University of South Florida

CDA 4621 Control of Mobile Robotics

Lab Report 4

Date: 12/11/2019

Members: Gil Olenscki Neto and Jack Yuan Jie Yang

List of files uploaded:

1 – Lab 4 Report

This file contains the explanation of the main parts of the code.

2 – M2.py

This file contains the code responsible to generate all the menus and call the functions responsible to perform the specific tasks present in the main menu. Depending on the selection it makes the robot, calibrate, navigate through the maze randomly, navigate the entire maze and then generate a mapping of all the walls for each cell, and find the smallest path from one cell to another.

3 – UnthreadedWebcam.py

Code utilized to use the camera and capture the information necessary for the robot be able to identify the blobs.

4 – ThreadedWebcam.py

Code utilized to use the camera and capture the information necessary for the robot be able to identify the blobs.

5 – params.yaml

Code utilized to use the camera and capture the information necessary for the robot be able to identify the blobs.

6 – UnthreadedWebcam.cpython-35.pyc

Code utilized to use the camera and capture the information necessary for the robot be able to identify the blobs.

7 – ThreadedWebcam. cpython-35.pyc

Code utilized to use the camera and capture the information necessary for the robot be able to identify the blobs.

List of problems found while performing the different tasks

Problem 1:

When performing task 2 the robot was not stopping exactly at the middle of the cell after some point. This was occurring because both wheels were moving at different velocities by a small amount. This would end up making a snowball in which the robot would stop either too much forward or backwards into a cell and the algorithm would not work properly. To solve such issue, we created a function that makes the robot check whether it is in the middle or not and then position itself back in the middle by moving in reverse or adjusting it by moving it sideways until it reaches the middle of the cell exactly.

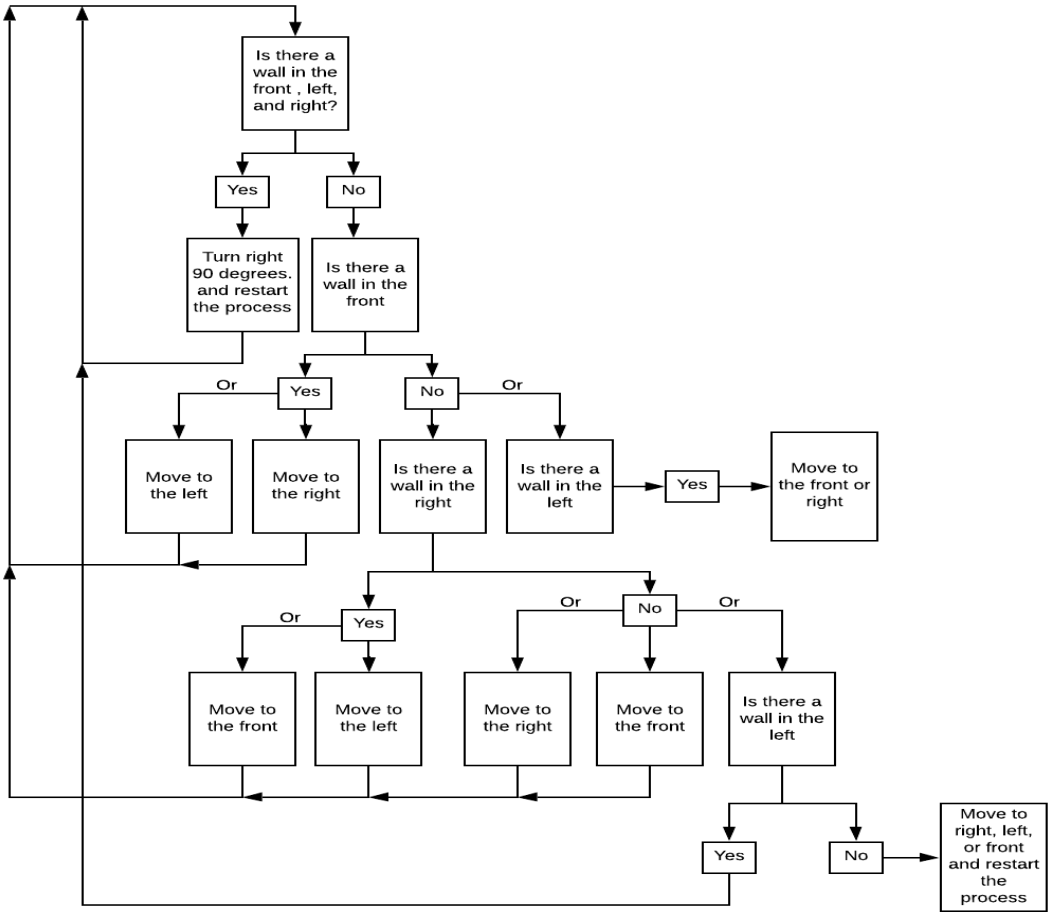
Problem 2:

When performing task 3 there was an issue when determining where the robot should go in case is was surrounded by walls in all sides besides the back and it had already visited that cell. We solved such issue by making the robot navigate through visited cells until it found a '?' in the map, then the robot would turn to the unvisited area.

Problem 3:

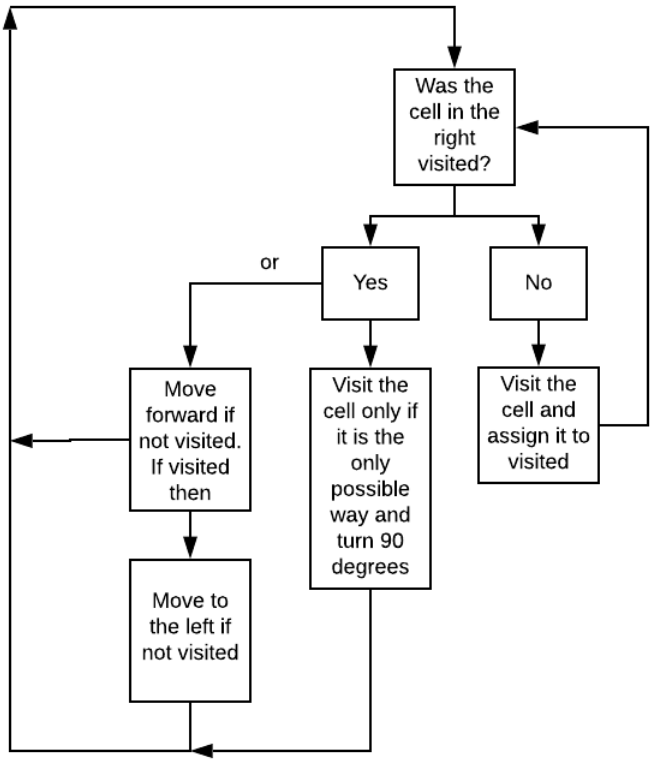
When performing task 4 there was an issue when determining which path to choose depending on which location the robot would start. To fix such issue we made the robot map be always based on its initial position when it started mapping the maze. This ensured that the robot could get to any position using the shortest path algorithm.

Diagram with brief explanation of the navigation algorithm:



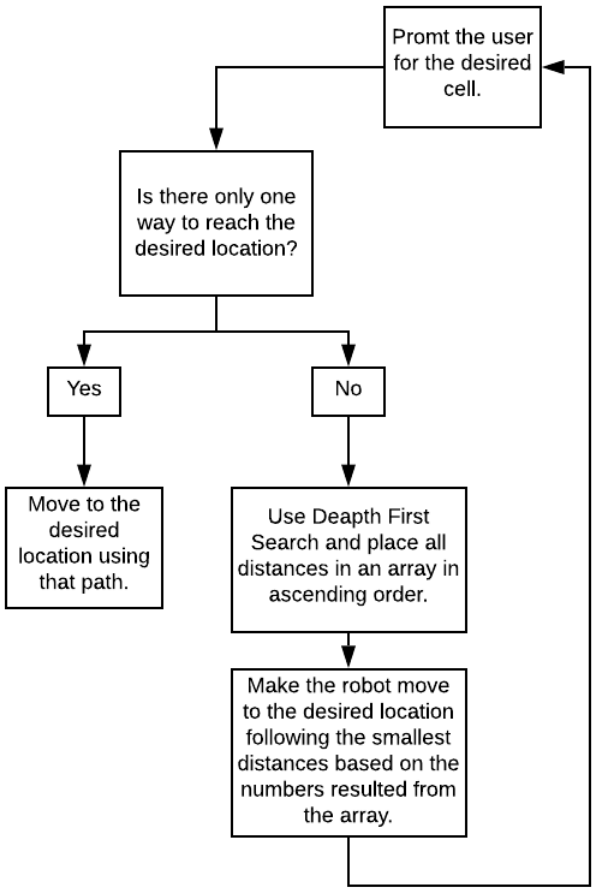
The navigation algorithm implemented first checks all sensors in order to know which of the locations (North, East, South, or West) has a wall. If the cell directly in front, right, or left does not have a wall that blocks the robot's movement then the robot moves to that cell. However, if there are multiple cells that the robot can go, it chooses the cell randomly. This random selection chooses a random number between 1 and the number of possible cells that it can move to. Taking that into consideration, based on the number generated randomly it moves to a specific cell. In addition to that, in case the robot moves to a cell in which all locations are blocked by a wall, the robot turns to the right and runs the loop again to check all locations, which will result in the robot turning to the right again and move to its previous location.

A diagram with brief explanation of the SLAM algorithm



The mapping algorithm uses the navigation algorithm with a class with a 2-dimensional array to keep track of the visited cells. In addition to that, the robot gives preference to visiting cells that were not visited unless the only possible way to move is going through a visited cell. The map is generated by changing the value in the matrix from '?' to nothing (an empty space) when the bool variable called visited becomes true. The robot knows when it visited a cell by knowing how much it moved from its previous position. Moreover, the robot terminates the mapping when all cells in the matrix are empty.

A diagram with brief explanation of the Path Planning algorithm



The path planning algorithm first maps the entire maze using the mapping algorithm and then prompts the user for the desired location. After the location is selected, the robot chooses the shortest path to move to that location. This is done by placing all cell as nodes in a graph and performing the depth first search algorithm to find the smallest distance(numbers) and organize them in ascending order, then this sequence will be placed in a stack and executed by popping the top of the stack as it moves to that cell.

Links for the robot performing

Task 2 – Random Navigation

https://www.youtube.com/watch?v=N34fwtbBBGY&feature=youtu.be

Task 3 – Localization and Mapping (SLAM)

https://www.youtube.com/watch?v=xPkiTewT8Og&feature=youtu.be

Task 4 – Path Planning

https://www.youtube.com/watch?v=0Q_wFZrtIl0&feature=youtu.be

Conclusion

This project portrayed a lot of challenges to properly implement each part. However, as we worked through every step, we were able to implement the functions utilized in the previous lab in combination with the new tasks present in this lab to understand every part required for it to successfully work.

The first main challenge encountered was when we were trying to ensure that the robot would always locate itself in the center of each cell. At first the robot wheels were not moving with the same velocity due to noise in the calibration, which would end up making the robot be slightly out of position. To overcome such issue, we made the robot analyze its current position and perform specific movements, such as, going in reverse, in order to position itself back in the center.

Another difficulty occurred when we were implementing the path planning algorithm. We ran into some difficulties about when the robot would perform this algorithm since it would sometimes get stuck in certain scenarios. However, we learned with our previous mistake and spent less time focusing in a non-working approach and we came up with different ideas, to handle all these conditions, we made several if statements with possible situations and how the robot should react to these situations. By doing so we were able to test how efficiently the task is being performed in addition to guaranteeing that every possible situation is addressed.