Autonomous lawnmower

Project completed by Oliver Walker and Wei Li

Report written by Oliver Walker

Abstract:

In this project the objective was to create a simulation of a self-sufficient lawnmower. The lawnmower should be able to be placed on your lawn and then cut all your lawn for you. The project utilizes the concepts gone over in PS6 where our robot using occupancy grid mapping to map out the area around it as it is mowing the lawn. All the generated map data is being used to generate statistics on the mower performance at the end of the program.

Introduction:

Our challenge is to have a lawnmower cut grass autonomously in an unmapped environment. We have made our robot capable of detecting any uncut grass in our simulated map as well as avoid the obstacles while moving. We choose he problem because it is interesting because it would be fun to watch the mower cut by itself while you sit back and relax. The concept is both functional and fun.

Project Description:

Objective:

The objective of this project is to simultaneously make the robot cut the grass and explore the map using Occupancy Grid Mapping.

Method:

We make the robot go counterclockwise and use spiral method. As the robot is exploring the map it is also cutting the grass around the robot in a 1.5-meter radius.

This is done with my cut_area function in the map_class.cpp. The function works by taking in the current coordinates of the robot translated into a spot on our vector of vectors and then in a loop I have the robot record everything within a 15 by 15 box around the robot as cut. It is 15 by 15 because each unit stands for a 10nth of a meter. 15 stands for 1 and a half meters. If the box goes out of bounds of the map or if the map is not labeled as open space, then it is ignored and not cut.

We noted that the robot can do a full circuit of any map in around 3 minutes, so we have the robot circle the map for 4 minutes at a time at a certain distance of wall following. After 4 minutes of wall following at that distance we slightly increase the distance by what represents half a meter of map. because our robot cuts in a 1.5-meter area, half a meter increases in the wall following function should produce decent results. At the 4 minutes intervals the robot would tilt inward to increase the distance by half a meter.

The next part of the code adds in all the uncut grass to our map with the add_uncut function I wrote. This function checks the entire main occupancy grid map for values less the 0.5 and ignores values representing uninitialized areas and adds all of that as uncut grass to the map that stores cut vs uncut values. The important thing to note in this part is that if the grass is marked as cut on the map, it will never be switched to uncut from this function.

Finally, we compare the results of how much area the robot managed to cut with it's 1.5 meters of cutting range. This is done in the compare results function. It simple scans through the map that stores uncut vs cut values and adds up the values denoting uncut grass, cut grass, and unknown area. The function then prints out all the information we gain from this data collection every 4 minutes of run time. The most important part of this function is it prints out the number of cut grass divided by the total number of grass to give us a percentage of how much grass we have cut.

On top of the above logic we are also constantly updating another occupancy grid map as done in problem set 6. This is done using our wonderful TA's PS6 homework solution as that accomplishes our goals for this part of the code perfectly. Essentially in this code the robot checks each of its sensors to look for a wall. If there is a wall a higher value is put down in that location. If there is empty space a lower value is put down in that location. Once all the values have been added to the occupancy grid map it is published out for viewing on rviz

Output:

The output statistics we define in compare_results() function shows our results in terms of number. It outputs the spaces that have been explored and the spaces that haven't, the cut and uncut spaces, and the percentage of the grass that is been cut. We display the real-time result simulation on the RViz and show the cut area result on RViz by publishing another topic: Map2.

In map1 when the obstacle is detect, RViz will mark it as a black dot using the draw_point() function, otherwise the empty spaces are marked as white using the draw_line() function.

In map2 the cut spaces are shown as a darker color when the robot moves while the uncut grass is shown as a lighter color.

Note: we had some integration issues with this part of the code and to run it we need to change the code to display on 1 publisher at a time. With this issue we need to run the program twice changing the source code each time to display the results properly.

Below is the pseudo code for the looping algorithm that controls the main movement of our robot.

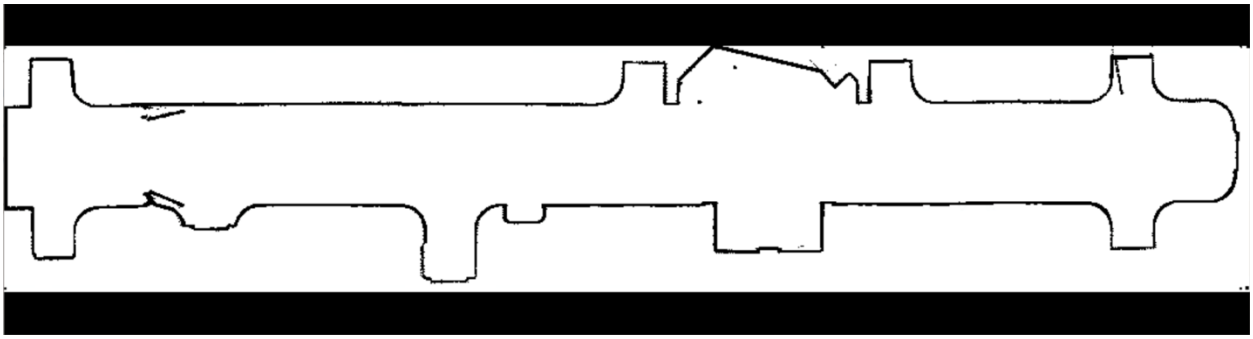
Pseudo Code:

```
robot.floor_cover(i)
function floor_cover(distance)
if (any front sensor too close to objects)
        avoid_obstacle()
else{
        if (too far left from distance i)
                veer to the right
        else (too far right from distance i)
                veer to the left
}
end function
```

Analysis of results:

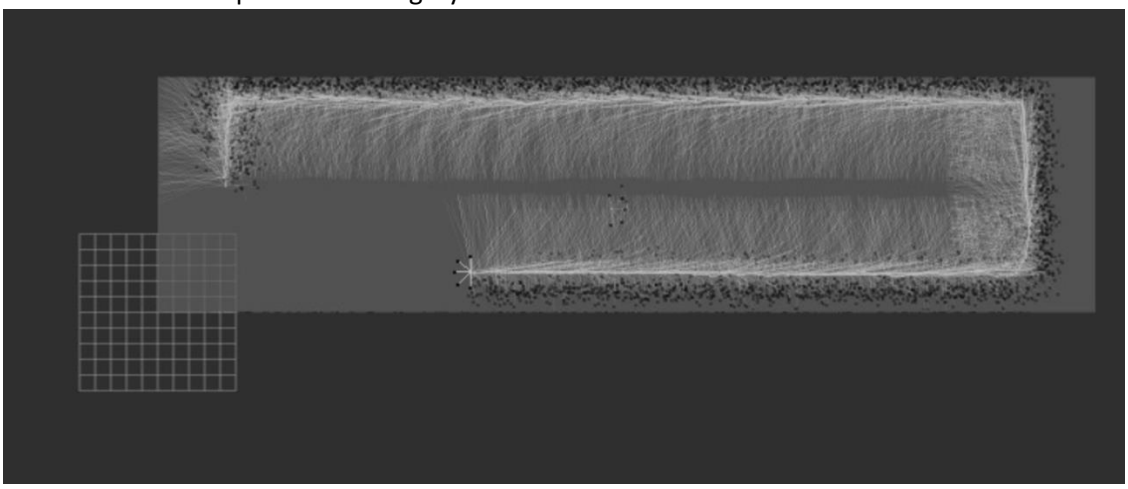
For the original map from PS6 our robot cut approximately 60% grass. The result is also limited by the sensor's detection range and make the cutting result not ideal. We can only sense out to 5 meters from the robot and we keep the robot bound to the wall, so that anything in the center of the map beyond 5 meters will not get cut. This leaves a large uncut center of the map. Also, the areas right up against the wall are to close for the robot to safely travel against and are skipped. Below are some pictures of the results.

The first picture is just the map from PS6.



The second picture shows the results of our visualization of the grass being cut in rviz. The dark area is cut grass, light area is uncut grass, and grey is unknown area.

This third image shows the robot mapping out the occupancy grid map as it is cutting the lawn. In this instance it is testing it on an additional map, but it is a similar image for any map. black represents obstacles white is open area. And grey is unknown.



Discussion:

When we integrated my work with Wei's work the compare function that displayed the exact percentage of our cut grass stopped working. The map still contains the correct data and rviz shows what it's supposed to so it would have been nice to figure out what went wrong but this code was not super important as we have the rviz visualization and ran out of time to fix that. We lost the number display but gained a graphical representation.

If I had more time to work on this project I would want to work on utilizing the map for more then just displaying results. If we oriented the robot to drive towards uncut grass on the map rather then doing the algorithm based on just a defined set of movements the results will get much better.

Partnership:

I wrote pretty much all the code we have that was not already written by the TA and my partner worked on adding in the bit that handled the second portion of RVIZ visualization.

My focus was any algorithm heavy components of the project.

Conclusion:

In summary, we achieved what we set out to do to a degree but have a lot of room for improving. On most maps we tested we average a cut ratio of about 60% of the map.

The next steps for the project are to move are algorithm design away from using just a predefined movement to also incorporate moving are robot towards areas of the map that we know we need to cut.

Optional closing response to question:

If I were to redesign the course, I would just change 2 things. The most important factors that lead to my having a bad experience with this course were the fact it was only offered at 8am and the learn it yourself approach to ROS. My commute to get to campus on top of me not being a morning person made arriving to the lectures on time a struggle. This isn't the ideal class for that to be an issue as it is already hard enough in its own right. I also felt that I would personally do a lot better in the class if at the start we had some lectures on getting introduced rather then setting us out into the world to go through ROS tutorials on our own. I think at the start at least personally, I do better with lectures with tutorials as purely supplemental rather then the only thing I have to go on.

To run code, look at the instructions in the readme that is included with the code.