WORCESTER POLYTECHNIC INSTITUTE
ROBOTICS ENGINEERING PROGRAM

Lab 4 – SLAM

SUBMITTED BY
Kevin Chin
Sukriti Kushwaha
Chad Nguyen

Date Submitted: May 1, 2024
Date Completed: May 1, 2024
Course Instructor: Professor Rosendo
Lab Section: RBE 3002 D’24

Introduction

The purpose of this lab was to combine all the knowledge gathered from previous labs and implement it in a system that completed the following tasks:

1. Localize the robot in the world
2. Drive around the map while avoiding obstacles
3. Find interesting areas to expand the map
4. Find optimal path to an identified point
5. Recognize when the map is complete
6. Navigate to a given position in the map

SLAM is a common method used to build a map and localize the robot simultaneously. To implement SLAM in our project, we utilized the gmapping and amcl ROS packages. On a high level, we used the robot's odometry and laser scan information to generate a map of the given field and identify the pose of the robot. Gmapping is an algorithm that takes laser readings as input and generates an occupancy grid. Creating a map encompassed calculating c-space, and finding frontiers from the gmapping results. We then implemented the A* algorithm to find the most optimal path to a chosen frontier. Localization consisted of matching poses between global and local frames from the robot's odometry information and the amcl estimates. amcl has a collection of localization algorithms from which we chose particle filtering for this lab. Particle filtering was used because it can handle non-gaussian uncertainty probabilities.

Methodology

The first step to complete phase 1 of the project was to tune gmapping. The parameters of gmapping can be tuned and changed in the gmapping_params.yaml file. The main field that needed to be tuned was the "delta" or the resolution of the grid size. The benefit of increasing the resolution is better path planning since c-space can be better tuned and gives more grid squares to position the robot on the map. However, the main issue with increasing resolution is that it will drastically increase the time it takes to calculate the path since there are so many more squares to check. Another key set of parameters to tune were the "xmin", "xmax", "ymin", "ymax" values. These four parameters control the size of the map generated by gmapping. Tuning this greatly helps reduce computational time since there are so many less squares to check on the map. However, if it is cut too small, then the map won't fit anymore making mapping the whole map impossible.

The next step is to find frontiers to explore. Since frontiers are unknown areas that are between walls, our frontier finding filter takes advantage of the fact that all frontiers between walls will be encompassed by unknown values (values of –1 in the occupancy grid) next to known empty cells (values of 0 in the occupancy grid). After finding all the unknown cells that could be frontiers, the next step is to group them together to check how large the frontier is and if it is worth exploring. We found that frontier groups that are less than 8 squares wide are not worth checking. This is since they are too small to drive to and eliminate frontiers found by stray readings that go past the actual wall. Frontiers are grouped together by finding unknowns next to a wall and then doing a BFS for more frontiers that are next to them using group_of_eight(). After grouping the frontiers together, the centroid is found as the middle cell added to the group of frontiers. We choose centroids by picking the ones closest to the robot via Manhattan distance.

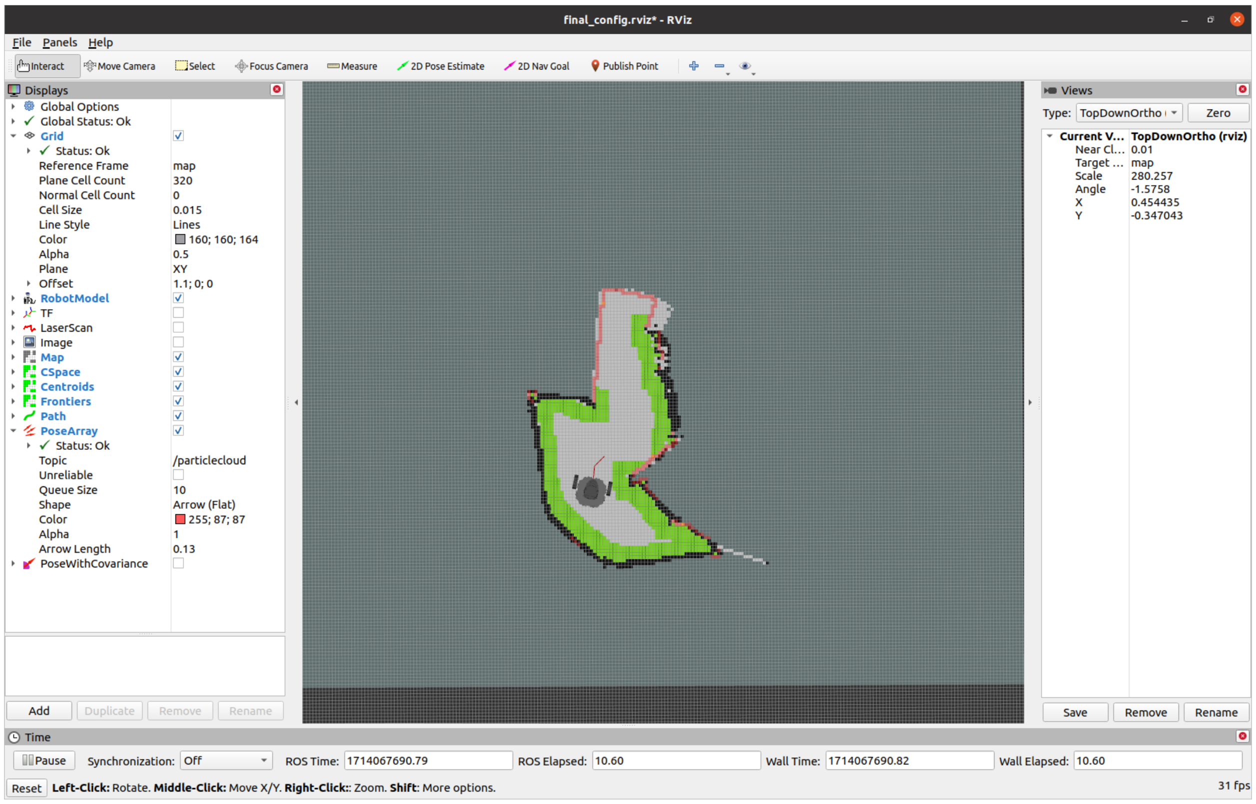
To calculate the path to the centroid of the nearest frontier, our team used a modified version of the lab 3 a-star algorithm. The first major modification made to a-star was updating the occupancy grid. Since the map being used is not always complete, we set the frontiers to be 0 so they are walkable and the other unknown cells to 1 so they appear as walls in the calculation. The next modification made was to switch from Euclidian distance to Manhattan distance for the greedy search heuristic to incentivize driving with less diagonals. This is because Manhattan distance penalizes cells that are more diagonal from the goal space than Euclidian distance. Another heuristic added to the a-star algorithm was penalizing the robot for being close to c-space. To accomplish this, we made cspace_in_8(), cspace_in_24(), cspace_in_48(), cspace_in_80(), and cspace_in_120(). These functions checked if there was a cell around them in varying sizes that contained cspace and returned true. This was utilized in the a-star algorithm to add increasing penalty the closer the robot was to cspace. This created a path that was more centered to the map when driving since that would be the farthest from cspace and create the least amount of penalty.

To know when the map is completely explored, we check all frontier centroids that have not been visited yet and if there are no paths found, then the map must be completed. The reason there could be frontier centroids with a closed map is because of the stray readings on the outside of the map space. This way, they can be ignored, and the map knows when it is done. After flagging that the map is done, phase 2 begins and the robot uses a-star once again to path plan to the start of the maze. When it returns to its location, the robot flags that it is done. When the robot is done, map saver is run in terminal to save the map for phase 3 localization.

Once the map is saved as a .pgm and .yaml file, we are clear to move onto Phase 3 of the project. This final phase starts with us killing our first launch file since gmapping can’t be run alongside AMCL localization. After that, we then move the robot to a random location in the field and run our “phase3.launch” file. The new launch file starts a map server using our saved map from Phases 1 and 2, opens up RViz so we can see the map, launches AMCL with our set parameters, and launches the lab 2 (lab2.py), lab3 (path_planner.py, lab3_listener.py), and localization.py nodes. The mentioned AMCL parameters are set so that the filter will only update upon moving 1 centimeter translationally and/or 0.05 radians angularly, will use our map topic for localization, and has a minimum of 50 particles and a maximum of 2500 particles.

The only new node we created for this phase is localization.py. Thise file works by having the robot spin at a set speed of 0.35 rad/s while listening to the given “global_localization” service and the “PoseWithCovarianceStamped” topic. The covariances errors of our x, y, and theta values of the particles are then measured as the robot spins until the error is 0.005 or below. Once the error meets our threshold, the robot knows where it is in the map and therefore stops spinning and waits for a position to drive to. Giving a position and driving to it is the same as in lab 3, we use the “lab3_listener.py” service to take in a 2d nav goal in RViz, then use “path_planner.py” to generate a path using A*, and finally give that path to “lab2.py” to drive to the goal. One last thing to note is that with gmapping/SLAM off, we use our old method of updating odometry for Phase 3.

Results

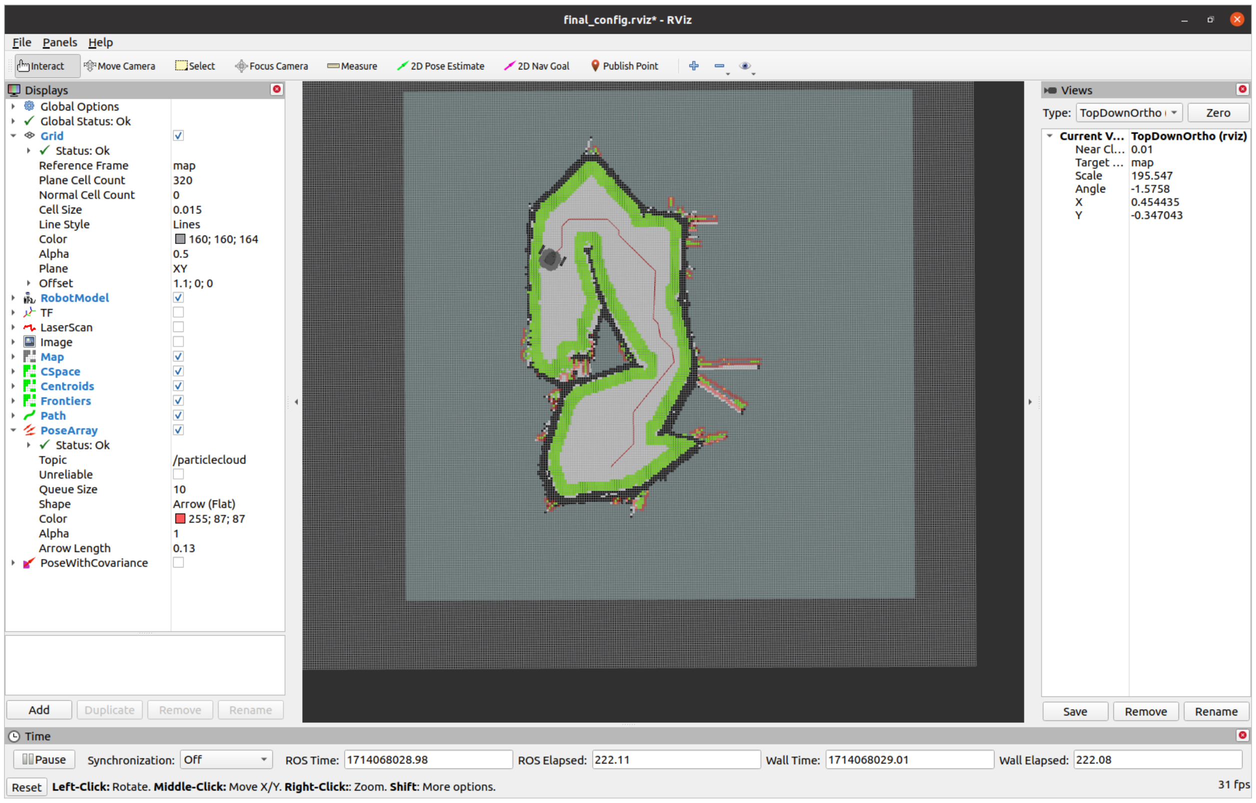


```
Global Options
Global Status: Ok
Grid                        ✓
  Status: Ok
  Reference Frame           map
  Plane Cell Count          320
  Normal Cell Count         0
  Cell Size                 0.015
  Line Style                Lines
  Color                     160; 160; 164
  Alpha                     0.5
  Plane                     XY
  Offset                    1.1; 0; 0
RobotModel                  ✓
TF
LaserScan
Image
Map                         ✓
CSpace                      ✓
Centroids                   ✓
Frontiers                   ✓
Path                        ✓
PoseArray                   ✓
  Status: Ok
  Topic                     /particlecloud
  Unreliable
  Queue Size                10
  Shape                     Arrow (Flat)
  Color                     255; 87; 87
  Alpha                     1
  Arrow Length              0.13
PoseWithCovariance

Type: TopDownOrtho
Current V...   TopDownOrtho (rviz)
  Near Cl...   0.01
  Target ...   map
  Scale        280.257
  Angle        -1.5758
  X            0.454435
  Y            -0.347043

Synchronization: Off   ROS Time: 1714067690.79   ROS Elapsed: 10.60   Wall Time: 1714067690.82   Wall Elapsed: 10.60
```

Figure 1: RViz map at the start of Phase 1 showing CSpace, frontiers, and a path.

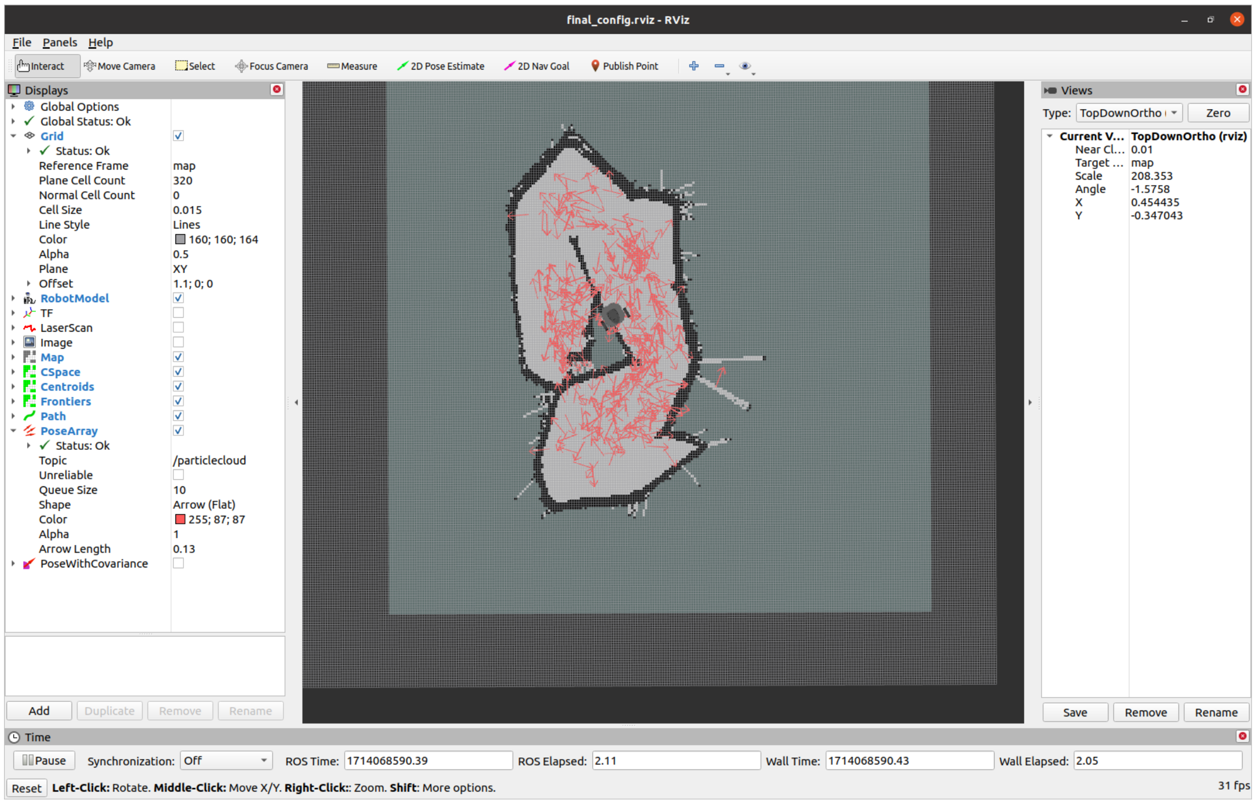


```
Global Options
Global Status: Ok
Grid                          ✓
  Status: Ok
  Reference Frame             map
  Plane Cell Count            320
  Normal Cell Count           0
  Cell Size                   0.015
  Line Style                  Lines
  Color                       160; 160; 164
  Alpha                       0.5
  Plane                       XY
  Offset                      1.1; 0; 0
RobotModel                    ✓
TF
LaserScan
Image
Map                           ✓
CSpace                        ✓
Centroids                     ✓
Frontiers                     ✓
Path                          ✓
PoseArray                     ✓
  Status: Ok
  Topic                       /particlecloud
  Unreliable
  Queue Size                  10
  Shape                       Arrow (Flat)
  Color                       255; 87; 87
  Alpha                       1
  Arrow Length                0.13
PoseWithCovariance
```

```
Type: TopDownOrtho         Zero
Current V...   TopDownOrtho (rviz)
  Near Cl...   0.01
  Target ...   map
  Scale        195.547
  Angle        -1.5758
  X            0.454435
  Y            -0.347043
```

```
Synchronization: Off   ROS Time: 1714068028.98   ROS Elapsed: 222.11   Wall Time: 1714068029.01   Wall Elapsed: 222.08
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Zoom. Shift: More options.   31 fps
```

Figure 2: RViz map during Phase 2 which shows a completed map, CSpace, and the path back to initial position.

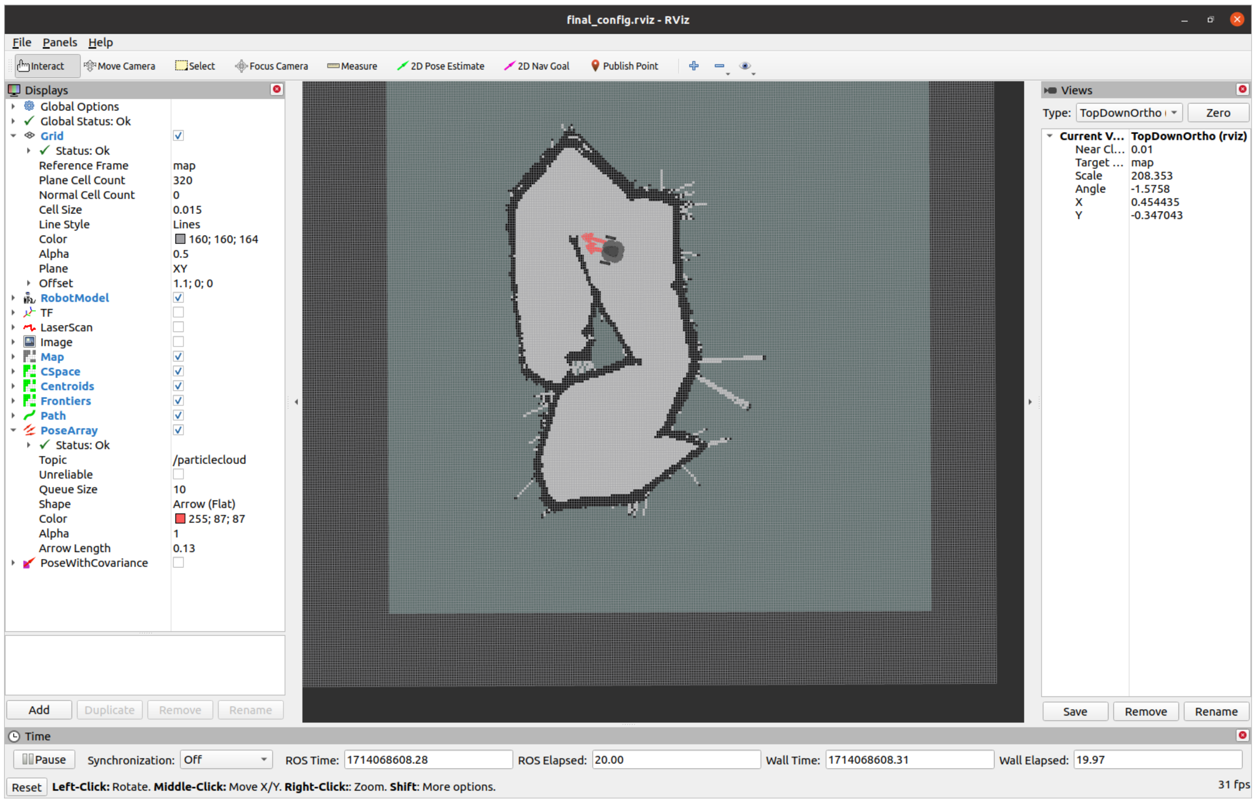


```
Global Options
Global Status: Ok
Grid                          ✓
  Status: Ok
  Reference Frame             map
  Plane Cell Count            320
  Normal Cell Count           0
  Cell Size                   0.015
  Line Style                  Lines
  Color                       160; 160; 164
  Alpha                       0.5
  Plane                       XY
  Offset                      1.1; 0; 0
RobotModel                    ✓
TF
LaserScan
Image
Map                           ✓
CSpace                        ✓
Centroids                     ✓
Frontiers                     ✓
Path                          ✓
PoseArray                     ✓
  Status: Ok
  Topic                       /particlecloud
  Unreliable
  Queue Size                  10
  Shape                       Arrow (Flat)
  Color                       255; 87; 87
  Alpha                       1
  Arrow Length                0.13
PoseWithCovariance
```

```
Type: TopDownOrtho         Zero
Current V...   TopDownOrtho (rviz)
  Near Cl...   0.01
  Target ...   map
  Scale        208.353
  Angle        -1.5758
  X            0.454435
  Y            -0.347043
```

```
Synchronization: Off   ROS Time: 1714068590.39   ROS Elapsed: 2.11   Wall Time: 1714068590.43   Wall Elapsed: 2.05
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click:: Zoom. Shift: More options.   31 fps
```

Figure 3: RViz map at start of Phase 3 showing particles all across the map trying to localize.


final_config.rviz - RViz
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Displays
Views
Add Duplicate Remove Rename
Save Remove Rename
Time
Pause
Reset

Global Options	
Global Status: Ok	
Grid	✓
Status: Ok	
Reference Frame	map
Plane Cell Count	320
Normal Cell Count	0
Cell Size	0.015
Line Style	Lines
Color	160; 160; 164
Alpha	0.5
Plane	XY
Offset	1.1; 0; 0
RobotModel	✓
TF	
LaserScan	
Image	
Map	✓
CSpace	✓
Centroids	✓
Frontiers	✓
Path	✓
PoseArray	✓
Status: Ok	
Topic	/particlecloud
Unreliable	
Queue Size	10
Shape	Arrow (Flat)
Color	255; 87; 87
Alpha	1
Arrow Length	0.13
PoseWithCovariance	

Type: TopDownOrtho Zero

Current V...	TopDownOrtho (rviz)
Near Cl...	0.01
Target ...	map
Scale	208.353
Angle	-1.5758
X	0.454435
Y	-0.347043

Synchronization: Off ROS Time: 1714068608.28 ROS Elapsed: 20.00 Wall Time: 1714068608.31 Wall Elapsed: 19.97

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click:**: Zoom. **Shift**: More options. 31 fps

Figure 4: Rviz map during Phase 3 showing all particles clustered together witht the robot successfully localized.

Discussion

The goals of the lab were to create and navigate a map of an unknown environment given observations by the robot's sensors. For navigation, we found that utilizing Manhattan distance for the A* heuerstic function and penalizing cells close to c-space helped generate routes that were close to the center of the path. There are several ways we could improve the planning and actuation of the trajectory taken by our robot. For instance, noisy gmapping values influenced the calculations for the optimal a-star path. Noisiness can lead to areas that are not walls being incorrectly identified as walls and vice versa. We found that testing our robot when there were not many other teams in lab helped minimize the noisiness in readings. Additionally, we used drive-turn-drive to move our robot to a given location which took time to plan and execute. In retrospect, implementing either pure pursuit algorithm or PID control could help speed up the driving time. Symmetrical parts of the map also caused localization to take more time since the robot may be confused between identifying similar areas. We found that tuning the particle filter parameters, including the number of particles, helped with decreasing the uncertainty between similar looking areas.

Conclusion:

Ultimately, we utilized the gmapping and amcl ROS packages to create and navigate a map of an unknown environment given observations by the robot’s sensors. We created a map by tuning gmapping parameters and grouping frontiers by finding unknown cells next to known cells using BFS. To explore and update the map, we calculated the distance to the closest centroid of a frontier and then generated an optimal path using A*. Once there were no unknown cells left, this meant the entire map was generated, and we saved it to use for localization. Once the robot is placed in an unknown area on the map, it rotates in place until the covariances errors of the x, y, and theta values of the particles in the particle filter are 0.005 or below. When the robot is localized, it calculates a plan to a goal point using A* and drives to it.

Appendix A: Authorship

Section	Author
Introduction	Sukriti Kushwaha
Methodology	Chad Nguyen and Kevin Chin
Results	Everyone
Discussion	Sukriti Kushwaha
Conclusion	Sukriti Kushwaha