CDA 4621 / CDA 6626

Grad Total: 110 points + Extra Credit

Undergrad Total: 100 points + Extra Credit

Spring 2024

Lab 4

Localization

Total: 100 points

Due Date: 4-2-2024 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 probabilistic robot localization. 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 1 x 1 meters. There are four colored cylinders (yellow, red, blue & green) located in the corners of the arena, having the same fixed size (radius "r" = 0.314 meters, height "h" = 1.5 meters), and centered "c" at the corners of the grid.

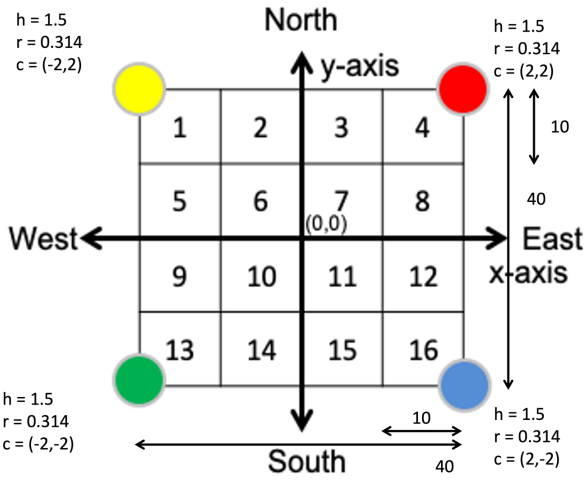
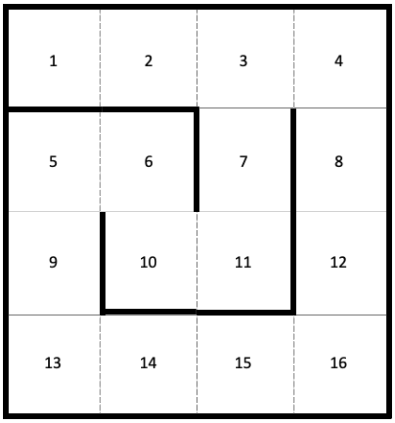


Figure 1. Global reference frame and grid cell numbering scheme. All units in meters.

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 $s=(x, y, n, \theta)$, 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.

(c) State probability. State probability is given by p(*s*) and should be based on state prediction, measurement, and update as studied in class. Use the sensor model wall measurement probabilities shown in Figure 3. You will need to calculate the probability that the robot is in each cell based on the sensor readings every time the robot visits a cell. Take into account the robot orientation and ideally, perform the measurement computations when the robot is aligned with a major cardinal orientation, i.e. when parallel to walls. The resulting probabilities should be normalized after each update cycle.

Sensor Model

	"no wall" s=0	"wall" s=1	
p(z=0 | s=0)	.7	.9	p(z=1 | s=1)
p(z=1 | s=0)	.3	.1	p(z=0 | s=1)

Figure 3. Sensor model wall measurement probabilities.

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: Landmark Localization
- Task 2: Wall Localization
- Statistics Task: Statistics on Localization
- Extra Credit Task: Kalman or Particle Filter Localization

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: Landmark Localization

Compute the robot pose from landmarks, using cameras, IMU, or any other sensor. Landmark positions are known in advance. Use the open world shown in Figure 4. The robot will move around the environment following any desired path starting from an unknown initial pose. You are not allowed to use the GPS or **starting_position** attribute to obtain the starting location. The pose should be estimated using triangulation or trilateration. You may use the IMU to compute orientation. The robot should end the task when all grid cells have been navigated. Print the estimated pose every time the robot enters a new grid cell.

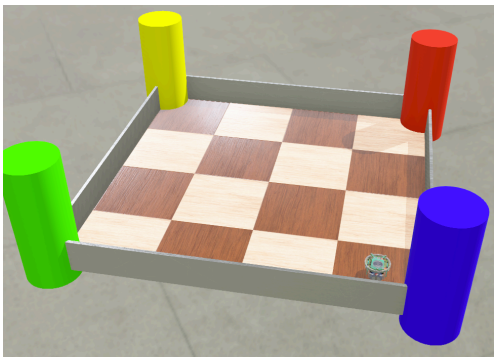


Figure 4. World with colored landmarks.

C.2 Task 2: Wall Localization

Compute wall-based localization estimates for each grid cell. The map is known in advance and the robot may follow any desired path. Test with the worlds shown in Figure 4, starting from an unknown initial pose, however the robot can know the initial starting position using the **starting_position** attribute. The robot should end the task when all grid cells have been navigated. Compute the estimated state probabilities using the sensor model shown in Figure 5. Assume a "perfect" motion model, i.e. no noise. Print probabilities for all grid cells at the estimated main cardinal orientation (based on IMU), every time the robot enters a new grid cell or rotates 90 degs. Do not consider the motion model.

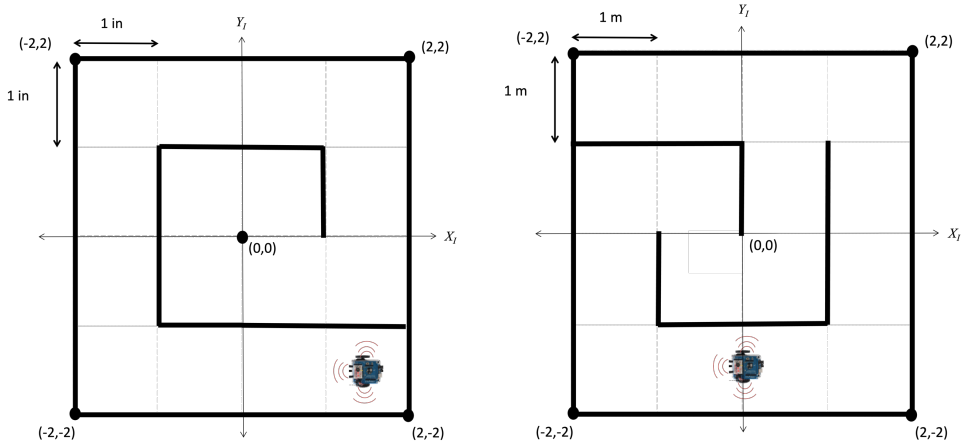


Figure 5. Different robot mazes for wall localization task.

C.3 Statistic Task: Statistics on Localization

Perform statistics on the estimated while comparing them to GPS output:

- Compare results from Task 1 showing variation in state estimation in different grid cells with robot starting from 3 different starting locations.
- Compare results from Task 2 showing variation in state estimation in different grid cells with robot starting from 3 different starting locations.
- Discuss your results including any navigation errors.

C.4 Extra Credit Task: Kalman or Particle Filter Localization

Apply either Kalman or Particle Filter localization to the maze shown in Figure 5 starting from an unknown initial pose. The robot may follow any desired path and may use landmark and/or wall information. The robot should end task when all grid cells have been navigated, or after 3 minutes. Print the estimated state probabilities every time the robot enters a new grid cell. You may use any third-party code. If not developed by you, then: (a) reference the source code origin, (b) fully integrate with Webots, and (c) clearly explain the code and integration with Webots. You will be graded on your own code, integration, and simulations.

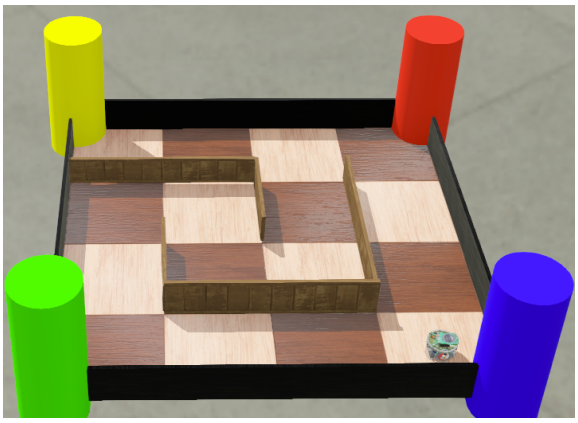


Figure 5. Maze with Landmarks.

D. Task Evaluation

Task evaluation is based on: (1) program code, (2) report, including a link to a video showing each different task, and (3) task presentation of robot navigation.

D.1 Task Presentation (90 points)

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

- Task 1 (45 points)
  - Prints correct information from all cells visited (20 points)
  - Keeps track of X and Y coordinates within a range of 0.3 m (10 points)
  - Keeps track of robot's current cell (10 points)
  - Navigates to all cells in grid (5 points)
  - Prints sensor information at each timestep (-5 points)
  - Robot hits walls (-5 points)
- Task 2 (45 points)
  - Prints correct information for all cells visited (20 points)
  - Keeps track of X and Y coordinates within a range of 0.3 m (10 points)
  - Keeps track of robot's current cell (10 points)
  - Navigates to all cells in grid (5 points)
  - Prints sensor information at each timestep (-5 points)
  - Robot hits walls (-5 points)
- Statistics Task (10 points) - Required for grad students. Extra credit for undergrads.
  - Statistics are correctly computed, analyzed, and graphed (10 points)
- Extra Credit (50 points)
  - Prints correct information for all cells (20 points)
  - Keeps track of robot's current cell (10 points)
  - Keeps track of X and Y coordinates within a range of 0.3 m (10 points)
  - Keeps track of orientation within a range of $\pi/8$ radians (5 points)
  - Navigates all grid cells (5 points)
  - Prints sensor information at each timestep (-5 points)
  - Robot hits walls (-5 points)

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 incorrect (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 trilateration-based localization.
    - Make sure that the video shows the required printed information.
- Task 2 Video
    - Record a video of the robot performing wall-based localization.
    - Be sure to record the robot performing wall-based localization in both the world with and without noise (clearly distinguish between the two).
    - Make sure that the video shows the required printed information.
- Extra Credit Video
    - Record a single video of the robot performing the particle filter localization.
    - Be sure to record the robot performing wall-based localization in both the world with and without noise (clearly distinguish between the two).
    - Make sure that the video shows the required printed information.

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 "Lab4_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.