Lab 4: Localization

CDA 4621
Fall 2024
Lab 4: Localization
Due Date: 12-2-2024 by 11:59pm

Objective

This lab will teach you about probabilistic robot localization. You will use: (1) Global Reference Frame and Grid Cell Numbering and (2) World Representation.

Requirements

Programming: Python
Robot: Webots 2023b (FAIRIS)

Sensors Utilized

To successfully complete this assignment, you will need to utilize several sensors on the robot. Each sensor provides unique data essential for robust navigation and environmental awareness. Below is an overview of the sensors, their specific functionalities, and references to previous assignments for guidance on usage:

- **Encoders:** The encoders track the rotation of each wheel, enabling the calculation of distance traveled and rotational changes. By using encoder readings, you can estimate the robot's displacement and changes in orientation over time, aiding in precise position tracking. *Refer to Lab 1 for an introduction to encoder usage.*
- **Compass:** The compass sensor returns the robot's orientation relative to the world frame, typically measured in degrees from the east direction (the x-axis). This sensor is essential for maintaining and adjusting the robot's heading during navigation tasks. *Refer to Lab 1 for initial setup and utilization of the compass.*
- **Lidar:** The Lidar sensor captures a 360-degree range image of the environment by measuring distances to obstacles and landmarks at various angles. This information allows the robot to map surroundings, avoid obstacles, and make path-planning decisions based on spatial awareness. *Refer to Lab 2 to review Lidar integration and data handling.*
- **Camera with Object Recognition:** The onboard camera, integrated with object recognition capabilities, allows the robot to detect and differentiate various objects in its environment. Specific landmarks, such as colored cylinders, can be identified, providing visual cues for navigation and goal localization. *Refer to Lab 3 for implementation details on using the camera and object recognition.*

Each of these sensors provides critical information that will need to be integrated into your navigation algorithm to ensure accuracy and efficiency. Be sure to review the respective labs to understand how to utilize these sensors effectively as you design your solution.

Global Reference Frame and Grid Cell Numbering

The reference frame is set up as follows:

- The positive y-axis points North, while the negative y-axis points South.
- The positive x-axis points East, and the negative x-axis points West.
- The origin, (0,0), is located at the center of the arena.

The arena is divided into a 5x5 grid of 25 cells, numbered from 1 to 25. Each grid cell is a 1 x 1 meter square, creating a structured layout for navigation and localization.

In each corner of the arena, there is a colored cylinder that serves as a landmark. These cylinders have a fixed size:

Global Reference Frame and Grid Cell Numbering (cont.)

- Radius, $r = 0.25$ meters
- Height, $h = 1.5$ meters
- Each cylinder is centered at the corner of a grid cell.

The four cylinders are uniquely colored (yellow, red, blue, and green) to assist in visual identification and orientation within the arena.

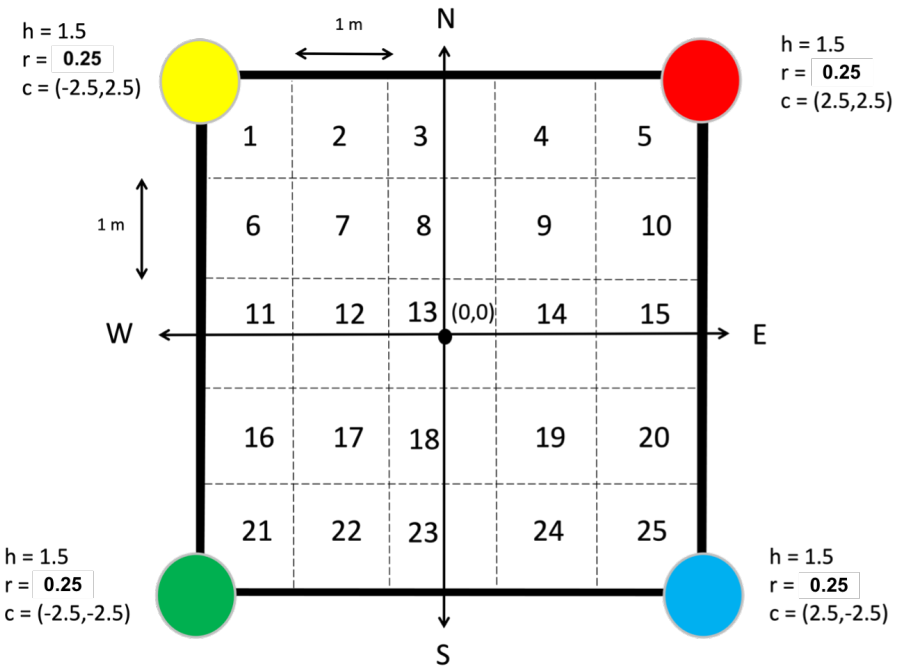


FIGURE 1. Global reference frame and grid layout for Task 1 (*maze7.xml*). The arena is divided into a 5x5 grid with colored cylinders located in each corner. All units are in meters.

Refer to this layout as you plan and execute navigation strategies within the defined grid.

Wall Configuration Data Structure

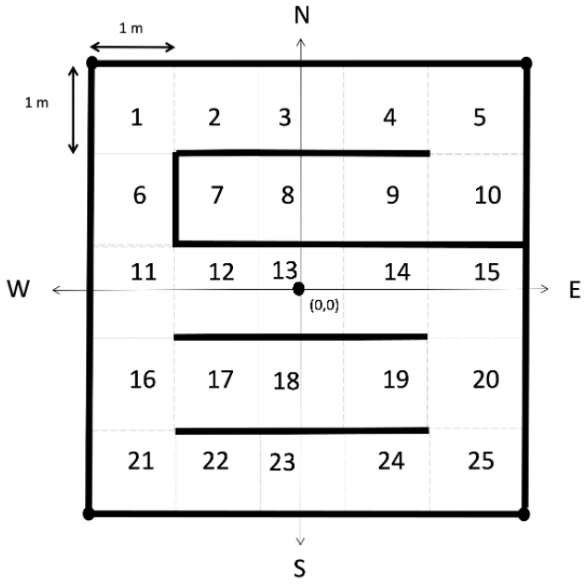
For this assignment, you will need to implement a data structure to represent the wall configuration of each grid cell in the arena. This structure will allow your robot to recognize and navigate around obstacles effectively.

As a conceptual guide, consider representing the wall configurations using a 25x4 matrix format, as illustrated in Figure 2. In this example:

- The matrix corresponds to the 25 grid cells in the arena.
- Each cell in the matrix contains four entries representing the presence or absence of walls in the West-North-East-South (WNES) orientation.
- A ”W” (wall) entry indicates the presence of a wall, while an ”O” (open) entry denotes the absence of one.

While you are encouraged to design your own data structure, this example can serve as a reference to help you get started.

Wall Configuration Data Structure (cont.)



Cell	Walls (WNES)
1	WWOO
2	OWOW
3	OWOW
4	OWOW
5	OWWO
6	WOWO
7	WWOW
8	OWOW
9	OWOW
10	OOWW
11	WOOO
12	OWOW
13	OWOW
14	OWOW
15	OWOO
16	WOOO
17	OWOW
18	OWOW
19	OWOW
20	OOWO
21	WOOW
22	OWOW
23	OWOW
24	OWOW
25	OOWW

FIGURE 2. Example of a wall configuration representation. The left side illustrates the physical world, while the right side shows the corresponding 25x4 matrix representation using WNES orientation.

Think carefully about how your data structure will allow you to efficiently check for walls in each direction and update it as the robot moves through the arena.

Background: Localization with Landmarks and Normal Probabilities

In this section, we will introduce the concept of using landmarks for robot localization within a grid. The robot uses distance measurements from known landmarks to estimate its location within a 5x5 grid by comparing these measurements with precomputed distances from each grid cell to the landmarks.

For each grid cell C_N, we compute the likelihood of the robot being in that cell based on the measured distances to landmarks. This probability is calculated by comparing:

- D_{R-L_i}: The distance from the robot's current position to each landmark L_i.
- $D_{C_N-L_i}$: The precomputed distance from the center of each cell C_N to each landmark L_i.

Using a normal (Gaussian) probability distribution function, we can calculate the likelihood of the robot's position given these distances.

Calculating Normal Probability in Python

To calculate the probability density for a given distance measurement, you can implement the normal distribution probability function using the following formula:

$$f(s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(s-\mu)^2}{2\sigma^2}\right)$$

where: - s is the mean (precomputed distance from the grid cell to the landmark), - μ is the observed distance (measured distance to the landmark), - σ is the standard deviation, representing expected measurement noise.

Write a function in Python to calculate this probability. Here is the pseudocode to guide you:

```
# Pseudocode for calculating the normal probability density
def calculate_normdist(s, mu, sigma):
```

Background: Localization with Landmarks and Normal Probabilities (cont.)

```
    """
    Calculate the normal probability density for a given state.

    Parameters:
        s (float): Precomputed distance from the grid cell to the landmark.
        mu (float): The observed distance (measured distance to the landmark).
        sigma (float): The standard deviation (expected noise level).

    Returns:
        float: The probability density value.
    """
    # Step 1: Calculate the coefficient (1 / sqrt(2 * pi * sigma^2))
    coefficient = 1 / (sqrt(2 * pi * sigma^2))

    # Step 2: Calculate the exponent (-((s - mu)^2) / (2 * sigma^2))
    exponent = -((s - mu)^2) / (2 * sigma^2)

    # Step 3: Calculate the probability by multiplying the coefficient with the
   exponentiated result
    probability = coefficient * exp(exponent)

    return probability
```

In this pseudocode:

- `s` represents the precomputed distance from a specific grid cell center to the landmark.
- `mu` represents the observed distance from the robot to the landmark.
- `sigma` represents the standard deviation, which models the expected measurement noise or uncertainty.

Example Calculation

To illustrate how to use this function, let's walk through an example. Suppose the robot measures a distance of $\mu = 2.915$ meters to a particular landmark, and the precomputed distance from a given cell center to the same landmark is $s = 2.828$ meters. Let's assume a standard deviation of $\sigma = 1$.

Using the pseudocode function `calculate_normdist`, we would calculate the probability as follows:

```
# Example values
mu = 2.915  # Measured distance to the landmark
s = 2.828  # Precomputed distance from cell center to landmark
sigma = 1   # Standard deviation

# Calculate the probability
probability = calculate_normdist(s, mu, sigma)
print("Probability:", probability)
```

In this example:

- The smaller the difference between `s` and `mu`, the higher the probability, indicating that the robot is likely close to the grid cell associated with `mu`.
- This probability represents the likelihood of the robot being at the cell given the measured distance to the landmark.

You will use this function to compute the probability for each cell in the grid and determine the cell with the highest probability as the most likely position of the robot.

Alternatively, you can use the Python Scipy library's norm function to achieve this.

Background: Localization with Landmarks and Normal Probabilities (cont.)

```python
# Import package
from scipy.stats import norm

# Parameters
mu = 2.915  # the measured distance to the landmark
s = 2.828  # the precomputed distance from a grid cell to the landmark
sigma = 1  # standard deviation

# Calculate the normal probability
probability = norm.pdf(s, mu, sigma)
print(probability)
```

Background: Localization Using Wall Configuration and Sensor Readings

In this section, we introduce the concept of using a sensor model to estimate the robot's position within a grid maze based on wall configurations. The sensor model, illustrated in Figure 3 (Figure 3), provides a way to calculate the probability of the robot being in a given cell based on its surroundings. This model relies on the robot's ability to detect walls in four directions (north, south, east, and west) using LiDAR or another distance-sensing device.

Sensor Model and Wall Configuration

The sensor model allows us to compare the observed wall configuration around the robot to known wall configurations of each cell in the grid maze. Each cell has a specific layout of walls, which can be represented in a data structure where each cell's configuration is recorded. For example, a cell might have walls on the north and east sides but open paths to the south and west. These configurations are stored in advance, providing a reference for comparing with the robot's observations.

Collecting Sensor Readings

To use the sensor model for localization, the robot must gather distance readings in four directions:

- **North:** The LiDAR measures the distance to any object directly in front of the robot.
- **South:** The LiDAR measures the distance to any object directly behind the robot.
- **East:** The LiDAR measures the distance to the right of the robot.
- **West:** The LiDAR measures the distance to the left of the robot.

These readings will indicate the presence or absence of walls in each direction (*You will need to consider whether there is a wall based on a min and max distance for each cell*), forming an observed configuration of the surrounding area.

Calculating Probabilities Using the Sensor Model

Once the robot has collected LiDAR readings in all four directions, it compares this observed wall configuration with the pre-stored configurations of each cell in the grid maze. The sensor model assigns a probability to each cell based on how closely the observed configuration matches the expected configuration of that cell.

For each cell C_N, the probability $P(z = 0|s = C_N)$ is calculated based on the match between:

- The actual wall configuration stored for C_N, which specifies where walls are located around the cell.
- The observed wall configuration derived from the robot's LiDAR readings.

If the observed wall configuration matches the stored configuration for a given cell, that cell will have a high probability, represented by $P(z = 0|s = C_N)$. If the configurations do not match, the probability $P(z = 0|s = C_N)$ for that cell will be lower. This probability calculation is based on each directional match.

Weighted Probability Calculation

Background: Localization Using Wall Configuration and Sensor Readings (cont.)

To determine the likelihood of being in a specific cell, calculate the weighted probability by multiplying the individual probabilities for each direction (north, south, east, and west) based on the match between observed and expected wall configurations. The weighted probability for each cell is given by:

$$P(z = 0|s = C_N) = P(z_{\text{north}} = 0|s = C_N) \times P(z_{\text{south}} = 0|s = C_N) \times P(z_{\text{east}} = 0|s = C_N) \times P(z_{\text{west}} = 0|s = C_N)$$

where each term $P(z_{\text{direction}} = 0|s = C_N)$ represents the probability that the observed reading in a specific direction (e.g., north) matches the expected wall configuration in that direction for cell C_N.

By calculating the combined probabilities $P(z = 0|s = C_N)$ for all cells in the maze, the robot can determine the most likely cell based on the current wall configuration around it. This approach provides a probabilistic localization method that leverages the robot's immediate surroundings rather than relying solely on distance measurements to landmarks.

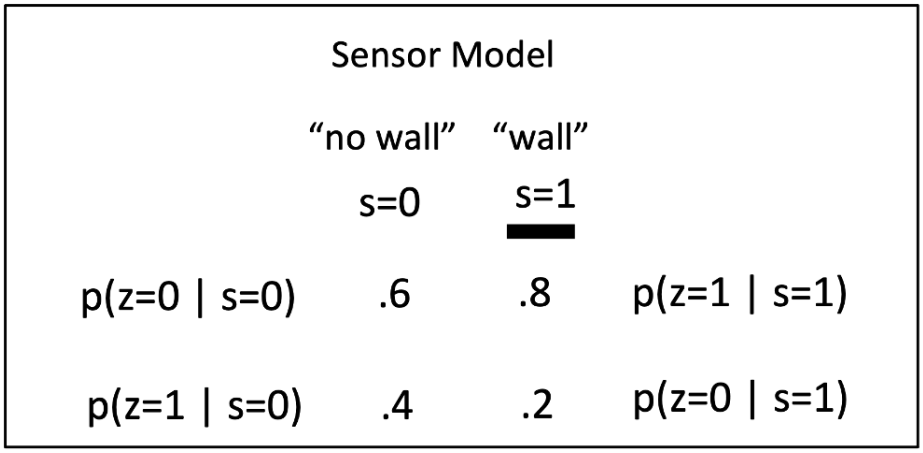


FIGURE 3. Sensor model wall measurement probabilities for state probability calculations.

Task 1: Localization with Landmarks

In this task, you will estimate the robot's position (pose) using landmarks and sensors, such as cameras, IMU, or LiDAR. The positions of the landmarks are known in advance, and the robot will navigate through an open-world environment (see Figure 4) starting from an unknown initial pose. You are not allowed to use the GPS to determine the starting location. The goal is to estimate the robot's pose based on distance measurements to landmarks and navigate the entire grid, printing and comparing the estimated positions as the robot moves.

- **Pose Estimation Using Normal Distribution:**
  - For each grid cell C_N, use a normal distribution function to estimate the robot's pose by comparing the robot's measured distances to each landmark with the precomputed distances from the cell centers to the landmarks.
  - Measure the distances from the robot's current position to each landmark using available sensors (e.g., camera, LiDAR, IMU).
  - Calculate the probability that the robot is in each cell by computing the weighted probability as the product of normal probabilities based on the distance comparison:

$$W_{C_N} = N_{C_N - L_1} \times N_{C_N - L_2} \times \cdots \times N_{C_N - L_k}$$

    where $N_{C_N - L_i}$ is the normal probability for cell C_N based on landmark L_i.
- **Identify the Current Cell Based on Highest Probability:**
  - Calculate the weighted probabilities for each cell each time the robot enters a new cell.

Task 1: Localization with Landmarks (cont.)

  - Identify the cell with the highest probability $W_{C_{best}}$ and assume this cell represents the robot's current position.
  - Print the estimated pose (coordinates of C_{best}) each time the robot moves into a new cell. This output should be displayed in the console and recorded in the video.
- **Complete Coverage of All Grid Cells:**
  - Program the robot to navigate through the entire grid, ensuring that it visits each cell at least once.
  - Use a systematic approach, such as a lawnmower pattern, to ensure complete coverage without unnecessary revisits.
  - Keep track of the cells visited by assuming the highest-probability cell is the robot's current location.
  - The task ends when the robot has successfully navigated all cells in the grid.
- **Data Collection and Analysis:**
  - For each estimated position, obtain the actual (x, y) position from the GPS sensor for comparison (this can be used only for evaluation, not for navigation).
  - Record both the estimated and actual positions for each grid cell entry and log this data.
  - Use the collected data to create a graph comparing estimated positions to actual positions.
  - Include a detailed analysis in the lab report, discussing any discrepancies between estimated and actual positions and evaluating the accuracy of the localization method.
- **Recording Requirements:**
  - Record a video showing the robot starting from at least two different initial positions and navigating the grid to complete the task.
  - Ensure that the console output of the calculated probabilities and the estimated pose for each cell is visible in the video.
  - The video should clearly demonstrate the robot's movement, pose estimation calculations, and grid coverage.

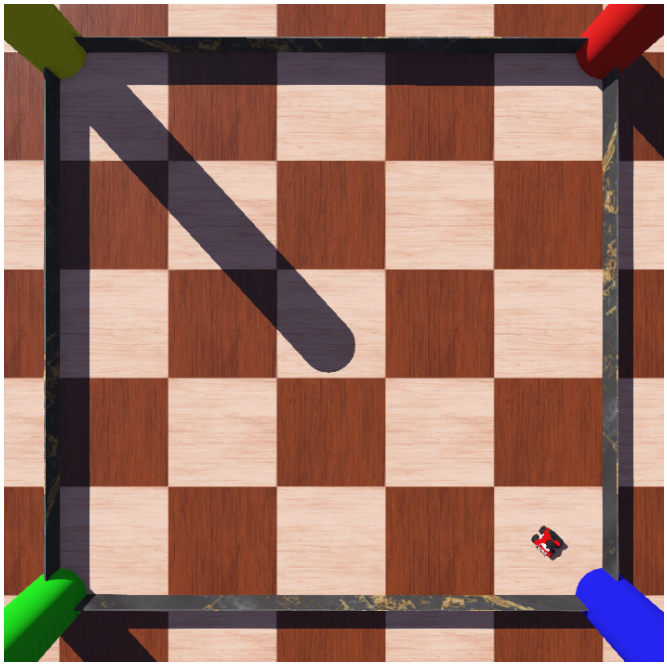


FIGURE 4. Open world layout for Task 1 with known landmarks for localization (maze file *maze7.xml*).

Task 1: Localization with Landmarks (cont.)

Be sure to include a thorough comparison of the predicted pose versus the GPS readings in your report, as this will be a key part of your evaluation.

Task 2: Localization with Walls and Landmarks

In this task, you will implement an algorithm that uses the sensor model to estimate the robot's position as it navigates through a 5x5 grid maze. The objective is to use LiDAR sensor readings to calculate probabilities for each cell based on wall configurations, as described in the background section. **The robot's starting location will not be known to the robot.** This task will demonstrate how the sensor model can aid in predicting the robot's location, even though it may not lead to full localization due to the symmetry of the maze.

The testing environment is shown in Figure 5, and the complete map configuration is known in advance to the robot. However, the robot will start from an unknown position in the maze, and students are not permitted to use GPS or any other starting position attribute to determine the robot's initial location.

- **Navigating Through the Maze:**
  - Program the robot to navigate from cell to cell within the 5x5 grid maze. Each time the robot enters a new cell, it should stop and take sensor readings in four directions: north, south, east, and west, aligned with the cardinal directions.
  - The robot should continue moving until it has entered at least 15 different cells. If the robot encounters a dead end, it is allowed to backtrack to a previous cell and continue navigating.
- **Map Data Structure and Sensor Model:**
  - Students must implement a data structure that stores the wall configurations for each cell in the maze, aligning the wall measurements with the cardinal directions (north, south, east, and west). This data structure will represent the known map of the environment.
  - Each time the robot enters a new cell, it should use the sensor model to calculate the probability $P(z = z'|s = C_N)$ for each of the 25 cells based on the observed wall configuration and compare it to the stored wall configurations for each cell.
  - Print the calculated probabilities for all 25 cells to the console each time the robot enters a new cell. This output should be visible in the recorded video.
- **Listing Cells with Highest Probability as Predictions:**
  - After calculating the probabilities for all cells, identify the cell(s) with the highest probability. If there are multiple cells with the highest probability (due to symmetry in the maze), list all of them as the robot's predicted position.
  - Print the predicted cell(s) to the console as the robot's best estimate of its current position. This prediction should be visible in the video as well.
- **Video Requirements:**
  - Record a video of the robot navigating the maze and performing the probability calculations. The video should show the robot starting from multiple initial locations and moving through at least 15 cells for each starting position.
  - Ensure that both the probability calculations and the list of predicted cells are printed to the console and clearly visible in the video. This will demonstrate the robot's process of estimating its position based on wall configurations.
- **Note on Expected Outcomes:**
  - The purpose of this task is to illustrate how the sensor model can be used to predict the robot's location based on surrounding wall configurations. However, due to the symmetry of the maze, the robot will not be able to fully localize itself using only the sensor model, as multiple cells may have identical wall configurations.

Task 2: Localization with Walls and Landmarks (cont.)

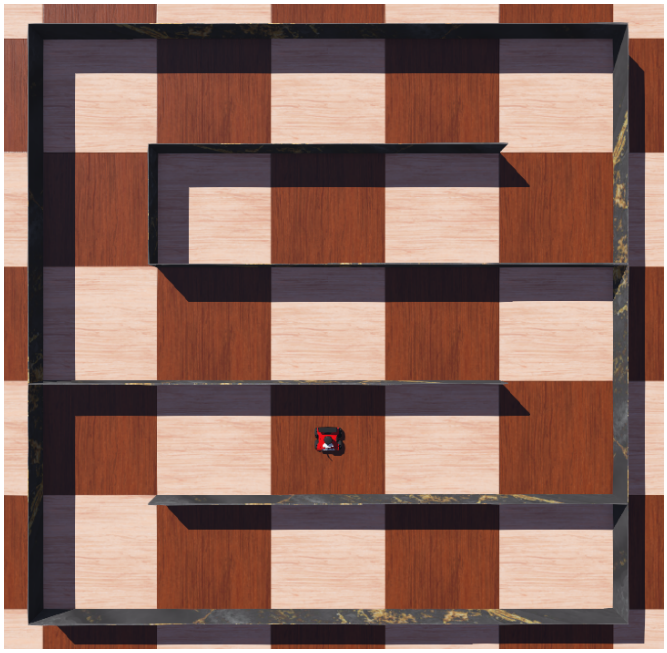


FIGURE 5. Maze environment for Task 2, with known wall configurations for Bayesian inference-based localization (maze file *maze8.xml*).

This iterative application of motion and measurement updates enables precise localization and navigation within the maze using Bayesian inference.

Extra Credit Opportunity (10 points)

For students who wish to attempt a more accurate localization, an additional 10 points of extra credit will be awarded to the student who provides the correct grid localization for all grid cells in the shortest time. To be considered, students must include a plot comparing predicted positions with GPS data and record the fastest completion time in their report. Additionally, students need to explain exactly how they implemented accurate predictions without ever using the GPS or starting location.

Code Evaluation Task 1 (45%)

- **Probability Calculation Using Landmarks (10 points):**
  - **Full Marks (10):** The algorithm accurately calculates the probability for each of the 25 cells based on measured distances to landmarks every time the robot enters a new cell, using the normal distribution formula precisely.
  - **Acceptable (7-9):** Probabilities are calculated for each cell, with occasional minor errors or small deviations from the normal distribution formula.
  - **Poor (1-6):** Probability calculations are implemented, but frequent errors or incorrect application of the normal distribution formula significantly impact accuracy.
  - **No Marks (0):** Probability calculations are either missing or incorrectly implemented for most cells, with no functional results.
- **Printing and Displaying Most Likely Cell (5 points):**
  - **Full Marks (5):** The algorithm correctly identifies and prints the cell(s) with the highest probability as the robot's predicted position each time it enters a new cell. This output is clearly visible in the video.

Task 1: Motion to Goal (cont.)

  - **Acceptable (3-4):** The algorithm identifies and prints the most likely cell, with minor errors or inconsistencies in visibility in the video.
  - **Poor (1-2):** The most likely cell is occasionally identified, but with significant errors or missing visibility in the video.
  - **No Marks (0):** The most likely cell is not identified or displayed in the video.
- **Using Highest Probability to Track Visited Cells (10 points):**
  - **Full Marks (10):** The algorithm accurately uses the highest probability to determine the current cell and maintains an accurate record of visited cells.
  - **Acceptable (7-9):** The algorithm tracks visited cells with occasional errors in determining the highest probability cell or maintaining an accurate visited cells record.
  - **Poor (1-6):** The algorithm tracks visited cells inconsistently, with frequent errors in determining the highest probability cell or maintaining a reliable record.
  - **No Marks (0):** The algorithm does not track visited cells accurately, with the record incomplete or unreliable.
- **Complete Grid Coverage (10 points):**
  - **Full Marks (10):** The robot successfully covers all cells in the 5x5 grid at least once, demonstrating complete and systematic grid coverage.
  - **Acceptable (7-9):** The robot covers most cells, with occasional missed cells or unnecessary revisits.
  - **Poor (1-6):** The robot fails to cover a substantial portion of the grid or exhibits frequent unnecessary revisits, impacting systematic coverage.
  - **No Marks (0):** The robot fails to cover most of the grid, with random or inconsistent movement patterns.
- **Plotting and Analysis (10 points):**
  - **Full Marks (10):** A clear and accurate plot is provided, comparing the robot's predicted positions (based on the highest probability cell) with its actual GPS-based positions. A thorough analysis of discrepancies is included in the lab report.
  - **Acceptable (7-9):** A plot is provided with a reasonably accurate comparison of predicted and actual positions, with minor errors or a limited analysis in the report.
  - **Poor (1-6):** The plot lacks clarity or has significant errors, with limited or unclear analysis in the lab report.
  - **No Marks (0):** The plot is missing or incorrect, and the lab report lacks any meaningful analysis.

Deductions

- **Print Statement Visibility (-20 points maximum):** Up to 20 points may be deducted if required print statements are not clearly visible in the submitted video. The severity of the deduction will depend on the clarity and frequency of missing print statements.
- **Not Showing Multiple Starting Locations (-10 points):** A 10-point deduction will be applied if the video for Task 1 does not show multiple starting locations.
- **Robot Collisions with Wall (-5 points):** A 5-point deduction will be applied each time the robot collides with a wall, up to a maximum of 5 points.

Code Evaluation Task 2 (45%)

- **Probability Calculation Using Sensor Model (15 points):**

Code Evaluation Task 2 (cont.)

  - **Full Marks (15):** The algorithm accurately calculates probabilities for each of the 25 cells based on wall configurations every time the robot enters a new cell, correctly implementing the sensor model formula.
  - **Acceptable (11-14):** The algorithm calculates probabilities with occasional minor errors, showing a generally correct but partially inconsistent use of the sensor model formula.
  - **Poor (1-10):** Probability calculations are implemented, but frequent errors or major inaccuracies in the sensor model formula significantly impact the result.
  - **No Marks (0):** Probability calculations are missing or incorrectly implemented for most cells, providing no meaningful results.
- **Printing and Displaying Predictions Based on Probabilities (10 points):**
  - **Full Marks (10):** The algorithm correctly identifies and prints the cell(s) with the highest probability as the robot's possible positions each time it enters a new cell, with clear and consistent output visible in the video.
  - **Acceptable (7-9):** The algorithm identifies and prints the likely cells, with minor errors or occasional inconsistencies in the output visibility in the video.
  - **Poor (1-6):** The probable cells are sometimes identified but with major errors or missing output in the video.
  - **No Marks (0):** The probable cells are not identified, or output is consistently missing from the video.
- **Robot Movement Through Non-Repeated Cells (15 points):**
  - **Full Marks (15):** The robot moves through at least 15 unique cells, revisiting cells only when encountering dead ends, and adheres to a systematic movement pattern.
  - **Acceptable (11-14):** The robot moves through 15 cells with minor, unnecessary revisits or slight deviations from the systematic movement requirement.
  - **Poor (1-10):** The robot does not move through at least 15 unique cells or frequently revisits cells, failing to show a consistent or systematic approach.
  - **No Marks (0):** The robot fails to reach 15 unique cells or shows random, repeated movement without any clear pattern.

Deductions

- **Print Statement Visibility (-20 points maximum):** Up to 20 points may be deducted if required print statements are not clearly visible in the submitted video. The severity of the deduction will depend on the clarity and frequency of missing print statements.
- **Not Showing Multiple Starting Locations (-10 points):** A 10-point deduction will be applied if the video for Task 1 does not show multiple starting locations.
- **Robot Collisions with Wall (-5 points):** A 5-point deduction will be applied each time the robot collides with a wall, up to a maximum of 5 points.

Report & Video Evaluation (10%)

A project submitted without a report, or without including the corresponding task video, will not be graded and will receive a "0" for the complete lab.

Report Sections: The report should include the following (points will be deducted if anything is missing):

- **Mathematical Computations:** Show how you calculated the speeds of the left and right servos given the input parameters for each task.
- **Conclusions:** Analyze any issues encountered when running the tasks and discuss how these could be improved. Discuss your results, including any navigation errors. Conclusions need to

Report & Video Evaluation (cont.)

reflect an insight into what was learned during the project. Statements like "everything worked as expected" or "I enjoyed the project" will not count as valid conclusions.

- **Video:** Upload the video to Canvas showing the robot executing the task (details below).
- **Authorship Statement:** Include and sign the following statement at the end of the report:
  "I developed all the code and written report with the following exceptions:
  [Student name: signature, date]
  [Code sections (specify lines): references and additional comments]
  If any external tools were used to generate code, you need to include the prompt and output."

Task 1: Landmark-Based Localization

- Show the robot moving from cell to cell within the 5x5 grid.
- Each time the robot enters a new cell, display the calculated probabilities for all 25 cells based on distance measurements to landmarks. This information should be printed to the console and visible in the video.
- Show the robot's predicted position by printing the cell(s) with the highest probability. This prediction should be visible in the video each time the robot moves to a new cell.
- Demonstrate complete grid coverage by having the robot traverse all cells in the 5x5 grid at least once. The video should capture this full navigation path.
- Repeat the task from multiple starting locations.

Task 2: Sensor Model-Based Localization Using Wall Configurations

- Show the robot navigating through the 5x5 grid and entering at least 15 unique cells, only revisiting cells in cases of dead ends.
- Each time the robot enters a new cell, display the calculated probabilities for each of the 25 cells based on wall configurations observed in four directions (north, south, east, west). This information should be printed to the console and clearly visible in the video.
- Show the robot's possible position(s) by printing the cell(s) with the highest probability. This prediction should be visible in the video each time the robot moves to a new cell.
- Repeat the task from multiple starting locations.

Note: Ensure that all probability calculations, cell predictions, and multiple starting locations are clearly visible in the video for both tasks. This will demonstrate the functionality and robustness of your localization algorithms across varied initial conditions.

Lab Submission

For Lab 4 there are two separate Canvas Assignments: Lab 4 Code Submission and Lab 4 Report Submission. Please follow the Submission policy outlined below. **Note:** Failure to adhere to the submission policy will result in a reduction of 20 points.

Code Submission Policy

Students will need to upload a Python file (i.e. Webots controller) for each Task. The Python file should follow this naming convention *USF_ID_LabX_TaskY.py*, where *X* is the lab number and *Y* is the task number. Example *rockybull5_Lab1_Task1.py*. Additionally, if a student has created any functions that are not present in the controller but rather the *MyRobot.py*, the student will also need to upload that python file as well using the following naming convention *USF_ID_MyRobot.py*.

If custom functions are used in a student's submission and they are not in the controller or the MyRobot Python files, an automatic Zero will be assigned for the code submission.

Each file should be uploaded to the same submission. **DO NOT** upload a zip file, Canvas allows multiple uploads per submission. Failure to meet these submission policies will result in 20 points off.

Report Submission Policy

Students will need to upload their videos to their USF OneDrive or USF Sharepoint and generate a shareable link. Please ensure that you make it viewable to everyone or at the very least the TA and the Professor (use their emails to add them to the access list).

Students will need to upload a PDF file of their lab report. Be sure that the lab report contains the link to the videos and any Metrics requested by the Lab. These Metrics will be used to determine which student receives the extra credit. Use the naming convention *USF_ID_LabX_Report.pdf*. Only PDFs will be accepted.

If the student used ChatGPT to generate code they will also need to provide the conversation used to generate the code. This can either consist of screenshots of the conversation or a shareable link that the system provides. This should be a separate document and not part of the report.

Each file should be uploaded to the same submission. **DO NOT** upload a zip file, Canvas allows multiple uploads per submission. Failure to meet these submission policies will result in 20 points off.

TA may ask you additional questions to gauge your understanding via Canvas Message or MS Teams. Failure to reply may result in point deduction.