A1: Mobile Robot Assignment

Alexander Berman, Justin Kaput, Zheng Hao Tan
EECS 467 Winter 2015 - Team 6 A1 Report
{anberman, jkaput, tanzhao}@umich.edu

TASK 1: MAPPING

For the mapping portion of this project, we made new file called *grid_map.cpp* under the *src/maebot* directory, which handles the display of the mapping process. Grey portions of the map remain unexplored by the MAEbot, whereas the white portions of the map represent a high-confidence open/free space where the MAEbot can freely move about. The black portions of the map represent obstacles (high-confidence occupied) that the MAEbot cannot move to without hitting them.

We ran our *grid_map.cpp* for both the *empty_environment.log* and *figure_eight.log* files provided to us to produce 2 final maps for this report. These two log files are saved in our *ground_truth* directory.

Our program uses the sensor trace data to create the map of the environment by incrementing or decrementing the log-odds-occupancy value in each grid cell. One of the challenges that we faced earlier on was how slow the map was rendering because we were going for the vx_box method. In our case, we were generating several small Vx boxes to build the map. This caused the map to take a really long time to render, so we decided to try using vxo_image_from_u8. This made it a lot faster to render the background and each of the individual pixels.

Empty Environment:
Fig. 1 - 6 are 6 snapshots of our MAEbot mapping the unexplored empty environment at 6 different locations. The *empty_environment.log* file was provided to us.

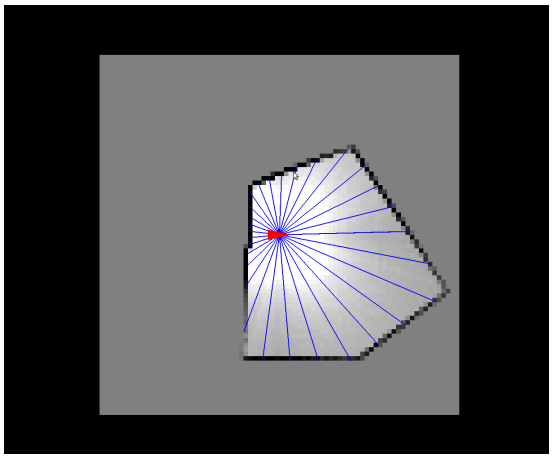

Fig. 1: MAEbot at the starting position

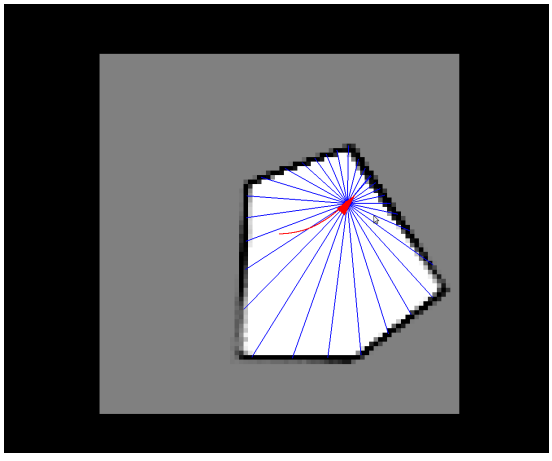

Fig. 2: MAEbot at the first corner

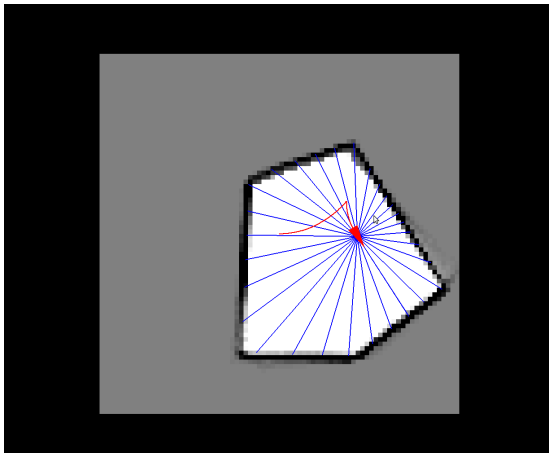

Fig. 3: MAEbot after making its first turn

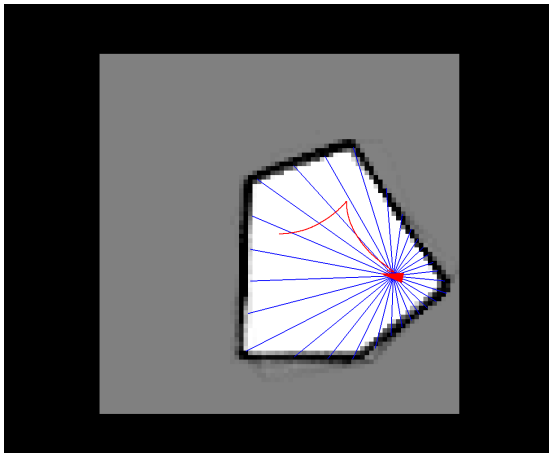

Fig. 4: MAEbot at the second corner

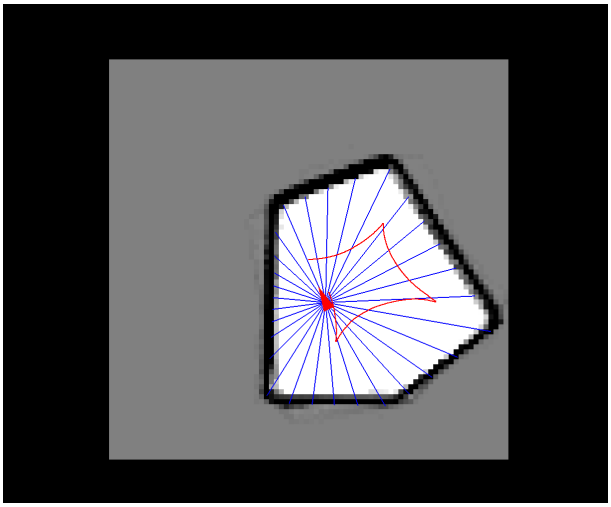

Fig. 5: MAEbot after making the third corner

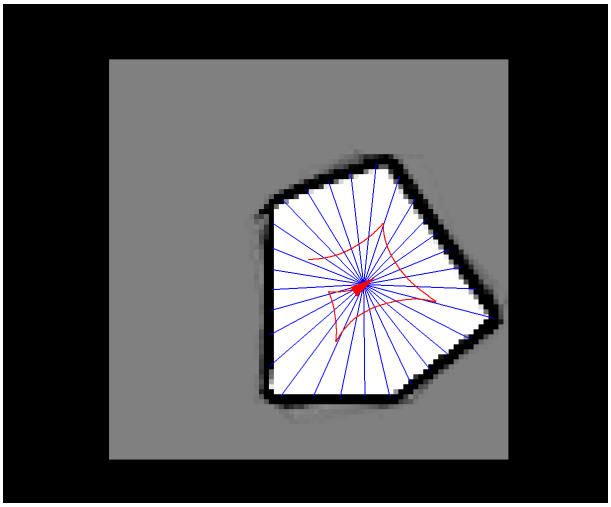

Fig. 6: MAEbot at the final position

Figure Eight:
Fig. 7 - 12 are 6 snapshots of our MAEbot mapping the unexplored figure eight environment at 6 different locations.

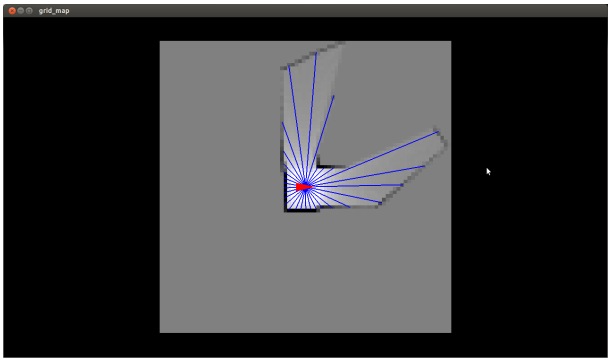

Fig. 7: MAEbot at the starting position

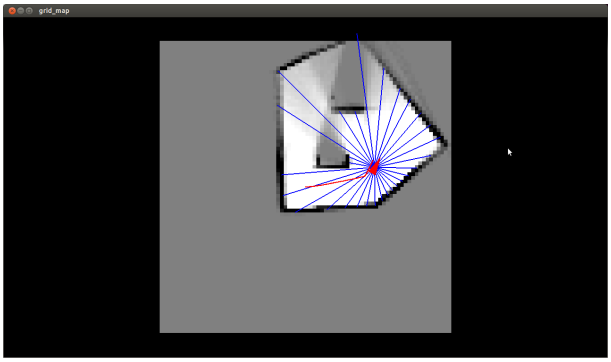

Fig. 8: MAEbot at the first corner

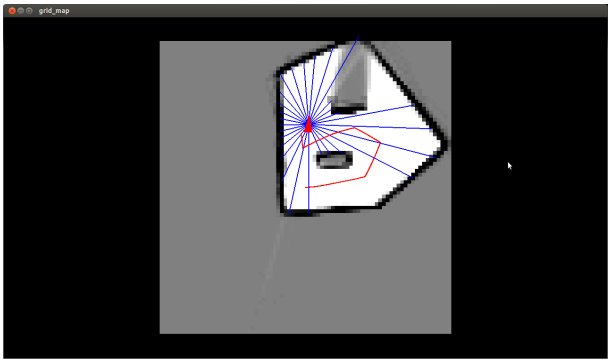

Fig. 9: MAEbot after making its second turn

The red lines drawn on the map shows the path that the MAEbot took based on odometry measurements given by $maebot\_pose\_t$. The blue lines shows the laser scans taken at a given time. We decided to draw some (not all) of the laser scans as this will clobber our map too much. We also used vxo_robot to draw the MAEbot in red because this gives us a better idea of the MAEbot's pose in the map.

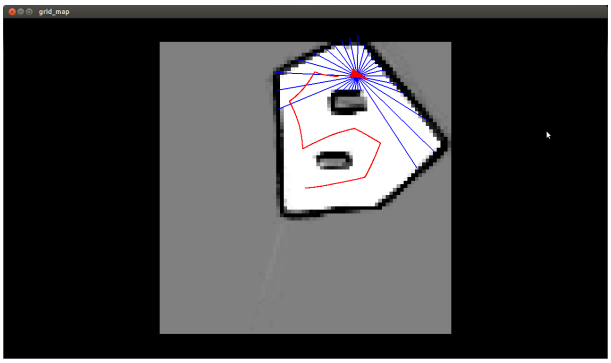

Fig. 10: MAEbot at the fifth corner

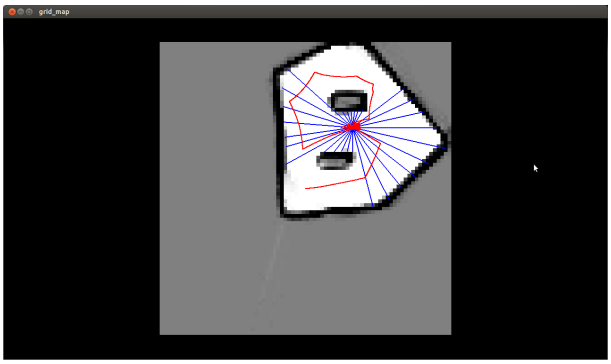

Fig. 11: MAEbot at the center of the Figure 8

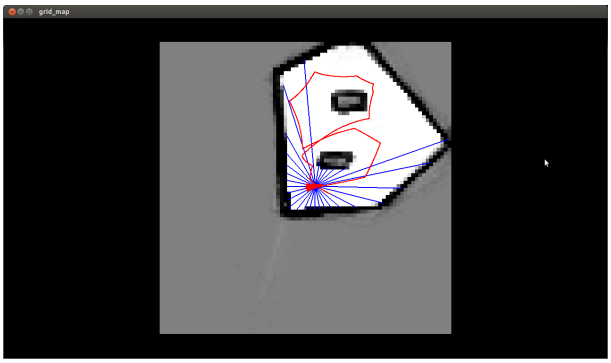

Fig. 12: MAEbot at the final position

TASK 2: LOCALIZATION

Empty Environment:

Fig. 13 - 18 are 6 snapshots of our MAEbot mapping the unexplored environment at 6 different locations.

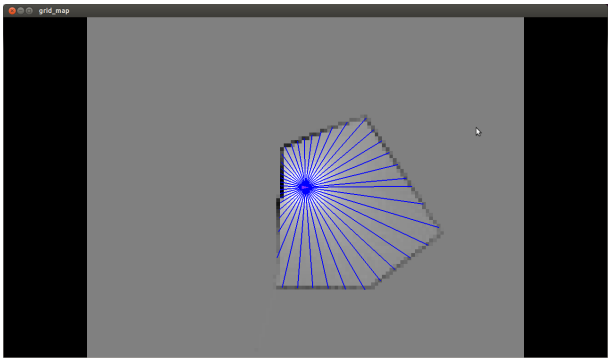

Fig. 13: MAEbot at the starting position

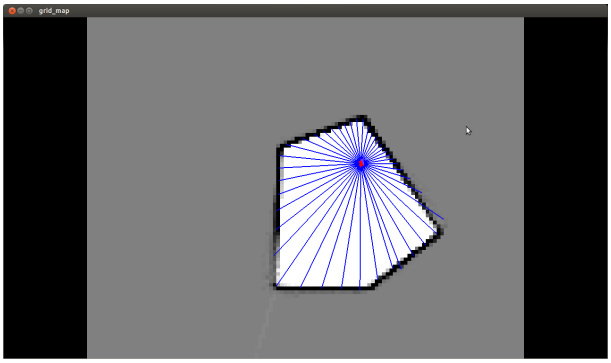

Fig. 14: MAEbot at the first corner

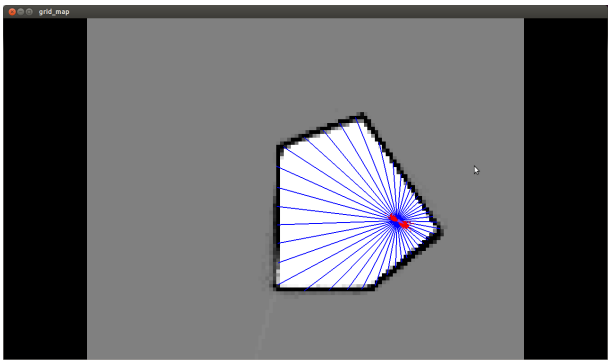

Fig. 15: MAEbot at the second corner

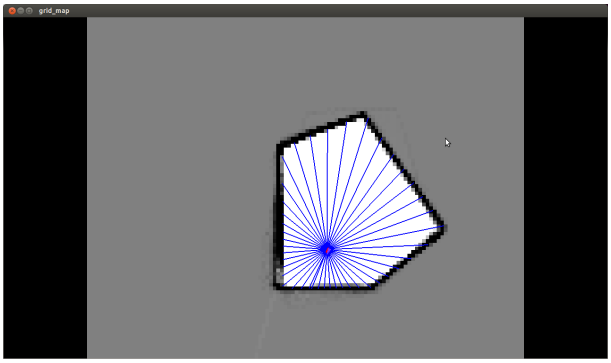

Fig. 16: MAEbot at the third corner

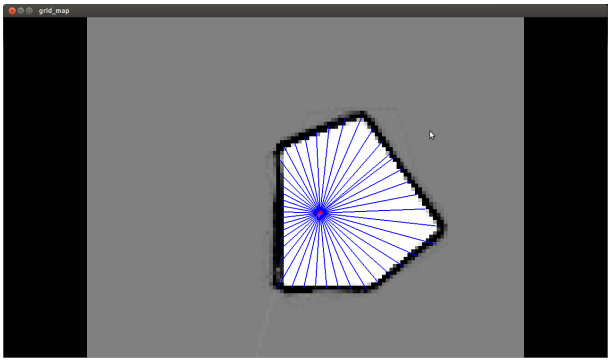

Fig. 17: MAEbot at the fourth corner

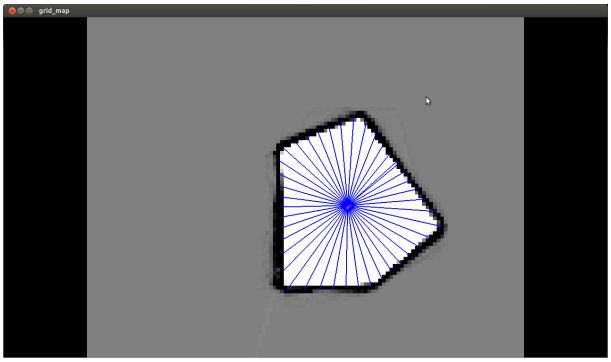

Fig. 18: MAEbot at the final position

Figure Eight:
Fig. 19 - 24 are 6 snapshots of our MAEbot mapping the unexplored environment at 6 different locations.

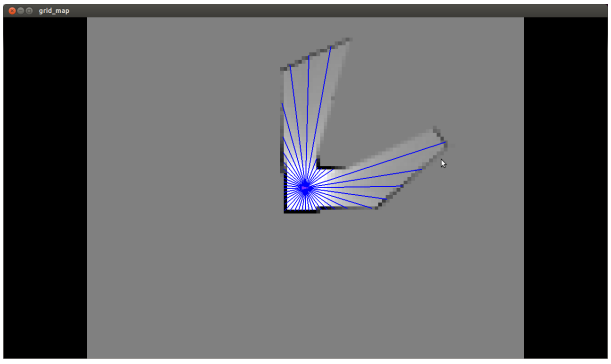

Fig. 19: MAEbot at the starting position

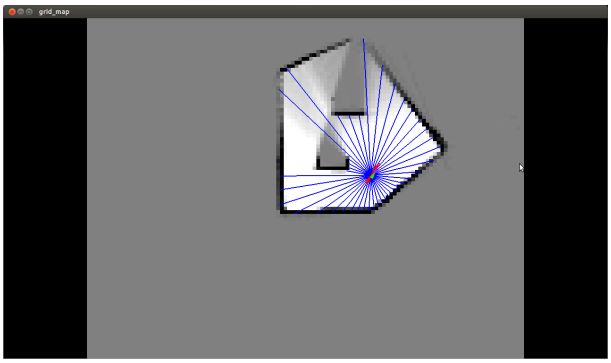

Fig. 20: MAEbot at the first corner

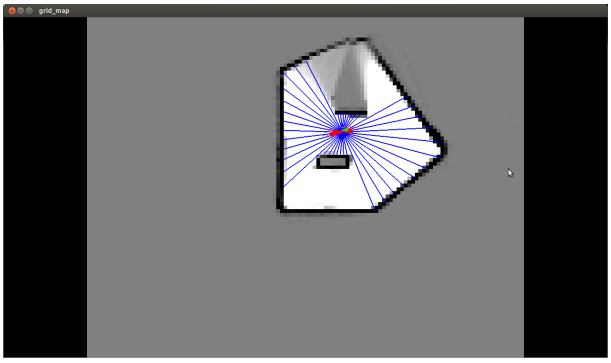

Fig. 21: MAEbot after making its first turn

We have 1000 samples running for this task, and all these particles are shown in red in the pictures above. We also ran a thread to generate each sample. This means that we have

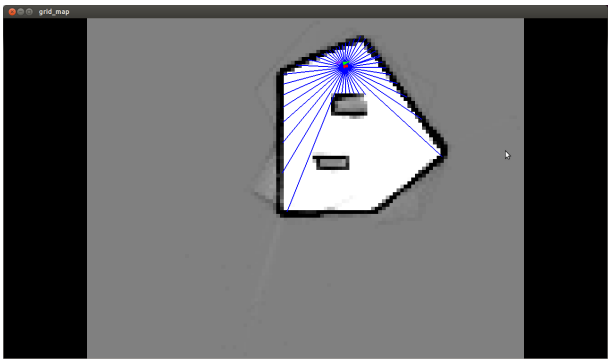

Fig. 22: MAEbot at the fifth corner

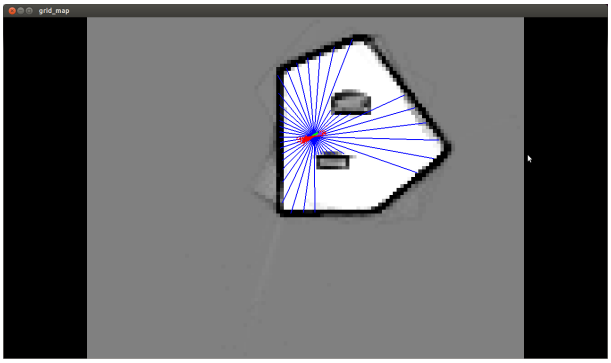

Fig. 23: MAEbot before making the last corner

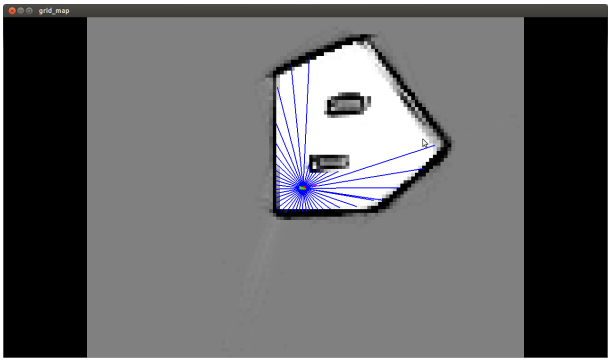

Fig. 24: MAEbot at the final position

at least 1000 threads running at any given time. However, this did not slow us down when it came to calculating and rendering the samples.

One of the bugs we got was a spiral effect for the poses generated. The samples formed a red, spiral shape as the MAEbot moved around the map. This happened because we were not generating the random distributions correctly. In the function where we were supposed to

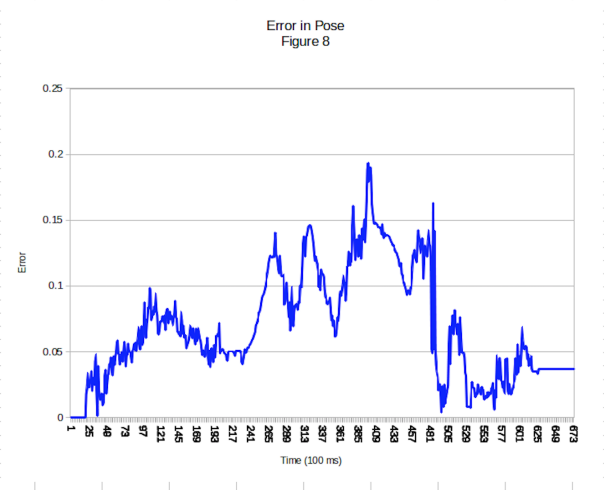

Fig. 25: Error between the Estimated Trajectory and the Ground Truth Trajectory

generate the samples, we were initializing the generator every time the function ran. This was a problem because since the C++ 11 generator function has a seed value to it, every run of the function will return the same value. This will cause the generated values to be less random than we wanted it to be.

We also had problems with rotations and angles for the MAEbot. We realized that we weren't wrapping angles to π by using the $wrap\_to\_pi$ function provided in the $src/math$ directory. This will cause the MAEbot to spin in different directions than what we wanted it to.

Task 3: Simultaneous Localization and Mapping (SLAM)

One of the challenges that we face for this project was getting the local computer to send LCM messages over to the MAEbot. This was a known bug on LCM, which we can only receive LCM messages from the MAEbot but not able to send messages to it. Since we were sending these messages over the network, other teams had trouble when they received our messages. We also tried setting $lcm\_create's$ TTL to be the 1 on both the local computer and MAEbot, but we were spamming and putting a significant load on the network. With Collin's help, we were able to send and receive data from both sides by running $bot - lcm - tunnel$. This method will also prevent other groups from receiving our LCM messages.

Since we had to run most of the computations on the MAEbot, and the MAEbot has limited computing power, running a GUI on it will slow it down by a lot. Hence, we made a separate file called $maebot\_grid\_map.cpp$, which is basically $grid\_map.cpp$ without Vx. This file will then run way faster on the MAEbot.

Path Planning:

For the path finding algorithm, we planned on implementing several algorithms and tweak them based on the environment to give us maximum speed and efficiency in exploring the map. Some of the options that we considered were breadth-first search, depth-first search and probabilistic road map.

We eventually implemented both breadth-first search and depth-first search algorithms (A*) to find a suitable path to a frontier for the MAEbot to move along. We soon realized that a depth-first search wasn't complete, so sometimes the MAEbot will just stall and claim that there is nowhere to move from a certain point. We reimplemented a breadth-first search algorithm to guarantee completeness. Since the world is partially observable, an optimal solution might not be necessary as the path will also change in time to ensure that the MAEbot has fully explored the area.

For this part of the project, we created a new lcmtype called "MAEBOT_WAYPOINT", which will contain coordinates for where the MAEbot should move to and a boolean

that says done if the MAEbot is done with its exploration. We also created a file called $project1.cpp$ under $src/maebot$. This file will handle how the MAEbot will drive under the given conditions such as MAEBOT_WAYPOINT.

When we got our A* path, we then removed all waypoints that could be reached without crossing obstacles except for the last waypoint that could be reached. We did this iteratively until we looked at all points on our original path.

We also had several bugs in getting the MAEbot to move along the path. We ran into some problems with the rotations of the MAEbot as we used tangent to calculate how the MAEbot is going to rotate into its specified path. This was giving us problems as it would only turn a maximum of 90 degrees and we would run into problems when it had to rotate more than that. We were calculating our desired angles in global coordinates and used them as local (MAEbot) coordinates.

Aerial view of the unexplored map: Fig. 26 shows an ariel image of the map to be explored.

Fig. 26: Aerial view of the environment that the MAEbot is going to explore

Fig. 27 - 30 are 6 snapshots of our MAEbot mapping the unexplored environment at 6 different locations.

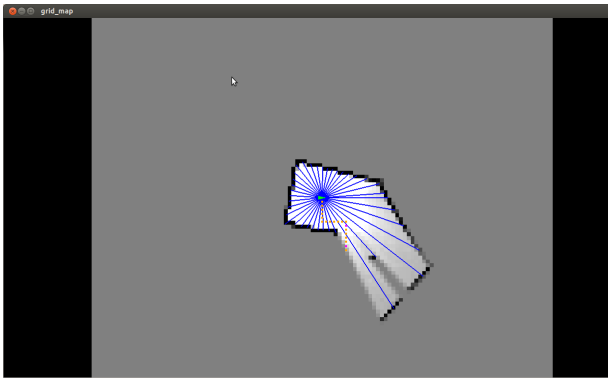

Fig. 27: MAEbot at the starting position

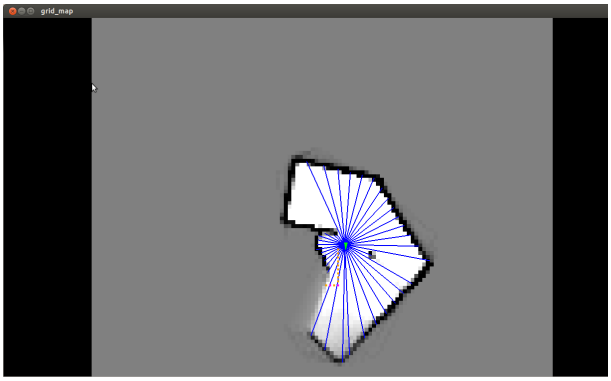

Fig. 28: MAEbot at the first corner

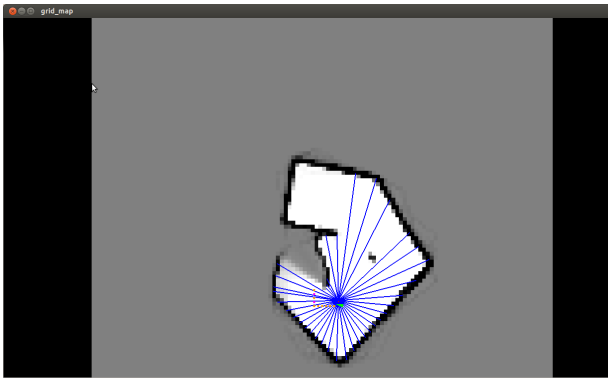

Fig. 29: MAEbot after making its first turn

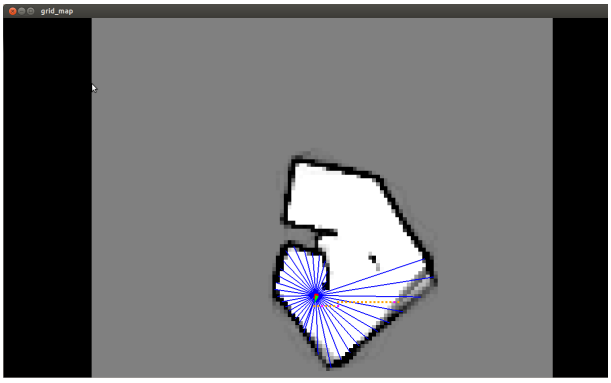

Fig. 30: MAEbot at the second corner

Aerial view of the unexplored map:
Fig. 31 shows an ariel image of the another map to be explored.

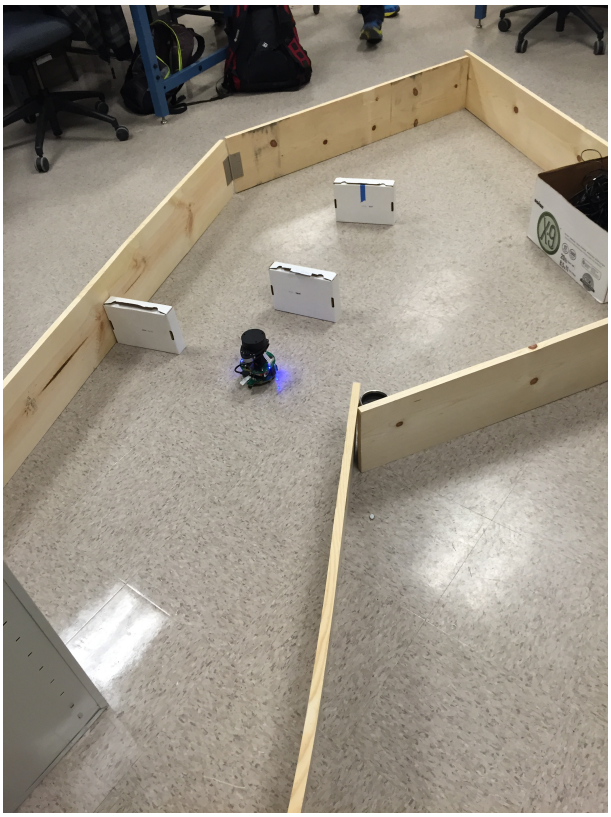

Fig. 31: Aerial view of the environment that the MAEbot is going to explore

Fig. 32 - 36 are 6 snapshots of our MAEbot mapping the unexplored environment at 6 different locations.

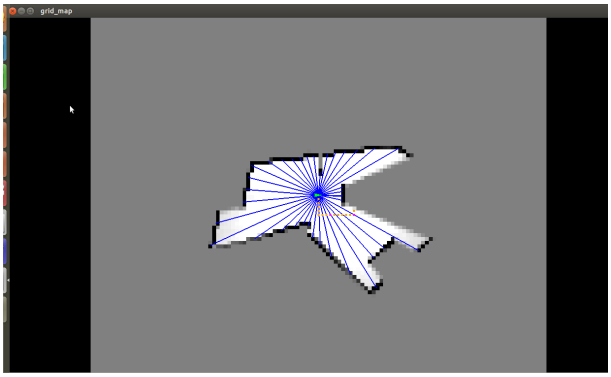

Fig. 32: MAEbot at the starting position

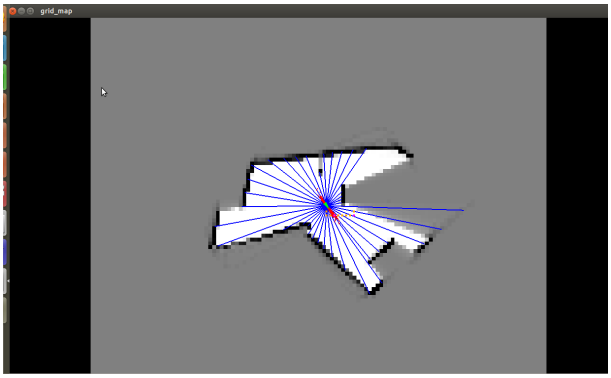

Fig. 33: MAEbot at the first corner

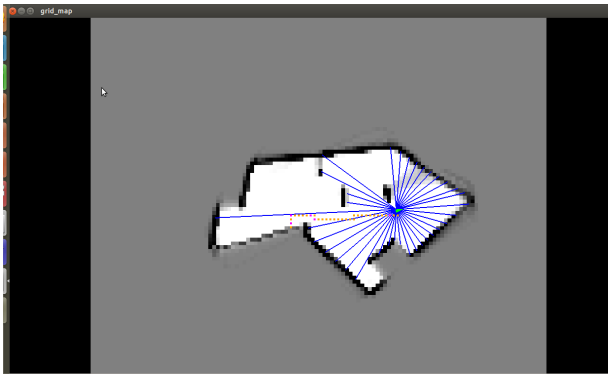

Fig. 34: MAEbot after making its first turn

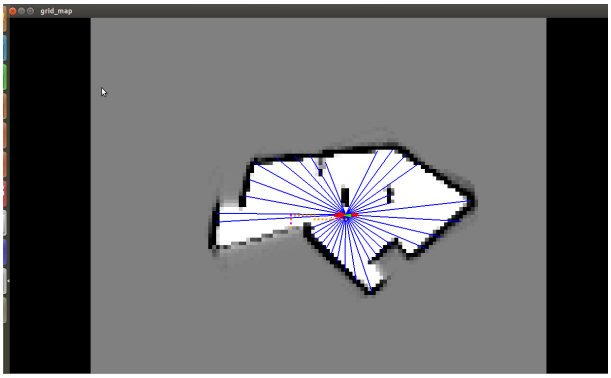

Fig. 35: MAEbot at the second corner

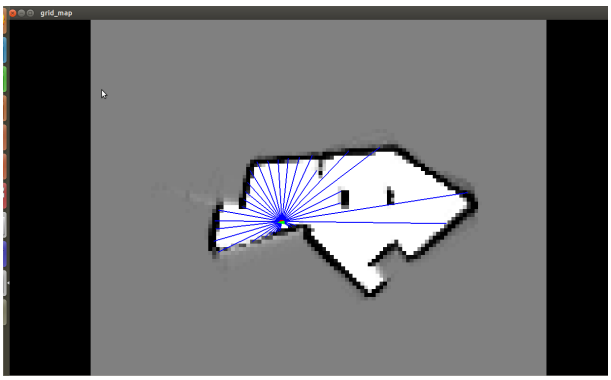

Fig. 36: MAEbot after making the third corner

Aerial view of the unexplored map: Fig. 37 shows an ariel image of the another map to be explored.

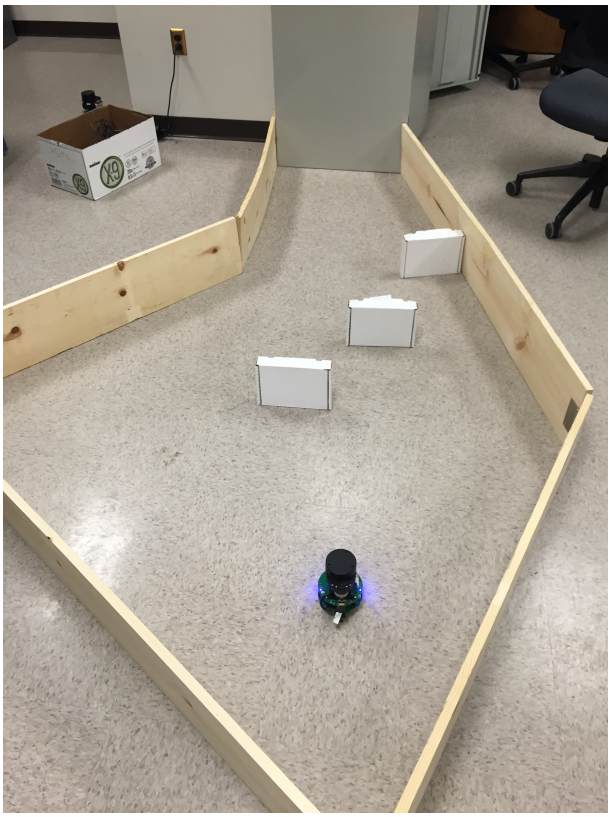

Fig. 37: Aerial view of the environment that the MAEbot is going to explore

Fig. 38 - 43 are 6 snapshots of our MAEbot mapping the unexplored environment at 6 different locations.

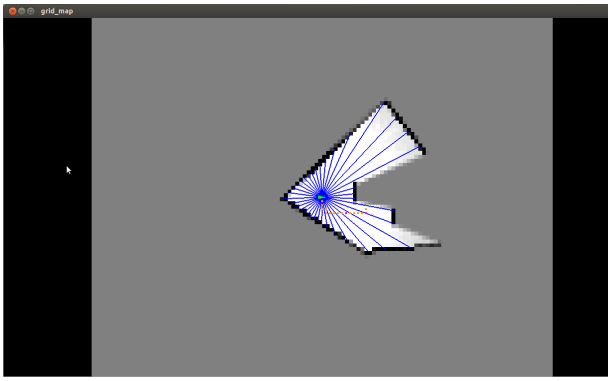

Fig. 38: MAEbot at the starting position

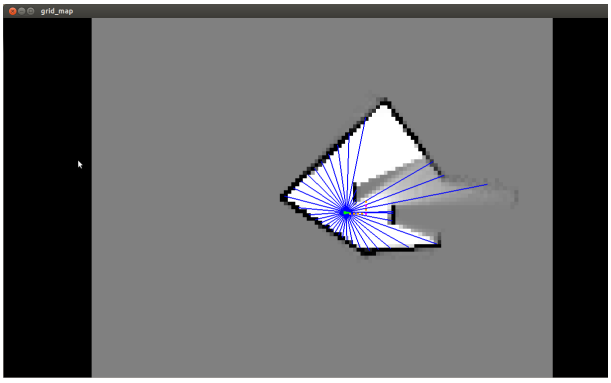

Fig. 39: MAEbot at the first corner

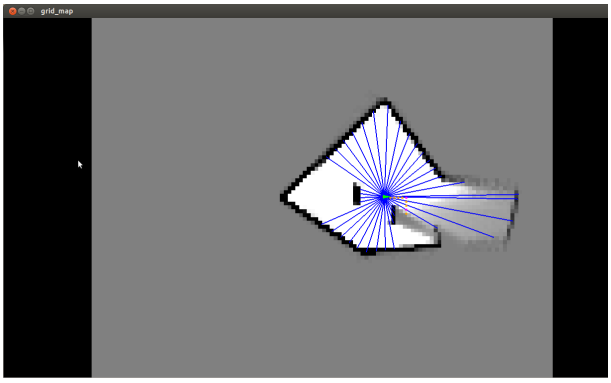

Fig. 40: MAEbot after making its first turn

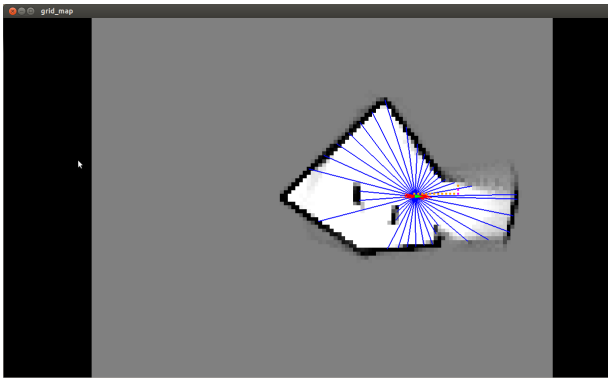

Fig. 41: MAEbot at the second corner

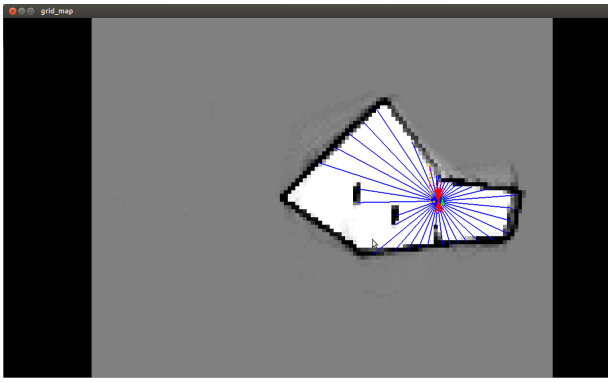

Fig. 42: MAEbot after making the third corner

Demonstration/Competition:

We demo-ed our MAEbot on Wednesday Feb 18, 2015 in class. We were given 3 tries to explore/map the environment.

We did a practice run on a separate environment built with walls that were slightly lower. The hard and sudden stops threw our laser scans off because the laser would somehow point up at stops and go outside the walls. We will then lose our localization.

Another problem that we ran into was that the MAEbot we initially used to get our code working wasn't available on the given day. Hence, we had to recalibrate another MAEbot's speed and movement during the demonstration. The proportional driving of the new MAEbot wasn't the way we wanted it to be, and our laser scans went off as well.

Our MAEbot didn't successfully map an area near the cabinet. The grey cabinet also doesn't reflect laser scans well unless the MAEbot is directly proportional to it. Hence, the MAEbot had trouble detecting the presence of the cabinet, and as a result, it would bump into it whenever it thinks it doesn't see it there.

A corner at the back of the given environment wasn't completely covered. Hence, this explains why our MAEbot wants to go in further and explore that area.

Our A* algorithm tends to "hug" walls a lot, which is a problem since we can occasionally get stuck at areas like these when the wires of the MAEbot sticks out and touches the walls. We could have fixed this by having an offset that keeps the MAEbot a certain distance away from any black region to prevent it from getting stuck.

We also had problems losing localization when we got close to a wall. We could have done an adaptive dilation where if we hit a black region (what we think is an obstacle), we could still move forward.

Other Challenges:

The MAEbot runs out of battery and does not have enough power to drive the motors even at a battery level of 3. We spent a long time figuring this out.

We also believe that there is a Wifi dead spot in the room. Whenever our MAEbot moved to that position, our localization will go off. We eventually minimized the error by removing the garbage can, which previously acted as an obstacle in our environment.

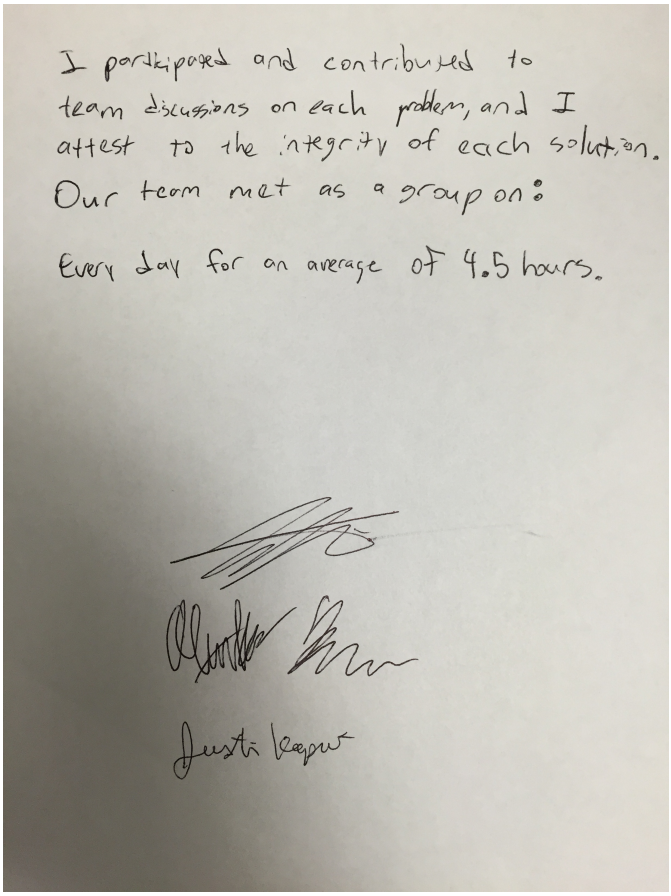

Fig. 43: Team Evaluation