E28 | Final Project
Group Members: Michelle Zhuang and Rey Mendoza

Our final project was inspired by the past project from Jessica Berg and Sydney Covitz in 2019 of a robot tour guide. Instead of creating a map and using gmapping like them, we implemented a particle filter to localize relative to pylon "beacons" with known positions. Due to practicality constraints we were unable to have our robot complete a tour of the Engineering Department, however our implementation can be extended from the Mobile Robotics lab to a course much bigger, such as the Engineering Department as a whole. The particle filter allows the robot to localize within a map, which can then be used to navigate to other locations on a pre-planned course (or tour).

Before implementation we spent time considering how to navigate the robot: teleoperate the robot or run on a predetermined path. Ultimately, we decided to have the robot run on a predetermined path. To implement this, we extended our code from project 2 task 1. Our existing code allowed the robot to travel from a starting point to some hard-coded point in the reference frame of the robot when the code first starts running. We extended this to iterate through a set of points, allowing us to set a course for the robot to follow. Additionally, we adjusted the control and control-filtering parameters so as to make the robots movement significantly faster and less smooth. This was done with the intent of messing up the robot's built in odometry, allowing room for our particle filter to perform better by using the beacon's to localize despite the failure of the odometry. With this, we had our control set up, and now we had to implement the particle filter for localization.

We modified helper functions from the `particle_filter_demo.py code` from lecture as a basis to create our measurement model, and measurement update step. These functions use the measured distance to known beacon locations, then calculate the probability of

getting those measurements if the robot is actually at each given point. Then it samples from that distribution of points, with probabilities determined by normalizing the previously calculated probabilities to generate an updated set of particles. For our motion model, we made some simplifying assumptions. First of all, we assumed that throughout a time step, the derivative of linear motion in the x direction, and the derivative of the angle are constant. This is a very reasonable assumption since in ROS, we set the linear and angular velocity, and this value remains set throughout the time step. However, since the velocities cannot actually change instantaneously, this is a slight simplification. Additionally, we assumed that the angular velocity is small enough that the angle can be considered constant during each time step. This is a very important assumption, because it greatly simplifies the model by allowing it to be linear with respect to the controls. With these assumptions, we found our motion model to be:

$$X(t + dt) \leftarrow X(t) + cos(\theta)\dot{x}(t)dt + \epsilon_x$$

$$Y(t + dt) \leftarrow Y(t) + sin(\theta)\dot{x}(t)dt + \epsilon_y$$

$$\theta(t + dt) \leftarrow \theta(t) + \frac{d\theta}{dt}(t)dt + \epsilon_\theta$$

The next step was to integrate the motion and measurement models into our code. Since the control code already had a control callback function called on a timer with a period of 0.01 seconds, we integrated these steps into the control callback function. Each time the control callback function runs, we update our particles using the motion model, then resample using the measurement model run using the odometry as a beacon with a distance measurement of 0. Additionally, we have a cone_callback function that updates current locations of pylons whenever pylons are spotted by the camera. Any time there is an updated list of cone locations, we resample particles using the measurement model using the pylons in sight as beacons. We

identify which cone is which by estimating the location of the pylons and identifying a cone in sight as the beacon to which it is closest.

Creating the map was where we ran into the most issues. Initially, we planned on having a map that included a large part of the second floor of Singer Hall, however we soon realized that this was not practical. Even with good measurement equipment it would be quite difficult to get accurate measurements of the locations of pylons large distances apart with obstacles such as walls, chairs, and tables in the way. To make matters more complicated, we did not have good measurement equipment; all we had was a tape measure and a ruler. As a result, we decided that it would be sufficient to set up the course in the mobile robotics lab, as the concept could be easily generalized to larger maps, particularly in practical applications where a map already exists, and the beacons in questions are objects already in place rather than pylons placed by us.

We ended up placing the pylon "beacons" to form a 1m (39.37 inches) by 1m square, with the origin (0, 0) at the bottom left of the square. After adjusting the implementation of the measurement and motion model to fit the scope of our project, we fine tuned the noise constants through continuous trials.

Three trials were conducted, measuring the robot's final position (Table. 1) and plotting the (x, y) coordinates of the robot's location based on the particle filter we implemented and based on the odometry in the robot (Fig. 1-3).

	Robot's location in inches (x, y)	**Particle Filter (x, y)**	**Odometry (x, y)**
Trial 1	(1.022,-0.492)	(0.904, -0.432)	(0.984, -0.466)
Trial 2	(1.0033,-0.477)	(0.947, -0.427)	(0.980, -0.467)
Trial 3	(0.99695,-0.454)	(0.936, -0.440)	(0.982, -0.465)

Table 1. Robot's location after the end of run for trials 1 through 3

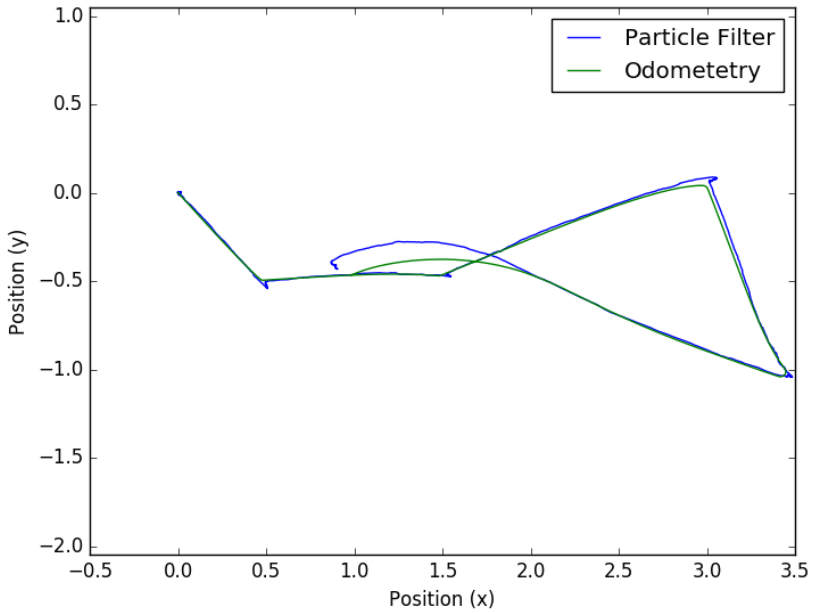

Figure 1. Plot of position (y vs x) of robot for particle filter and the odometry for trial 1

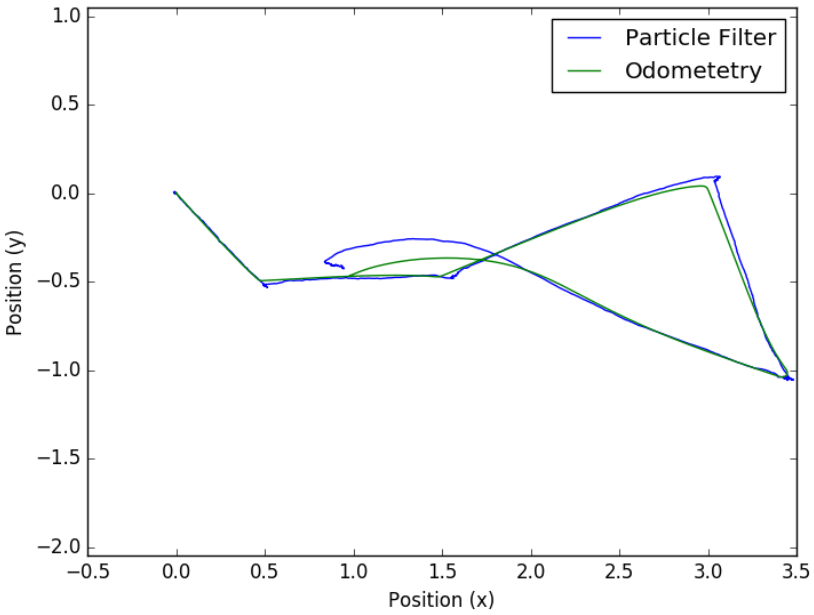

Figure 2. Plot of position (y vs x) of robot for particle filter and the odometry for trial 2

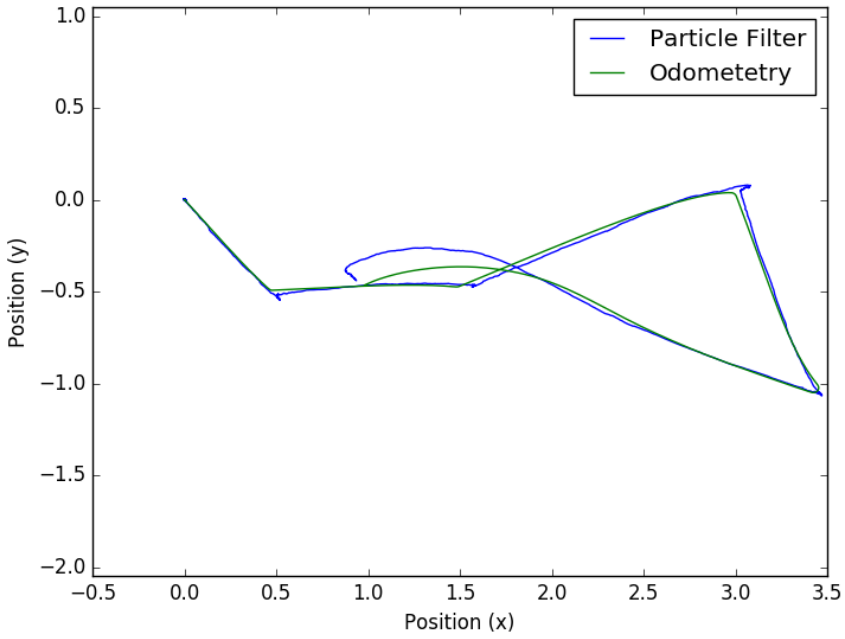

Figure 3. Plot of position (y vs x) of robot for particle filter and the odometry for trial 3

Link to Google drive with videos of trial 1-3.

From Table 1, we can see the main difficulty we had evaluating the performance of our particle filter, which is that the built-in odometry performed surprisingly well. The original idea was that the odometry would serve as a baseline measurement model, and then the location of the cones would allow the particle filter to get an even more precise location. However, the odometry was really accurate, especially when we acknowledge the imprecision of the beacon locations we provide to the robot, which were measured by hand, and therefore subject to human error. It is also worth noting that the process of measuring the robot's final location by hand included a significant amount of human error, as it was quite difficult to get the measurement using a tape and measuring at an estimated 90 degree angle from hand-placed tape markers on the ground. However, we can see that the particle filter was fairly accurate, with errors of 10% or less on all trials.

We also can note something important from the plots of the estimated location of the robot. The particle filter appears to be in much closer agreement with the odometry up until the very end, suggesting that the particle filter errors were much lower throughout most of the run. It makes sense that the particle filter would be less accurate on that part of the course since that is the part where the robot's path has a notable curve. Our motion model assumes no curves, so it makes sense that it would have larger errors during curved parts of the course. A more advanced motion model might be able to deal with these curves better.