Daniel Morton

Final Competition Report

MAE 4180: Autonomous Mobile Robots

May 20, 2021

Contents

Overview of the Team's Approach

Localization: The localization uses a particle filter with a modified weighting algorithm. As usual, the weight is mainly based on how well the depth measurement matches the expectation. If a beacon is observed, these depth-based particle weights are multiplied by their likelihood given the measured distance to the beacon and the expected distance from the particles. And, if a beacon is expected but none is observed, then the particle receives a weight of 0. A particle also receives a weight of 0 if it is outside the bounds of the map. At the beginning of the program, 48 particles spanning 360° at 7.5° intervals are initialized at each WP. The robot then turns until the particle set converges to a single WP, and from there onwards, a weighted average of all particles determines the pose estimate. Particle filter was selected over EKF since the discontinuities in the walls at corners and termination points would lead to significant errors in the EKF localization.

Roadmap: The roadmap was generated in a way that ensured coverage of the map at a low node quantity and minimal computation time. This was a three-step process: first, all WPs and EC WPs were added to the list of nodes, and second, a grid of points with a uniform spacing of 0.5 m was overlaid on the map, with all points in the free space added to the nodes. If these points did not achieve a specified minimum number of nodes, the remainder would be randomly sampled from the free space. Generally, only about 150 nodes ensured dispersion across all map regions. Then, instead of evaluating the roadmap edge connections between all nodes, a k-nearest method was used to significantly reduce the computation time.

Path Planning: Once the initial WP was determined, all permutations of the WP/EC WP order were evaluated to see what the score for the robot would be after 42 meters of travel distance (the typical maximum travel distance observed during the 5-minute tests). Since the EC WPs are worth more, this algorithm typically prioritized these. The maximum-score path was then selected, with ties in the score broken by the lowest overall cost of the path.

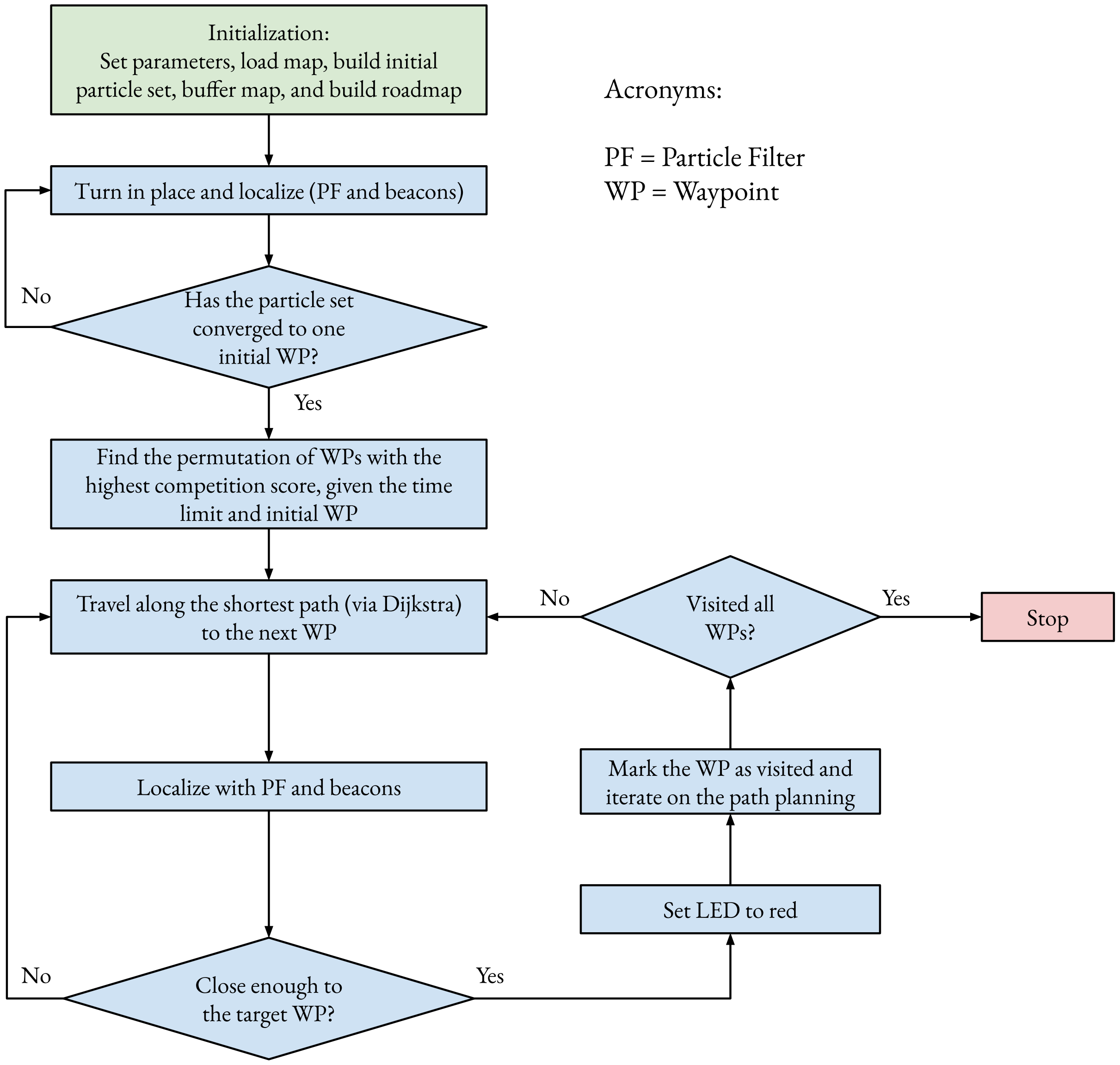
Individual Contribution, Integration, and Testing

My area of focus for this project was primarily in the roadmap generation, implementation of Dijkstra's algorithm, and some path planning. In more detail, these responsibilities included:

- Conversion of the map to polyshape format, including the buffer around obstacles and determination of enclosed regions where the robot cannot travel
- Generation of the roadmap and testing/optimization of various node sampling methods
- Implementation of Dijkstra's algorithm with our roadmap and our path planning algorithms
- My path planning algorithm: At a waypoint, calculate the Dijkstra cost of traveling to each other unvisited waypoint, and then go to the one with the lowest cost. When the waypoint is reached, mark it as visited and then find the next path
- Additional functions for building structs as well as plotting the trajectory/map

Both group members also performed test-runs of the code in the simulator for debugging and determination of the best path-planning algorithm. I contributed approximately 15-18 hours to this process.

Flow Chart of the Solution


Initialization:
Set parameters, load map, build initial particle set, buffer map, and build roadmap
Turn in place and localize (PF and beacons)
Has the particle set converged to one initial WP?
No
Yes
Find the permutation of WPs with the highest competition score, given the time limit and initial WP
Travel along the shortest path (via Dijkstra) to the next WP
Localize with PF and beacons
Close enough to the target WP?
No
Yes
Set LED to red
Mark the WP as visited and iterate on the path planning
Visited all WPs?
No
Yes
Stop

Acronyms:

PF = Particle Filter
WP = Waypoint

Discussion of Competition Performance

Run #	WPs Detected (10 pts)	EC WPs Detected (20 pts)	Incorrect WPs (-5 pts)	Bonus time	Total
1*	1	1	0	N/A	30
2	1	2	0	N/A	50
3	1	1	0	N/A	30
4	3	1	0	N/A	50

If the issue in the competition was fixed (See Figure 4, 1st plot):

Bonus	3	4	0	N/A	110

*The first run did not count because the run was reset before both group members could confirm

Run 2 restarted the 5 minute timer, and runs 2-4 were performed within this time limit. A maximum of 50 points were obtained in runs 2 and 4, though run 4 visited more waypoints overall. See Figure 1 in the appendix for the plots of these runs.

What went well?

Some of the major successes of our group's program were the particle filter and the overall path planning algorithm (despite some errors during the competition, which will be discussed later). The initialization of the robot's position worked reliably over all four runs – there were no times when the robot believed it started from an incorrect waypoint, and the robot reliably traveled in the correct direction no matter the starting orientation at a waypoint. This was also a fast process – the robot would only turn until the particle set converged, which meant if a beacon was immediately in sight, the robot did not need to spend extra time turning an entire rotation. Additionally, the PF's tracking of the pose while traveling between waypoints was quite accurate. There were no cases where the estimated pose was outside of the bounds of the robot, as observed during the competition with the live-plotted particles on the simulator. This observation also indicates that the PF had little issue dealing with the discontinuities in the walls, which, as expected, was a huge benefit of the PF as compared to the EKFs used by other groups.

The path planning algorithms also worked quite well, though the robot did not complete the full extent of the planned path. When debugging the solutions after the competition, we found that the planned paths would have earned at least 100-120 points in 5 minutes. The variation in this value is based on the starting waypoint – depending on where the robot starts, the theoretical maximum obtainable score within the 5-minute timeframe can differ. This would have resulted in a victory during the competition. See the reference trajectory in Figure 4 for an example of this path. Additionally, a major strength of this path planning was its speed – this took a maximum of only about 3 seconds during the competition, between the end of the initial localization and the actual movement. Compared with other groups, this would have yielded our team a significant advantage in terms of time available for finding waypoints.

Even the roadmap performed well in terms of calculation time - this was a strength of our program as compared with other groups, who had upwards of 900 nodes on their maps and were evaluating the connections between all of them. Had we done something similar, we would have needed to evaluate over 600x the edge connections, which would have led to some of the long delays seen in other groups' programs.

What didn't work / What could be improved?

The main issue during the competition was due to how the roadmap was built on the final competition map. We had set the radius parameter for the map buffering to be 0.3 meters – this was more than was necessary based on the radius of the robot, but we knew that the localization would not be 100% perfect, and wanted some extra space to avoid bumps in case the robot wavered slightly from the planned path. During testing with the practice maps, this worked excellently. However, on this new map, the narrow channels near the top middle EC waypoint caused some issues with the roadmap. The large buffer led to extremely narrow pathways in the free space, and the roadmap only generated a single one-way path into this region. The reason for this path being one-way was that the limited nodes in this area meant some nodes were within the k-nearest of others, but this was not the case in the other direction. This was an unforeseen consequence and it led to issues with the directionality of the Dijkstra's algorithm in use, such that the robot entered this region of the map but could not find its way out. See Figures 2-3 for a visualization of this issue in the buffer/roadmap.

Each of the following modifications would have fixed this error:

- Increasing the k-nearest value from 8 to 12
- Reducing the buffer on the map to 0.25m or less
- Appending the reverse order of the edge connections to the roadmap (and evaluating only the unique edge connections to avoid duplicates)

Since this was not anything we had run into during testing, debugging this issue required digging into the code and seeing what was returning values we were not expecting. Eventually, we found that Dijkstra's algorithm returned NaN when at this troublesome waypoint, which helped us understand where the larger issues were.

Fixing that issue with the roadmap generation would have led to a consistently successful performance in the competition. However, there are certainly plenty more ways to improve the overall performance, especially with how noise and delays are handled.

Testing the robot's performance with noise on the RSDepth sensor data (standard deviations ranging from 5cm up to 1m) showed that this program could consistently handle even extreme values of noise, likely due to the weighted averaging of the particles. At the SD = 0.05 condition, there was no noticeable difference in the trajectory/localization as compared with the no-noise case. Even at SD = 1, the robot was able to visit 6 waypoints and achieve a score of 90, without bumping into a single wall. The main difference in this case was that the robot didn't follow as direct of a path between waypoints, and the orientation saw the largest errors (much of the time lost due to inefficient motion between waypoints was because of turning). See Figure 4 for these plots.

Testing the robot's performance with communications delays revealed the primary area for improvement, especially if this was to be transitioned to a physical robot. At a delay of only 0.1 seconds, the robot's trajectory was highly erratic, and it spent a significant amount of time moving back and forth along the planned path. While it still managed to avoid bumping into the walls, it was only able to reach 4 waypoints within the 5 minutes (score: 60). And at a delay of 0.2 seconds, the robot was unable to move anywhere past the starting region. This issue with delays was expected though, since we did not account for these with interpolation or other means. See Figure 5 for these plots.

Appendix

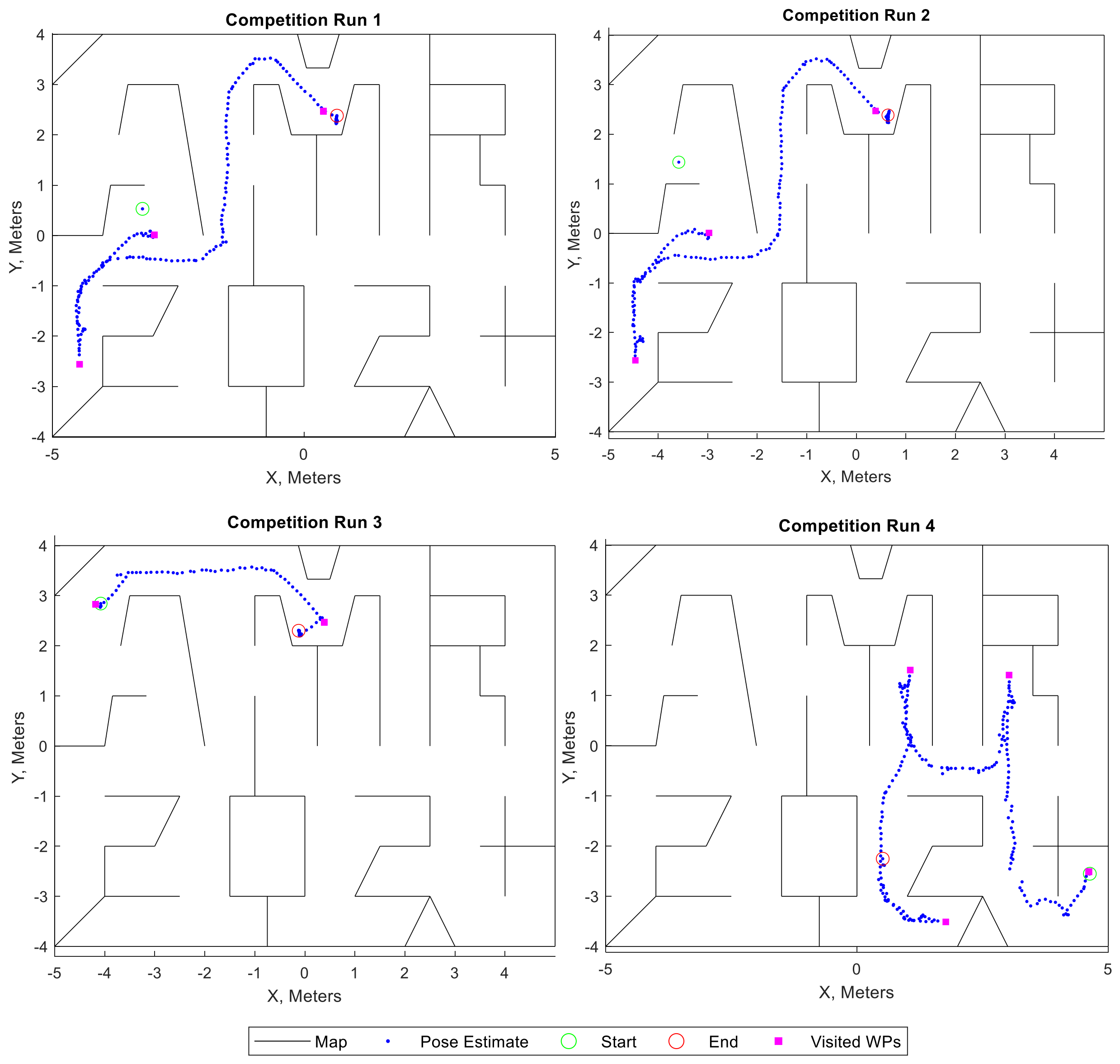


Figure 1: Competition data for runs 1-4

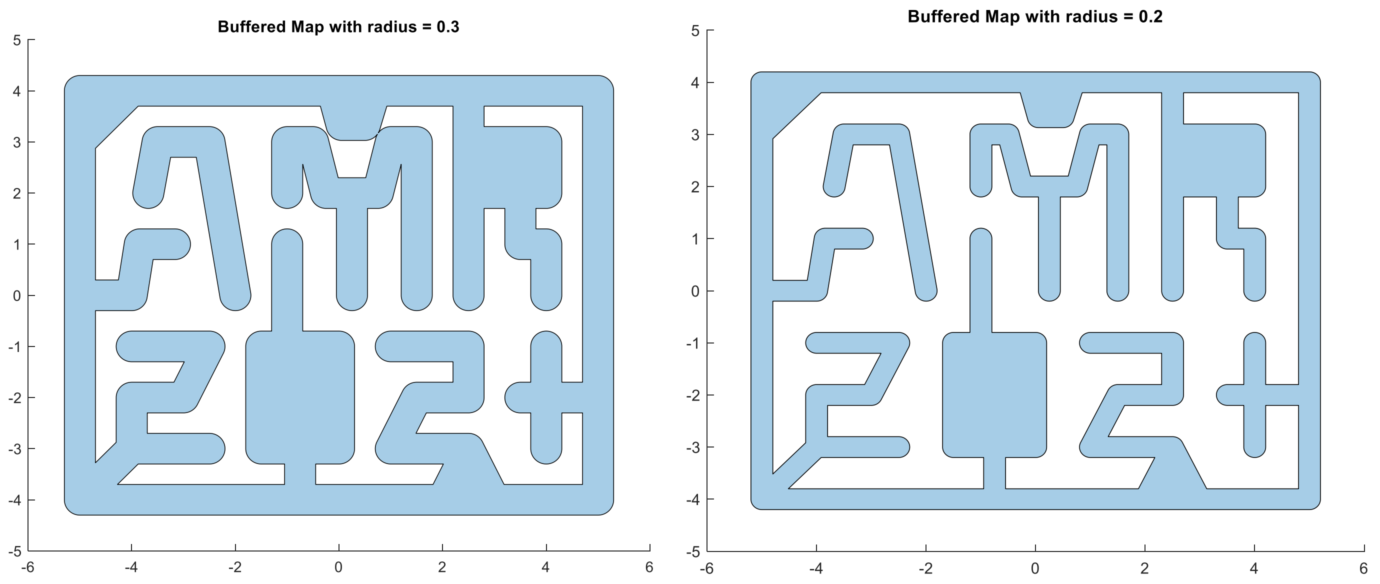

Buffered Map with radius = 0.3
Buffered Map with radius = 0.2

Figure 2: Comparison of map buffer radii. Left: 0.3 m, Right: 0.2 m. The narrow pathways with the 0.3 m buffer led to an impassable region at the top middle of the left map.

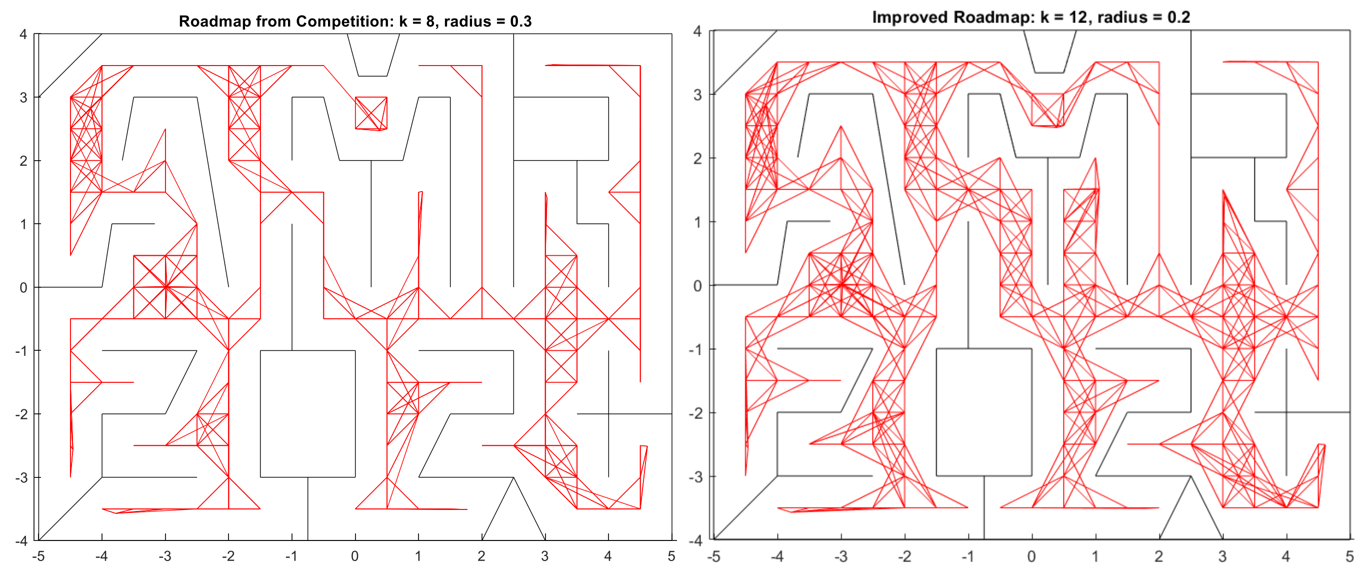

Roadmap from Competition: k = 8, radius = 0.3
Improved Roadmap: k = 12, radius = 0.2

Figure 3: Comparison of the competition roadmap (left) and the improved version which increases k from 8 to 12, and reduced the obstacle buffer from 0.3 m to 0.2 m.

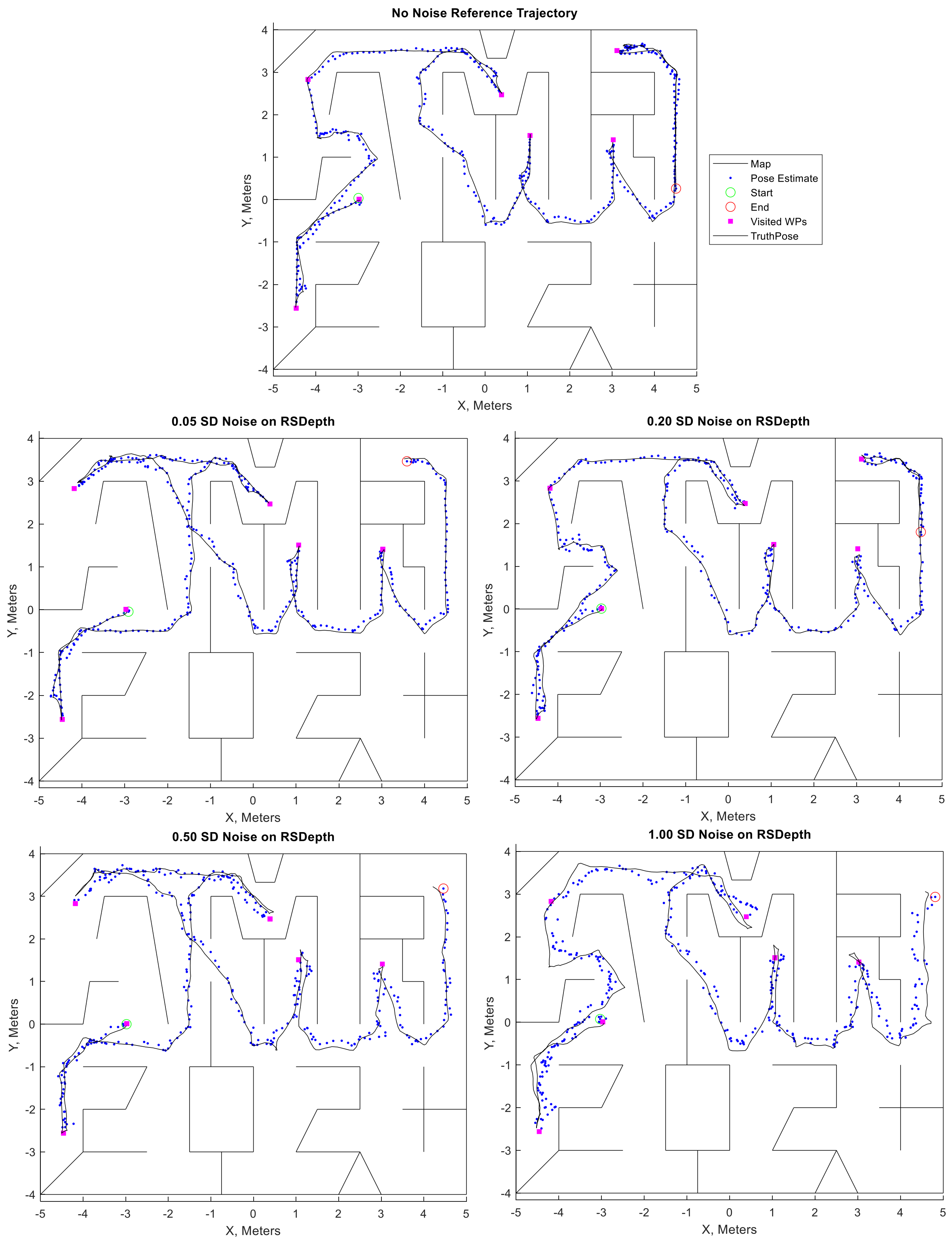


Figure 4: Comparison of trajectories with various noise values

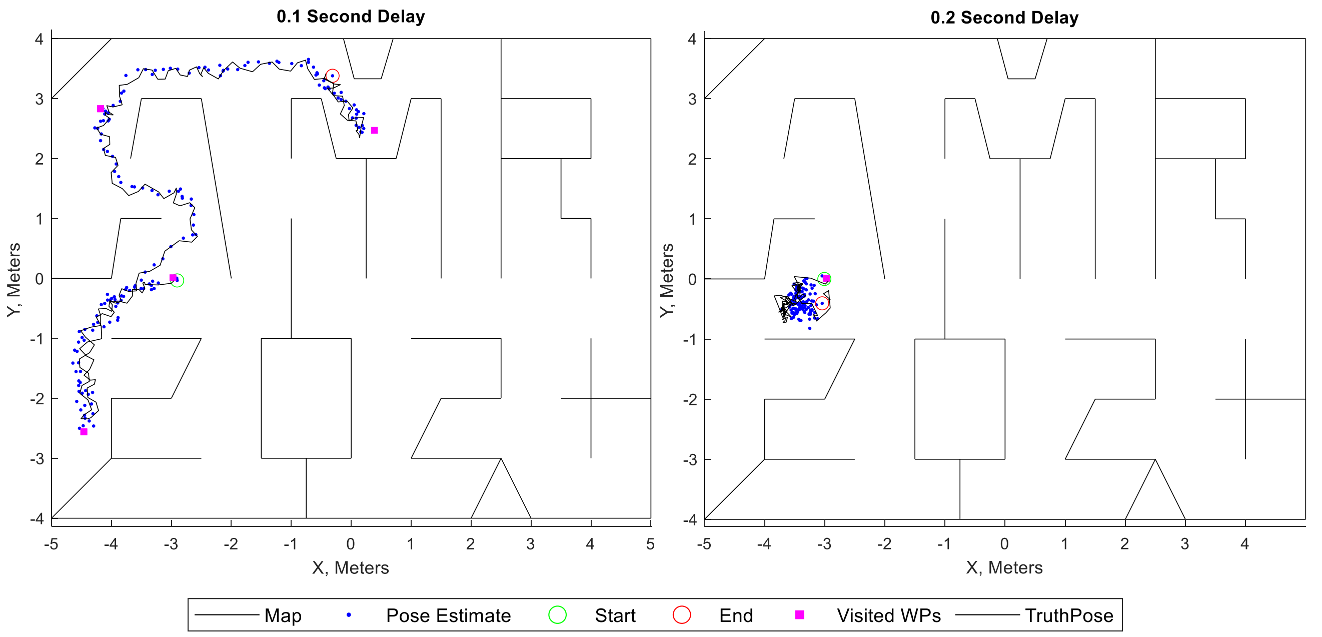


Figure 5: Trajectories with communication delays of 0.1 and 0.2 seconds