Project 1: Basic Autonomous Ground Robot

Contents

Submission

All codes should be zipped into `.zip`. Please zip up the workspace folder.

Submission for **demonstration code** and **presentation** to LumiNUS is due 15 minutes before your demonstration for running, on 4 March, (W7 Wed) sometime between 1300-1700h, depending on your presentation.

Further cosmetic changes (to make it more readable and organised for report) to the **code** can be made and submitted to LumiNUS by 2359h, 4 March (W7 Wed)

Report should be submitted by 2359h, 13 March (W8 Fri). Penalties for late submission of report will be -2 marks per three hours. Late submission in the first three hours will be -2 marks. The next three hours, the total penalty will be -4 marks.

Queries

Email Yan Kai (lai.yankai@u.nus.edu) if you have are unsure of anything, but please cc Prof. Prahlad (elepv@nus.edu.sg).

Overview

Run a robot through an unstructured, unknown obstacle course through 4 areas – 1 start, 1 end and 2 intermediate zones. The path is approximately 30 metres long, with bends in a 20x20m map.

Practice maps would be given. However, the demonstration map would not be known until the demonstration day. The demonstration map will be the same for everyone. The map would not be distributed to protect against academic dishonesty.

1. Demonstration (10):

- Code must be able to be run on any lab computer upon unzipping and sourcing.
- Code must be launched from a `.sh` file. It must include a `.launch` file that can accept parameters on the intermediate, start and end areas, as well as the world. The world should be launched with **Turtlebot3 Burger** in **Gazebo**.
- Max 5 minutes

Marks		Requirement
1,2 pax	3 pax	
1	NA	Reach area A
1	NA	Reach area B
1	NA	Reach area C
1	4	Pass through all areas
1	1	No collision
1	0.5	Can complete in 300s, all areas reached
1	1.5	Can complete in 200s, all areas reached
1	1	Every iteration of main node (map updating + trajectory generation) and motion node (motion control to via points) takes less than 0.2s
1	1	Moves continuously (without stopping)
1	1	Show the occupancy grid (free, unknown, occupied) as an RGB image with the trajectory, robot and via points (goal and start included)

2. Presentation (3):

- Max 5 minutes
- Come 15 minutes early to setup presentation with projector before demonstration

Marks		Requirement
1,2 pax	3 pax	
0.5	0.5	Finish within 5 minutes
0.5	0.5	Words can be seen 2 metres away from projector (contrasting, not too small)
1	1	Concise explanations of all 6 categories under ***Coding*** below
1	1	Some evidence of efficient implementation, algorithmically or programmatically

3. Coding (10, Bonus [*] 3)

- The requirements will be marked during demonstration and presentation. They will be checked for both in the first version of your code. So make sure you present all of those things. You can provide more details in the report.

Inverse Sensor Model (1)

Marks		Requirement
1,2 pax	3 pax	
1	1	Implement the Inverse Sensor Model

Motion model (1, 1*)

1	1*	Implement Odometry Motion Model with wheel encoders
1	1*	Fuse simply with other sensor information for Odometry Motion Model

Map Updating (2)

1	0.5	Implement Binary Log Odds
1	1.5	Implement inflation zones ***or*** potential fields to provide guarantees against collisions

LOS (1, 0.5*)

1 + 0.5*	1 + 0.5*	Implement General Line Algorithm

Or

1	0.5	Implement Bresenham Line Algorithm

Global Planning (3, 1*)

1	0.5	Implement A*
1.5	1	Post process to get at least a near-optimal any-angle path, represented by a series of turning points

Or

1.5 + 1*	1.5 + 1*	Implement Jump Point Search
1.5	1.5	Post process to get at least a near-optimal any-angle path, represented by a series of turning points

Or

1.5 + 1*	1.5 + 1*	Implement either ANYA or Theta* (any-angle) path planner
1.5	1.5	Post process to get a series of turning points

Or

1.5 + 1*	1.5 + 1*	Implement any other global path planner that is not {DFS, Floodfill, GBFS, Dijkstra, Potential Field, Voronoi}, and that is faster than A* during online planning
1.5	1.5	Post process to get at least a near-optimal any-angle path series of turning points

Local Planning & Motion Control (2, 0.5*)

0.5	0	Implement motion control to move about global turning points without smoothing
0.5*	0	Move continuously from start to end

Or

1	1	Implement trajectory generation (path smoothing) using splines between global turning points. Path must be collision free.
1	1	Generate intermediate via points on the spline trajectory
0.5*	0.5*	Move continuously from start to end

Or

1	1	Project global turning points to local space
1	1	Implement Potential Field to guide the robot while avoiding obstacles
0.5*	0.5*	Move continuously from start to end

4. Report (7)

- Softcopy submission of the report onto LumiNUS.
- Try (optionally) to limit the report to at most 20 pages. Quality and quantity. But concise, don't write garbage.
- Use narrow margins and single spacing, font-size 11.
- Don't fill the page with one or two diagrams.
- ***If the corresponding code does not exist and is not demonstrated in the presentation, the marks are voided***.

Marks		**Requirement.** Clearly and concisely,
1,2 pax	**3 pax**	
1	1	Elaborate on the motion model that you used, qualitatively (what you put in, how it transforms that into something useful, any advantages over other models)
1	1	Explain why inflation zones are used and how you implement it efficiently ***or*** your guarantees against collisions if you want to use potential fields to implement obstacle avoidance
1	1	Elaborate on the line algorithm, qualitatively (your input, what it does, what it returns, advantages / disadvantages).
1	1	Describe your global path planning algorithm
1	1	Elaborate on the advantages and disadvantages of your path planning algorithm
1	1	Describe your path smoothing and post processing algorithm, if any. Also, list the advantages and disadvantages.
1	1	Describe how your motion control works and how it follows the generated trajectory, if any.
1	1	Evidence of consistent efforts to make implementations more efficient, either algorithmically (re-ordering steps, identifying problems with lecture, online or existing algorithms) or programmatically (masks, type casting, etc.).