MAE 5180
Final Competition Individual Report
Group 14: Kowin Shi, Jie Huang
By: Kowin Shi, kss223

Overview

For initialization, we implemented a particle filter that splits the particles evenly (50 per waypoint), and only performs resampling independently in each waypoint set until it sees a beacon or turns for a specified number of degrees. If a beacon is visible, it is very likely to converge to the true starting pose as that is a unique identifier. If none are visible within the turn limit, a weight is accumulated for each waypoint by adding up the resampling weights, and the particle set that belongs to the highest weighted waypoint is picked. This exploits the uniqueness of each starting location, and performs additional exploration if information collected is not enough to guarantee a good enough estimate.

For localization, we chose to implement a blend of particle filter (using depth and beacon) with dead reckoning. The total particle count remains the same as initialization (number of waypoints x 50). Since the robot system is not fully observable given the limited sensor information, using a vanilla particle filter can be problematic especially when beacons are not visible. To prevent the robot estimate from unrealistically jumping between possible locations, we incorporated dead reckoning as a part of the measurement with a tunable weight multiplier. This localization result also overwrites dead reckoning to correct any error accumulation, when a beacon is seen for a continuous number of times. This is chosen over EKF as it performed better in testing, and has more parameters that can be tuned to improve characteristics such as convergence, tracking, disturbance rejection and smoothness.

For path planning, we chose a visibility roadmap with bloated walls over RRT due to the map's orthogonal nature and the roadmap's tendency to produce straight paths with minimal turns. This helps with localization accuracy. Djikstra's algorithm is used to find the shortest path to a waypoint for its fast speed. For wall detection, we wrote a custom algorithm that accumulates the depth sensor beam crossings with each optional wall. This is chosen over bump sensor solutions as it can detect walls quickly and from afar, significantly reducing the risk of localization divergence due to discrepancies in map. Overall, we chose a greedy approach to travel to the shortest path waypoint instead of planning the overall shortest path. This is motivated by the fact that the map will change from the initial assumption due to optional walls, thus the latter will not guarantee an optimal solution but will incur more computational costs.

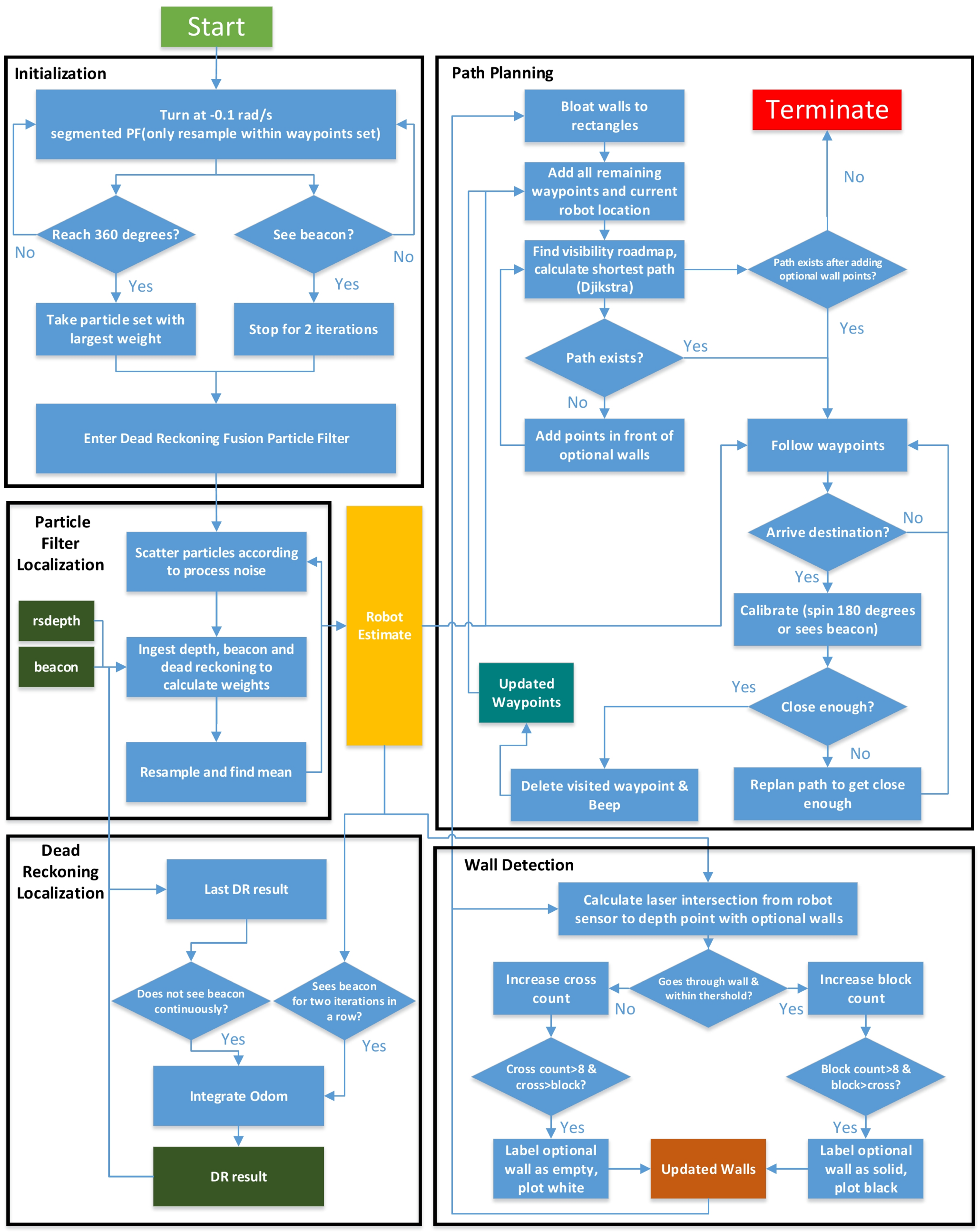
MAE 5180

Final Competition Individual Report

Group 14: Kowin Shi, Jie Huang

By: Kowin Shi, kss223

Flow Chart



Start
Initialization
Turn at -0.1 rad/s segmented PF(only resample within waypoints set)
Reach 360 degrees?
See beacon?
No
Yes
Yes
No
Take particle set with largest weight
Stop for 2 iterations
Enter Dead Reckoning Fusion Particle Filter
Particle Filter Localization
Scatter particles according to process noise
rsdepth
beacon
Ingest depth, beacon and dead reckoning to calculate weights
Resample and find mean
Robot Estimate
Dead Reckoning Localization
Last DR result
Does not see beacon continuously?
Sees beacon for two iterations in a row?
Yes
Yes
Integrate Odom
DR result
Path Planning
Bloat walls to rectangles
Terminate
Add all remaining waypoints and current robot location
No
Find visibility roadmap, calculate shortest path (Djikstra)
Path exists after adding optional wall points?
Yes
Path exists?
Yes
No
Add points in front of optional walls
Follow waypoints
Arrive destination?
No
Yes
Calibrate (spin 180 degrees or sees beacon)
Updated Waypoints
Yes
Close enough?
No
Delete visited waypoint & Beep
Replan path to get close enough
Wall Detection
Calculate laser intersection from robot sensor to depth point with optional walls
Increase cross count
Goes through wall & within thershold?
Increase block count
No
Yes
Cross count>8 & cross>block?
Block count>8 & block>cross?
Yes
Yes
Label optional wall as empty, plot white
Updated Walls
Label optional wall as solid, plot black

Initial Assumptions

- Optional walls are all solid
- Segmented PF start with each particle set having same x y coordinates as waypoint, but angular pose divided evenly from 0 to 360 degrees
- Initial waypoint set consists of both waypoints and EC waypoints
- Initial wall set consists of all walls that are not map boundaries

MAE 5180
Final Competition Individual Report
Group 14: Kowin Shi, Jie Huang
By: Kowin Shi, kss223

Individual Contribution

I worked on localization, visibility roadmap and path calculation algorithms. Jie worked on integration and wall detection, and we helped debug each other's code. We correctly identified the central challenge to be localization, in particular determining the robot's pose in a partially observable system with a low update frequency. After getting bad results with a particle filter using just depth measurements, we decided to exploit as much information as the robot has available. Incorporating beacons improved convergence significantly as they provided much more unique information than depth, and dead reckoning bridged the gap when beacons were not visible. By having the particle filter and dead reckoning exchange information, we were able to compensate for the shortfalls of each. The other challenge was to tune parameters for the low control frequency. We first tuned the controller to be robust in the simulator where the average frequency was 5.03Hz, but found that it diverged easily on physical robots where the frequency was 0.68Hz as shown in lab data. In lab testing showed that ratio of control frequency to robot velocity had a large influence on the convergence for particle filters, as an incorrect estimate can be corrected if the pose change is small (~0.1m), but will only be exacerbated if the changes are large (~0.3m). Thus, tuning using a configuration file with 0.15ms lag and 0.2m noise on depth and camera helped achieve much more robust localization in the lab. Since particle filter worked well, we did not bother developing EKF as we were only a two member team.

The other big challenge was path planning which was not anticipated. Originally we implemented a novel quadruple RRT which was slow and generated suboptimal paths (See figure A3). The random nature of RRT sampling resulted in convoluted paths and required constant turning. This worsened localization, as the low control frequency did not lend well to quick directional changes. Therefore we switched to visibility roadmap and Djikstra's algorithm. We spend about 2 days integrating code, and continued debugging throughout the entire development period. It is worth mentioning our debugging and visualization strategy. I wrote a script that plotted the truthPose and robot estimate trajectories, along with all the depth and beacon data from the robot perspective (figure A2). This was made into a gif animation, and we could ingest a raw .mat datafile from lab and output a very detailed visualization. This helped us significantly in developing high level algorithms and the wall detection logic, as we could see what the robot sees in every step of the way. See examples here.

Lab Time

We both attended as many lab sessions as allowed by the rules together, totaling 17 hours.

Competition Performance

From working with physical robots in the lab, we found that the low control frequency made our algorithm unpredictable at high speeds, no matter how well it performed in the simulator. Even though we were able to complete long distance paths across the map several times, it was inconsistent and we were only able to finish one entire run of every waypoint on the map with a velocity of 0.1m/s (see figure A1). The main source of failure seemed to occur when the robot travels through narrow passages with localization lagging behind by about 0.2m, causing it to often contact obstacles. Moreover, our wall detection algorithm required several measurement samples to prevent erroneous detection, but the low refresh rate meant that any discrepancy in our initial assumption would throw the localization off before the results are updated. Thus, robustness became our main goal and we lowered velocity to 0.06m/s, while keeping stages in our algorithm that calibrated upon getting to a waypoint, even if it cost us time. The goal was to ensure our design worked 100% of the time, and not risking a restart or a false detection for potentially higher scores.

MAE 5180
Final Competition Individual Report
Group 14: Kowin Shi, Jie Huang
By: Kowin Shi, kss223

Because of this focus, our design performed reliably during the competition, making no mistakes and needing no restarts. It detected 2 waypoints (although we misinterpreted the rules and did not beep for the starting location), 2 EC waypoints, and all 3 optional walls. It almost arrived at another waypoint before the timer ran out. From figure A2, it could be seen that the higher ratio of sampling frequency to velocity resulted in a decent robot estimate, with the error not being more than 0.2m at any point. From our observations, the robot also arrived directly on top of 3 of the waypoints. While the second to last EC waypoint was slightly off on the perimeter, the robot decided that it was close enough to not warrant an adjustment move in the calibration stage. There did not appear to be any problems with our algorithms, and the slightly higher control frequency in competition (with just one robot running) made our localization results even better than the runs in lab. The competition control frequency was 1.48Hz, compared to the 0.68Hz with 6 robots running simultaneously.

Although there were no apparent failures with our code, there are definitely areas that can be improved upon to score more points. Adding the beep to the starting location would earn us an additional 10 points on top of the 80 points. Our competition speed of 0.06m/s would take the robot around 10 minutes to finish the entire map, from simulator testing. Seeing how accurate the localization was compared to the truthPose, I believe we had a decent safety margin to accommodate a velocity increase to 0.8m/s. With that speed, we would have definitely reached the last waypoint and could have potentially reached additional EC waypoints. However, from simulator runs we found that we needed at least a velocity of 0.1m/s to complete the entire map within 7 minutes. The issue with higher velocities (besides larger robot pose changes) is that the sensor measurements drift too far from each other in terms of time and position, while our algorithm assumes that all readings are collected simultaneously (See figure A4). This is especially problematic for wall detection, as that uses the current robot estimate and depth points to construct lines for checking intersections. At higher velocities, we would have to trust sensor measurements less by using a wider distribution for weight calculation, the effect of which is untested with the competition setup.

In terms of overall competition strategy, we may have been too conservative in our approach, which limited our performance. The calibration step after reaching every waypoint was costly in time, since it rotated at 0.1 radians per second until it saw a beacon or reached 180 degrees, then stared at the beacon for 7 seconds. However, since we never had the opportunity to run a lone physical robot with the same setup as the competition, I believe we made the wiser decision to go for reliability and repeatability over a risky attempt. It is impossible to tell without testing and data if these new suggestions would still make the robot perform as robustly as it did.

If there was more time, we would like to optimize the localization parameter tuning further by running the robot on a real competition setup. We would like to find the safety margin and its relationship with the control frequency to velocity ratio, so that we can run our algorithm at the edge of stability for maximum efficiency. Moreover, seeing that many groups used different localization algorithms such as EKF and sigma point filter, we would like to experiment with them to see if they are superior to our implementation in terms of accuracy and computational cost. Lastly, we would also like to explore different path planning strategies, and see how the global optimal shortest path to all waypoints without knowing the real map compares to our design choice of a greedy approach. This was a very valuable learning experience, and exposed us to an abundance of interesting research questions regarding the observation and control of autonomous mobile robots in an uncertain environment.

MAE 5180
Final Competition Individual Report
Group 14: Kowin Shi, Jie Huang
By: Kowin Shi, kss223
Appendix

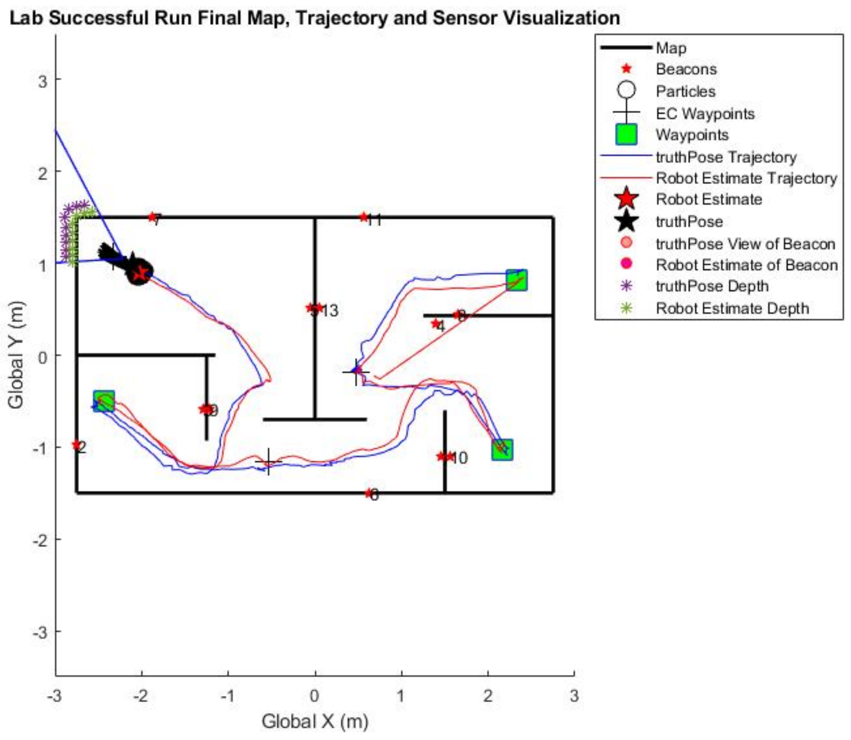


Figure A1. Lab Data Visualization

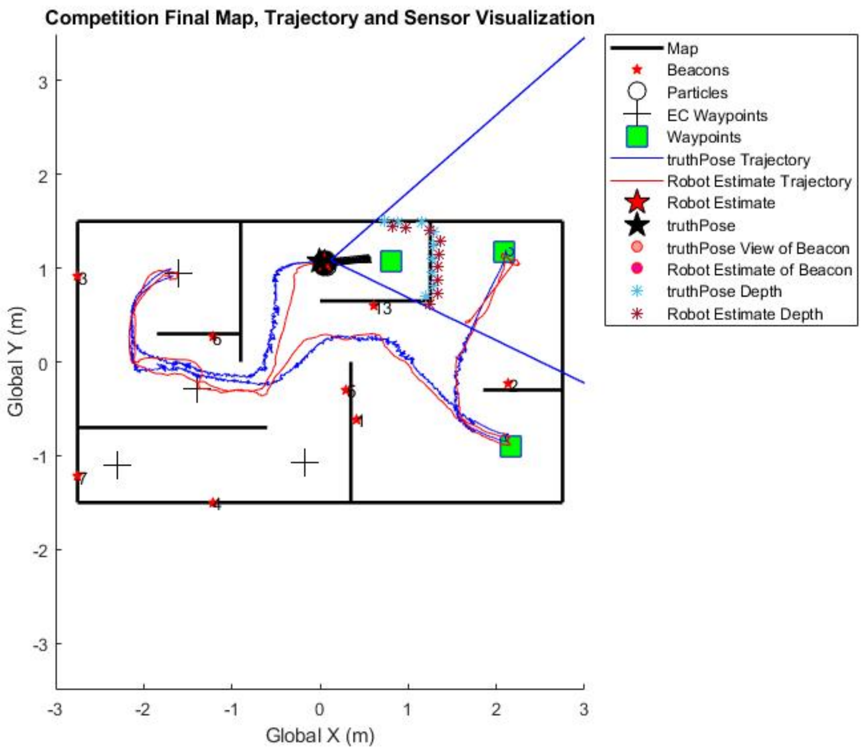


Figure A2. Competition Data Visualization

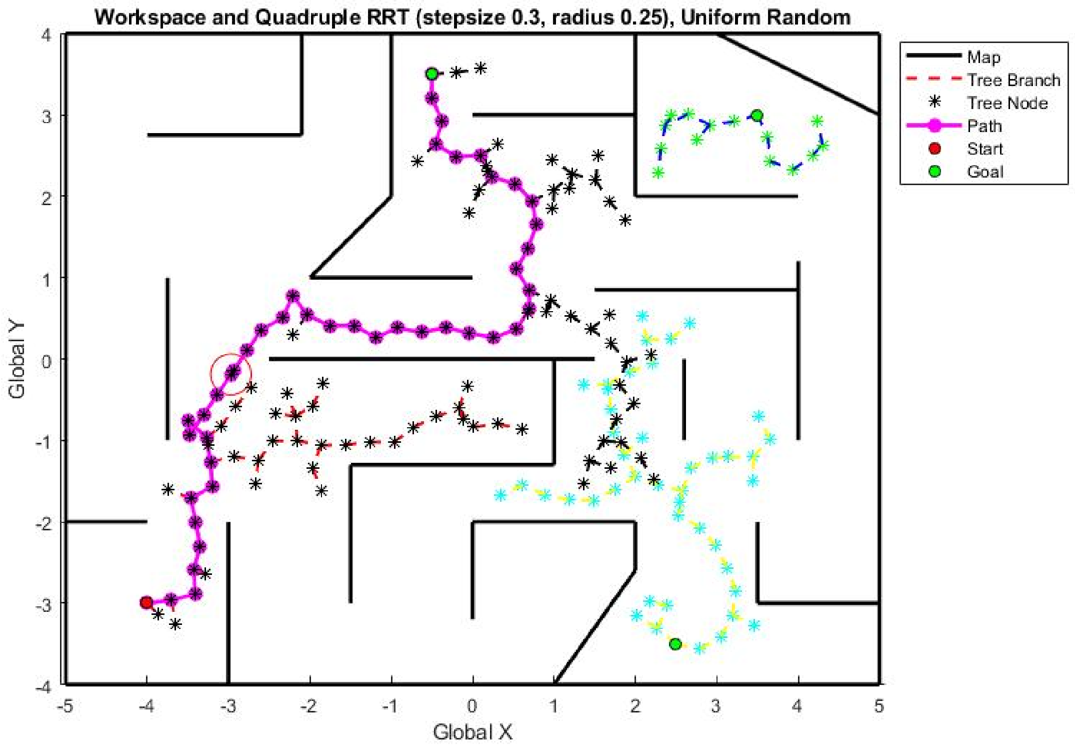


Figure A3. Quadruple RRT Experimentation Results

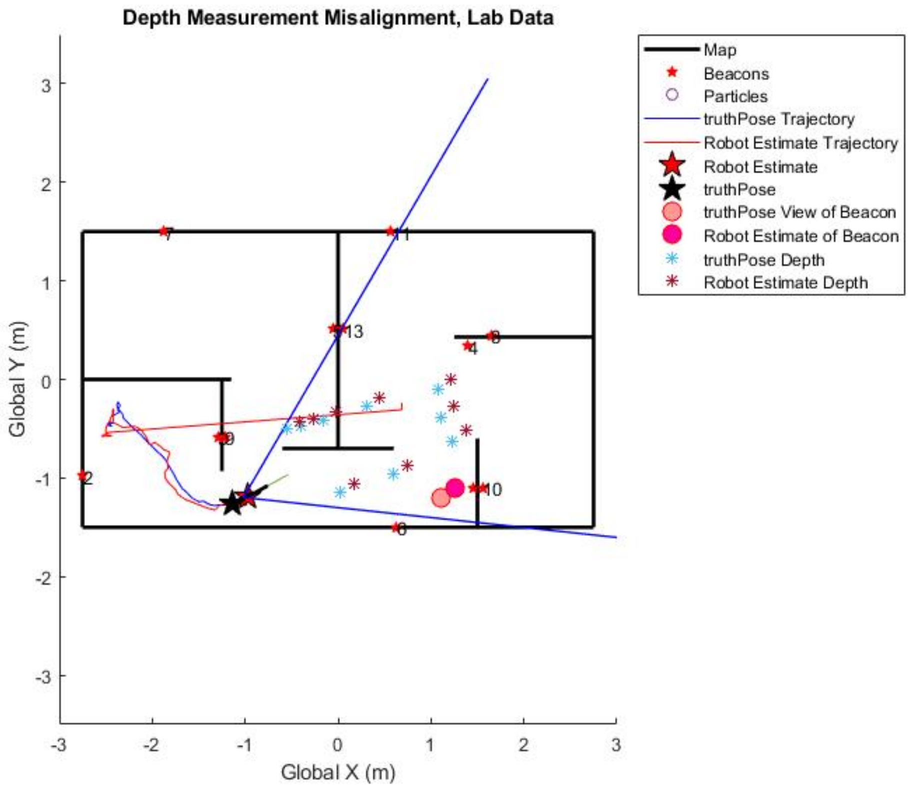


Figure A4. High Velocity Depth Measurement Misalignment, Post-Processed Lab Data

MAE 5180
Final Competition Individual Report
Group 14: Kowin Shi, Jie Huang
By: Kowin Shi, kss223

Works Cited

We used Dijkstra's Minimum Cost Path Algorithm by Joseph Kirk, jdkirk630@gmail.com and the provided intersectPoint function.