CDA 4621 / CDA 6626
Grad Total: 110 points + Extra Credit
Undergrad Total: 100 points + Extra Credit
Spring 2024
Lab 5
Mapping & Path Planning
Total: 100 points

Due Date: 4-18-2024 by 11:59pm

The assignment is organized according to the following sections: (A) Objective, (B) Requirements, (C) Task Description, (D) Task Evaluation, and (E) Lab Submission.

A. Objective

This lab will teach you about robot mapping and path planning. You will use: (1) Global Reference Frame and Grid Cell Numbering, (2) World Representation, and (3) Robot State.

A.1 Global Reference Frame and Grid Cell Numbering

The global reference frame and grid cell numbering scheme is shown in Figure 1. The positive y-axis points North while the negative y-axis points South. The positive x-axis points East while the negative x-axis points West. Origin (0,0) is in the middle of the arena. The arena consists of 16 grid cells, numbered 1-16, each measuring 10 x 10 inches.

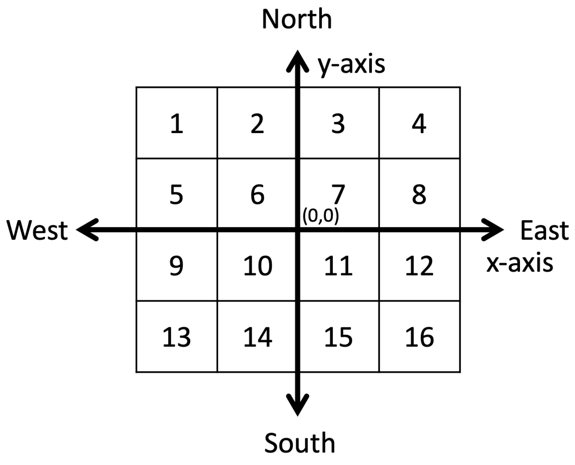
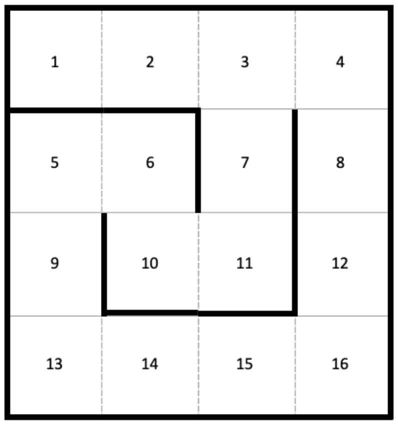


Figure 1. Global reference frame and grid cell numbering scheme.

A.2 World Representation

You will need to implement a strategy for representing the wall configuration of a world. As a conceptual guide, you might consider a 16x4 matrix format, as exemplified in Figure 2, where the left side portrays the actual world while the right side provides the corresponding matrix representation, with walls specified according to the West-North-East-South (WNES) orientation, where "W" denotes the presence of a wall, and "O" indicates the absence of one.



Cell	Walls (WNES)
1	WWOW
2	OWOW
3	OWOO
4	OWWO
5	WWOO
6	OWWO
7	WOWO
8	WOWO
9	WOWO
10	WOOW
11	OOWW
12	WOWO
13	WOOW
14	OWOW
15	OWOW
16	OOWW

Figure 2. (Left) Sample wall configuration and corresponding wall representation (Right) using a 16x4 matrix configuration describing the 16 cells, and 4 walls organized as West-North-East-South (WNES), where "W" represents "Wall", and "O" represents "No Wall".

A.3 Pose (State) Printing

In all tasks, you need to print the following information (once per cell). Be sure to include the print statements in your videos:

(a) Visited cells. Use as an example the table below to display already visited cells, where an "X" indicates a visited cell, and "." indicates a yet-to-be-visited cell. The top left corner corresponds to cell number "1" as described in Figure 1.

```
....
XXX.
XXX.
X...
```

(b) State pose. State pose is given by *s*=(*x, y, n, θ*), where "*x,y*" represents the robot position in global coordinates, "*n*" represents the grid cell number, and "*θ*" represents the robot orientation with respect to the global frame of reference. You are not allowed to use the GPS. To track the robot's pose, you'll first need to access its starting position attributes, providing the initial reference point from which all motions and updates will be calculated. The starting state includes coordinates (*x, y*) an orientation angle (*θ*). These are the steps you need to follow at the very beginning of your code to obtain the robot starting attributes:

a. Access the **starting_position** attribute of the robot object. This attribute is designed to hold the initial position and orientation data.
b. Extract the x-coordinate from the **starting_position** which represents the robot's initial position along the horizontal axis.
c. Extract the y-coordinate from the **starting_position** which denotes the robot's initial position along the vertical axis.
d. Obtain the theta value from the **starting_position** which indicates the initial orientation angle of the robot.

Below is a snippet of code that demonstrates how to accomplish these steps:

```
    # Move robot to a random staring position listed in maze file
robot.move_to_start()

start_pos = robot.starting_position
x = start_pos.x
y = start_pos.y
theta = start_pos.theta
print(x,y,theta)
```

After obtaining these initial values, you will need to continuously update the robot's current position (x, y) and orientation (theta) as it moves. You can use sensors, like encoders and a compass, to calculate the robot's pose over time. This will need to be implemented in the main loop that runs continuously as the robot moves, fetching sensor data at regular intervals, and adjusting the x, y, and theta values accordingly. Consider there is no noise for the motion model.

B. Requirements

Programming: Python

Robot: Webots Rosbot (https://github.com/biorobaw/FAIRIS-Lite - see "README.md")

C. Task Description

The lab consists of the following tasks:

- Task 1: Wall Mapping
- Task 2: Path Planning
- Statistic Task (Required only Grads, Extra Credit Undergrads)
- Extra Credit Task: Occupancy Grid

Note: Do not modify the world, nor add any additional devices to the robot, or modify sensor or camera configurations from those already given.

C.1 Task 1: Wall Mapping

Mapping requires dynamically finding the complete internal wall configuration of the maze. Robot may follow any desired path and should end the task when all grid cells have been navigated, or after 3 minutes. The robot may (and should) travel through grid cells multiple times. Robot will start at grid cell "16" oriented "North". After navigation, print the entire map. Try to do the mapping when robot is parallel to walls. Sample world configurations with internal walls are shown in Figure 3. The actual world configuration will NOT be known in advance, it will be specified by the TA prior to program execution.

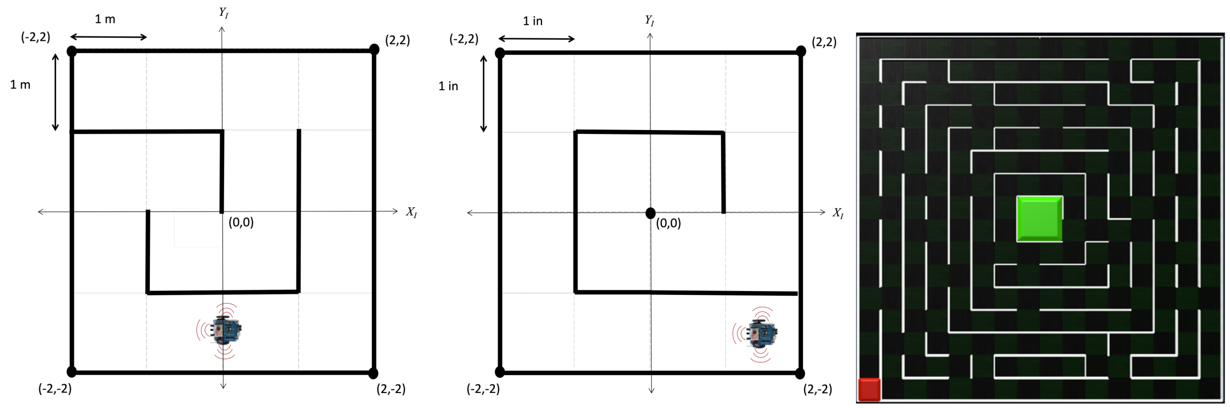


Figure 3. (Left and Middle) Sample small robot mazes for wall mapping task. (Right) Sample large world maze (micromouse) with start location in red and goal location in green.

C.2 Task 2: Path Planning

Path planning requires finding and navigating the shortest path between a start and goal location. For this task, the map and end goal location will be available to your program before execution (end goal orientation is not important). Note that you can use the maps from your Task1 solution if you save the data structure at the end of the mapping. For path planning you should use the wavefront planner with 4-point connectivity. The robot should end task when all grid cells have been navigated, or after 3 minutes. The robot may travel through grid cells multiple times. Prior to navigation, print the entire planned path. During navigation, the robot needs to print its current location. At the goal, the robot should stop and print "GOAL". Path planning task will be performed twice, with robot starting from two different initial states: (a) the robot knows its starting at grid cell "16" oriented "North", and (b) the robots starts in a random grid cell location and orientation (given by the TA only at the start of the run). Do not hardcode the path to be followed by the robot. Test with the small and large worlds as shown in Figure 3.

C.3 Statistic Task: Wall Mapping and Path Planning

Perform statistics on the following:

- Run wall following at 3 different speed combinations while trying to minimize overall travel time.
- Analyze and graph min, max and variance of velocities, orientations, and distances to walls.
- Discuss your results including any navigation errors.

C.4 Extra Credit Task: Occupancy Grid

Apply the occupancy grid algorithm to generate the maze maps shown in Figure 4. Subdivide each grid cell as desired, for example, with 5x5 subcells per grid cell the occupancy grid map would consist of a total 20x20 cells. Use the following values for occupancy: 0.3 for all empty cells at a distance that is less than the measured wall, 0.6 for the occupied cells at a distance that is equal or one cell behind the measured wall, and 0.5 for unknown (leave as previous values). The robot should end task when at least 90% of the all the subcells have been assigned new values, other than 0.5, or after 3 minutes. The robot may (and should) travel through grid cells multiple times. <u>The actual world configuration will NOT be known in advance</u>, it will be specified by the TA prior to program execution. You can extend existing python code as long as you reference its source and explain it.

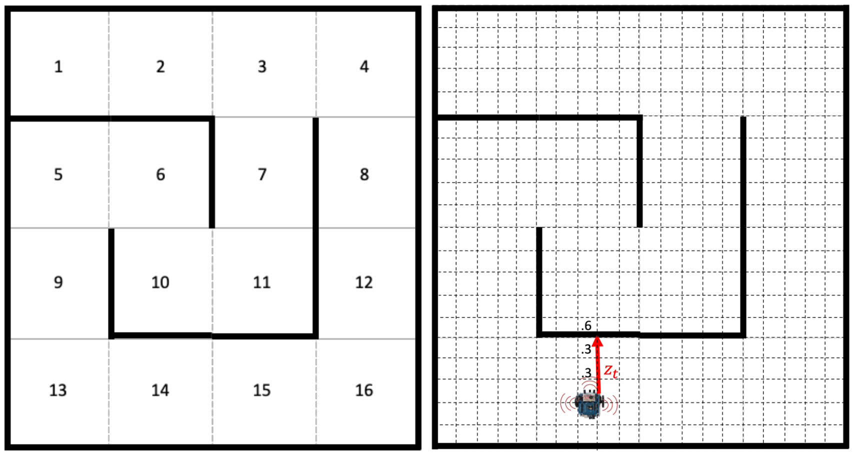


Figure 4. (Left) Sample 4x4 wall configuration and (Right) sample 20x20 occupancy grid map (5x5 subcells per original grid cell), corresponding to the 4x4 grid cell configuration.

D. Task Evaluation

Task evaluation involves: (1) programs, each task as a different controller, and (2) report, including links to a video showing each different task. Points will be taken off for any robot that crashes into walls or gets stuck in any way. Note that the TA will be testing the different tasks under slightly different mazes to evaluate your solutions.

D.1 Task Presentation (90 points)

The following section shows the rubric for the tasks shown in the video:

- Task 1 (45 points)
  - Robot correctly maps each cell for the entire maze from the initial starting location (30 points)
  - Robot stops after mapping 90% of all cells (10 points)
  - Prints correct information at each timestep (5 points)
  - Robot hits walls (-5 points)
  - Robot cannot complete task under 3 minutes (-5 points)

- Task 2 (45 points)
    - Robot computes correct shortest path to the goal from known initial starting and ending location (10 points)
    - Robot computes correct shortest path to the goal from unknown initial starting and ending location (10 points)
    - Robot travels correct shortest path to the goal from known initial starting and ending location (10 points)
    - Robot travels correct shortest path to the goal from unknown initial starting and ending location (10 points)
    - Prints correct information at each timestep (5 points)
    - Robot hits walls (-5 points)
    - Robot cannot complete task under 3 minutes (-5 points)
- Statistics Task (10 points) - Required for grad students. Extra credit for undergrad students.
    - Statistics are correctly computed, analyzed, and graphed (10 points)
- Extra Credit (45 points)
    - Robot correctly applies the occupancy grid mapping algorithm (20 points)
    - Robot computes and navigates the correct shortest path from the start to the goal (10 points)
    - Robot stops after mapping 90% of all cells (5 points)
    - Robot correctly maps each cell for the entire maze from the initial starting location (5 points)
    - Prints correct information at each timestep (5 points)
    - Robot hits obstacles (-5 points)
    - Robot cannot complete task under 3 minutes (-5 points)

NOTE: Do not make calls to print statements inside the “while” loop or functions called from the “while” loop more than once per cell (do not print at each time step). Otherwise, you will be penalized up to 15% of your lab grade. You may have print statements outside the “while” loop.

D.2 Task Report (10 Points)

The report should include the following (points will be taken off if anything is missing):

- Mathematical computations for all kinematics. Show how you calculated the speeds of the left and right servos given the input parameters for each task. Also, show how you decide whether the movement is possible or not.
- Conclusions where you analyze any issues you encountered when running the tasks and how these could be improved. Conclusions need to show an insight of what the group has learnt (if anything) during the project. Phrases such as “everything worked as expected” or “I enjoyed the project” will not count as conclusions.
- Video uploaded to Canvas showing the robot executing the different tasks. You should include in the video a description, written or voice, of each task. You can have a single or multiple videos. Note that videos will be critical in task evaluations.
- Deductions:
    - Handwritten text or images (-5 points)
    - Missing video link requires in person presentation (-20 points)
    - The submission format is not correct (see assignment details) (-10 points)
    - Failure to answer TA's email within 48 hours of the initial email (-50 points)
    - Additional deductions may apply depending on submission

Videos

Video needs to be clear and audible and must show the robot performing the correct path and showing the program outputs printed to the console, as outlined in the assignment. TA may ask you additional questions to gauge your understanding via Canvas Message or MS Teams. Failure to reply may result in point deduction.

- Task 1 Video
  - Record a single video of the robot performing the mapping task of each map.
- Task 2 Video
  - Record a single video of the robot performing path planning task of each map.
- Task 3 Video (Extra Credit)
  - Record a single video of the robot performing extra credit task of each map.

E. Lab Submission

Each student needs to submit the programs and report through Canvas under the correct assignment. Submissions should have multiple file uploads containing the following:

- The "zip" file should be named "yourname_studentidnumber_labnumber.zip" and should contain the Python file containing your controller. Controllers should be named as "Lab5_Task*X*.py", where *X* is the task number. The zip file should contain any supporting python files needed to run your code. For example, if you created functions to perform actions and these are kept in a file, this file needs to be included. At the top of each python file include a comment with the relative path from FAIRIS_Lite that this file needs to be placed in. Example #WebotsSim/libraries/my_functions.py.
- Videos should be contained in a separate zip folder with the name "yourname_studentidnumber_labnumber_videos.zip" and added as a separate file upload to Canvas.
- The report should be in PDF and should be uploaded to Canvas as a separate file.
- All zip files must be in .zip format. We will not accept RAR, 7z, tar, or other.