/homework3.ipynb\`)
   - Task 3: a link to your GitHub repository
3. Evaluation of tasks will be based on:
   a. Results Performance/Accuracy
   b. Technical Explanations
   c. Effort
   d. Code Executability & Style
   e. Report