Lab 5 Report
Andrew Logue, Peter McDevitt, Rishi Mayekar

2) Why is sensor noise problematic when mapping?
Mapping with proximity-based sensors is solely dependent on the real-time data obtained while the room is swept by the robot. If there is sensor noise even to a small degree, an object if only swept once with a noisy reading will be placed in a very different spot on the map, off by several tens of centimeters even. This can become a big problem when it comes to pathfinding, potentially causing a path to go through an object just because it was mapped to be a few centimeters off where it actually is.

3) How did you choose the value with which you increment the map entry? What happens if the value is too small, what happens if it is too large?
This value was chosen primarily based on trial and error. If the value is too small, objects simply would take too long to be detected, and would require the robot to take measurements for longer in a given area of the house. If the increment is too much, noise and random erroneous readings in empty areas of the world are introduced into the actual map, as every reading would be weighted or 'trusted' more.

4) How did you choose the value to threshold your map? What happens if the value is too small, what happens if it is too large?
The threshold value was primarily decided by the map entry increment, but was also based on the map output after a scan was made and how populated it was. The more times an object was sensed by the mapping process, the more likely it was to pass. Therefore the larger this threshold is, the fewer the number of objects that are registered on the map and therefore the more sparsely populated it is. Conversely with a lower threshold, more objects or false positives are registered. This threshold was adjusted based on how many important objects were registered, and if all we could see were walls, we would lower the threshold value. On the other hand, if false artifacts that were detected by the sensors due to the robot moving too fast were being shown, we would raise the threshold value.

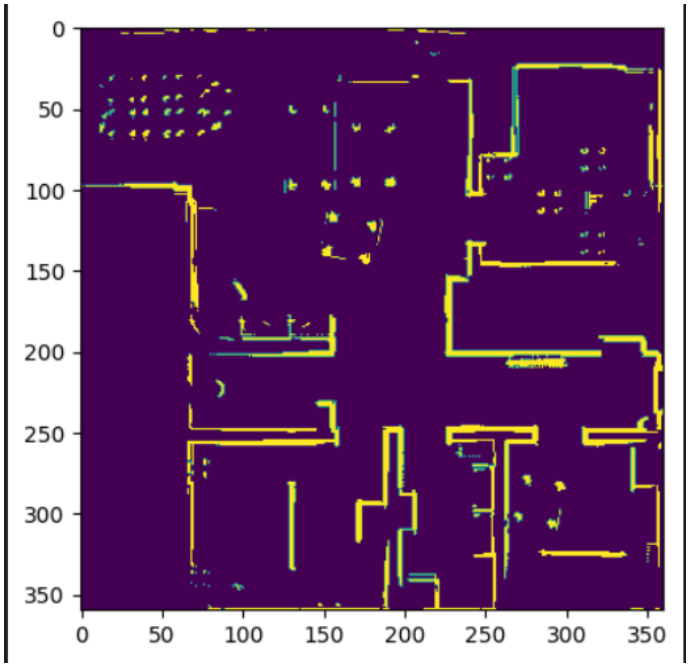
5) As Tiago is traveling along the path that the planner provided, suppose it detects an object in its way. How could you modify your solution to plan around/gracefully handle this unforeseen object in the robot's path?
In this case, the most time-efficient way to do so would be to stick close to the object and path along the side of the object in the direction that the rest of the path goes until the path is encountered again on the other side of the object. This can be done by having the front few proximity sensors constantly checking for obstacles as the robot turns towards the new path.

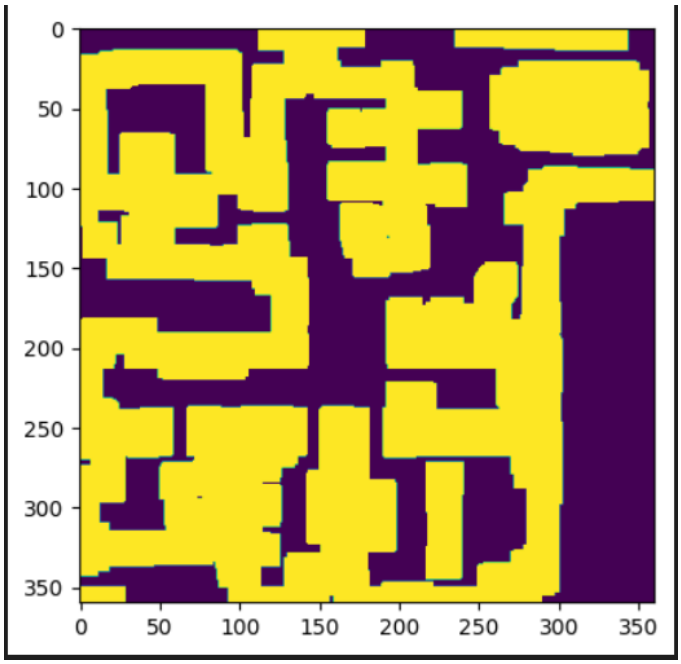
6) Could we use an algorithm like RRT to generate a viable path instead of Dijkstra's algorithm/A*? If yes, how would the path look different? If no, why not?
RRT could definitely be a valid method to use to generate a path here given the goal. It is especially applicable due to the fact that it uses the map to full effect, exploring perhaps even multiple different solutions to the problem. However the only issue besides having to choose a

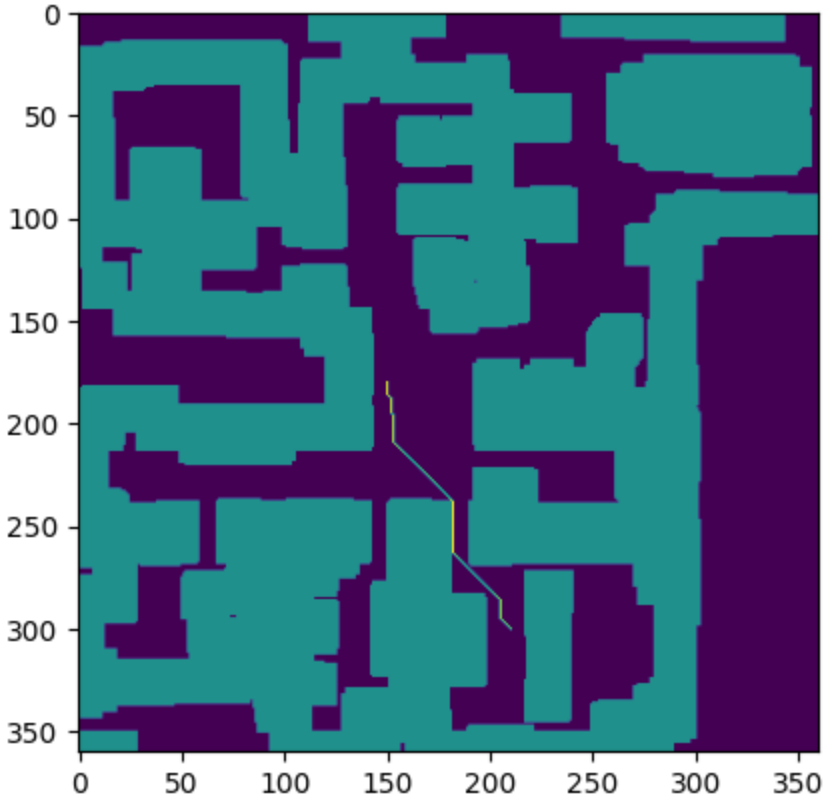
path is that there is no guarantee the path chosen has a dominant attribute, whether it be shortest length, fewest turns, etc.

7) Roughly how much time did you spend programming this lab?
8 hours

Maps:






0
50
100
150
200
250
300
350
0
50
100
150
200
250
300
350