Lab 4: Particle Filter for Localization

Josephine King, Peter Johnson

***Abstract*—Localization has been a central problem in the field of robotics. This report details the design and implementation of a particle filter (PF) for the localization of a robot in a coordinate frame using a known landmark as reference. With a lidar and a compass as the sensors, the filter was able to achieve an overall path tracking Root Mean Square Error of 0.2041 m.**

I. Introduction

LOCALIZATION is a fundamental problem in the field of autonomous mobile robotics. Mobile robot localization is the process of determining the pose of a robot relative to a given map of the environment. This report details the localization of a robot equipped with a lidar and an inertial measurement unit (IMU) navigating a 10 by 10 meter square path located in the center of an open field. At the center of the square is a pylon, the only object discernible by the lidar. This pylon serves as a landmark at a known location in the global coordinate frame (5.0 m, -5.0 m), allowing for particle filter (PF) localization. The global frame defines east as the positive-x direction and north as the positive-y direction. The yaw angle is 0 rad along the global x-axis, and the positive angle direction is counter clockwise (CCW). The IMU's coordinate frame is centered on the robot, with the y-axis pointing to the robot's left and the x-axis pointing forward. Similarly, the lidar's y-axis points forward on the robot and the x-axis is to the right. Figure 1 shows all three coordinate frames relative to the robot. The robot begins its square navigation at the origin of the global frame, which is the north-west side of the square.

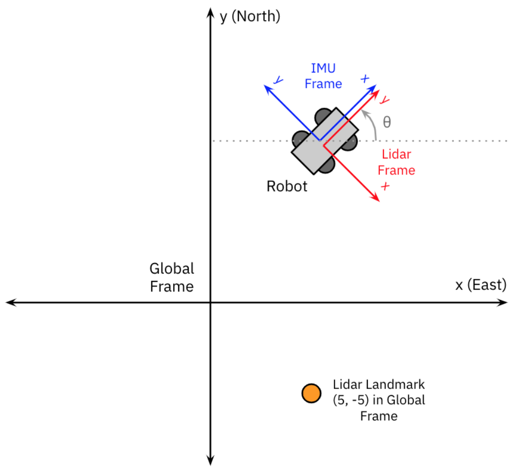

Fig. 1. Top-down view of robot with local frames, global frame, and landmark.

II. Particle Filter Design

A. Background

The particle filter is a localization technique which uses probabilistically generated "particles" to estimate a robot's state. Each particle p_t^i at a given time step t consists of a state, x_t^i, and an associated weight, w_t^i. To begin, the set of particles P_t is initialized randomly throughout the configuration space with evenly assigned weights. From here, the following algorithm is run at each time step, where u_t and z_t are the odometry and measurement vectors at each time step, respectively. The set of particles P_{t-1} from the previous time step is used to estimate the set of particles for the current time step.

Algorithm 1 Particle Filter Algorithm

```
Input  : $P_{t-1}$, $u_t$, $z_t$
Output: $P_t$
for $i = 1...num\_particles$ do
    pick $p_{t-1}^i$ from $P_{t-1}$
    sample $\mathbf{x}_t^i$ with probability $P(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i, u_t)$
    calculate $w_t^i = P(z_t|\mathbf{x}_t^i)$
    add $p_t^i = [\mathbf{x}_t^i w_t^i]$ to $P_t^{predict}$
end
for $j = 1...num\_particles$ do
    sample $p_t^j$ from $P_t^{predict}$ with probability $w_t^j$
    add $p_t^j$ to $P_t$
end
return $P_t$
```

Lastly, a single state estimate may be determined at each time step from the set of particles P_t.

B. Motion Model

The robot's state is $\mathbf{x_t} = [x\ y\ \theta]^T$. In a PF, the density $P(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i, u_t)$ is sampled to generate a prediction for a particle's new state $\mathbf{x}_t^i$. A motion model with added random noise performs this sampling.

While the robot recorded x and y accelerations with an IMU, these data are not used by the PF. Instead, the motion model for the robot uses a random velocity V_p and perturbed yaw measurements yaw_p from the IMU's compass. Additionally, the motion model assumes that there was no slip perpendicular to the vehicle's motion. The motion model is as follows, where dt = 0.1s is the length of each time step.

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} + V_p cos(\theta_{t-1})dt \\ y_{t-1} + V_p sin(\theta_{t-1})dt \\ yaw_p \end{bmatrix} \tag{1}$$

Because the robot was moving at slightly different rates at each time step or not moving at all, the random velocities V_p are selected from a uniform distribution that ranges from 0 to $2 \times V$, where V is the average velocity of the robot over its square path. V is estimated to be the length of the robot's expected path divided by the total time travelling the path.

$$V = \frac{4 \times 10m}{(\text{ of time stamps}) \times 0.1s} \tag{2}$$

The motion model relies on compass measurements to determine the robot's yaw, θ_t. To account for the compass's variation, the raw yaw is perturbed by added Gaussian noise. Because there are many different velocities and angles at each time step, there are many different predicted states and the conditional probability can be used to determine which predicted state most closely matches the measurements for that time step. By generating many different predictions, the PF is more likely to capture the true state, which would be highly weighted and favored in resampling.

C. Measurement Model

The conditional probability of a measurement given a particle's state is used to determine the weight of that particle. The measurement vector is defined to be $z_t = [x_p, y_p]^T$ where x_p and y_p are the x and y distances from the robot to the pylon in the global coordinate frame. z_t is obtained by rotating the lidar measurements x_{pL} and y_{pL} clockwise by θ_t from the lidar's frame to the global frame:

$$z_t = \begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} x_{pL}cos(\theta_t) + y_{pL}sin(\theta_t) \\ -x_{pL}sin(\theta_t) + y_{pL}cos(\theta_t) \end{bmatrix} \tag{3}$$

The measurement model relating the measurement z_t and the state $\mathbf{x_t}$ is as follows, where (X_L, Y_L) is the position of the pylon in the global frame and (x_t, y_t) is the position of the robot in the global frame.

$$z_t = \begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} X_L - x_t \\ Y_L - y_t \end{bmatrix} \tag{4}$$

D. Resampling Algorithm

After the particles are propagated through the motion model and their weights are calculated, the set of particles is resampled such that each particle's frequency in the set corresponds to its relative weight. The following algorithm is used to resample the particles:

Algorithm 2 Resampling Algorithm

```
Input  : $P_t^{predict}$
Output: $P_t$
$w_{tot} = \sum_j w_j$
for each particle $i = 1...num\_particles$ do
    $r$ = rand('uniform')$*w_{tot}$
    $j = 1$
    $w_{sum} = w_1$
    while $w_{sum} < r$ do
        $j = j + 1$
        $w_{sum} = w_{sum} + w_j$
    end
    $p_i = p_j^{predict}$
    add $p_i$ to $P_t$
end
return $P_t$
```

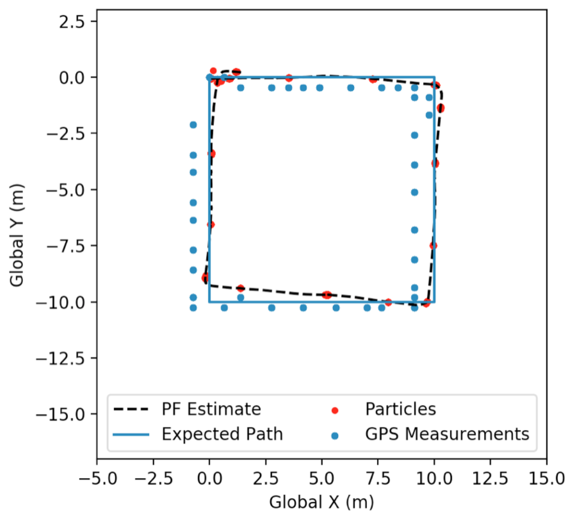

Fig. 2. Top-down view of the robot's expected path, GPS path, and PF estimated path with unknown starting state. Particles are plotted every 50 time steps. The path starts at the north west corner of the square and moves clockwise around it.

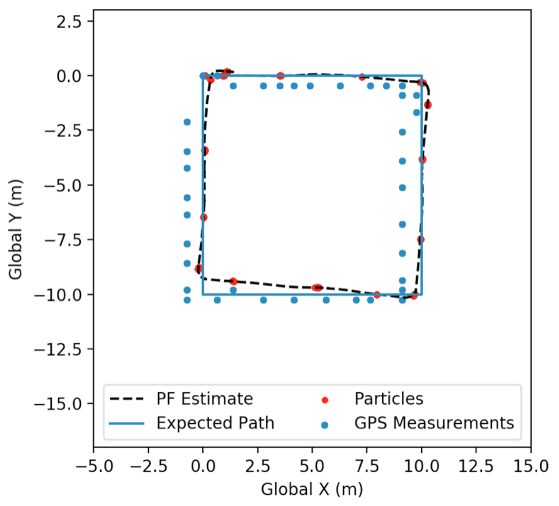

Fig. 3. Top-down view of the robot's expected path, GPS path, and PF estimated path with known starting state. Particles are plotted every 50 time steps. The path starts at the north west corner of the square and moves clockwise around it.

E. Clustering Algorithm

For many particle filter applications, a single state estimate is needed at each time step. There are many different algorithms to calculate a state estimate, such as subtractive clustering, which has a Big O run time of $O(N^2)$. When subtractive clustering was used with 1000 particles, the run time was over 10 hours. Thus, instead of subtractive clustering, the particle filter employs a simple estimation method, which estimates the state to be the state of the most highly-weighted particle. This simple estimation algorithm is $O(N)$, bringing the run time for 1000 particles down to less than an hour.

While less computationally expensive, the simple estimation algorithm has drawbacks. It cannot accurately capture cases in which particles break into more than one cluster, and may not always choose a particle in the center of a cluster like

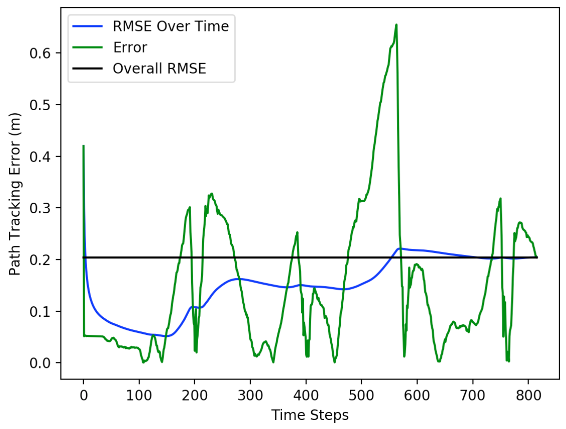

Fig. 4. Plot of the path tracking RMSE over time for 1000 particle PF with unknown start, as well as residuals for each individual time step

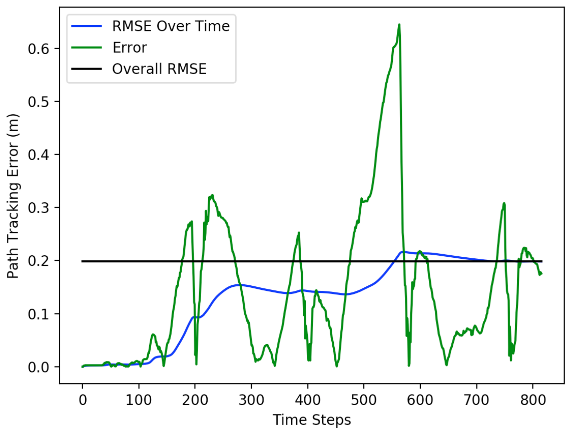

Fig. 5. Plot of the path tracking RMSE over time for 1000 particle PF with known start, as well as residuals for each individual time step

subtractive clustering does. However, for this data set, which uses accurate lidar and compass measurements in an open field with a single landmark, the particles tend to remain in one tight cluster, allowing for use of this simple algorithm.

III. Performance

The filter was tested with a varying number of particles, both with a known and unknown starting position. The estimated path of the robot using the PF matched closely to the expected path, as shown by Figures 2 and 3. According to the estimated path and expected path, the GPS appears be off by about 1m, which makes sense given the accuracy of most GPS systems. The particles, which are plotted every 50 time steps, have a greater spread in the direction of the robot's travel because of the no slip assumption and the significant variance in randomly selected velocities. In the case of an unknown starting state (Figure 2) the particles are more widely scattered in the beginning, as the particle's states have not yet converged to a few estimates.

The error of the estimated path to the expected path is shown in Figures 4 and 5, along with total RMSE and the RMSE over time. In the case that the estimated position is either in

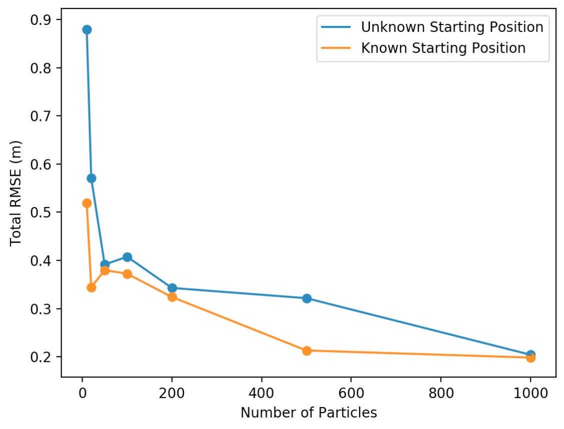

Fig. 6. Number of particles vs. calculated RMSE for both known and unknown starting positions.

between the x-bounds of the square or the y-bounds of the square, error may be calculated according to the equation

$$error_t = min(|x_{est,t} - 10|, |x_{est,t}|, |y_{est,t} + 10|, |y_{est,t}|) \quad (5)$$

If the position is beyond these bounds, it may be computed as the distance to the nearest corner, using the distance formula. These equations were used to calculate the error at each time step, and the RMSE was calculated according to the equation

$$RMSE_t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t} error_i^2} \quad (6)$$

where t is an integer representing the time index. As shown by Figures 4 and 5, the error is largest at the northeast, southeast, and northwest corners of the square (around 200, 400 and 600 time steps). These spikes in error align with Figures 2 and 3, which show a large deviation from the expected path at the corners. The overall RMSE is about 0.2041 m for navigating around the entire square.

Increasing the number of particles resulted in improved path tracking error. See Figure 6. With a known starting location, often 10 particles would be able to achieve a rudimentary path estimat with less than 0.5 m RMSE. For an unknown starting location though, 1000 particles or more were needed to produce a state estimate with less than 0.3 m RMSE.

IV. Conclusion

A particle filter was successfully designed and implemented to localize a robot using a known landmark. With a known start location, only about 200 particles were necessary to successfully localize the robot. With an unknown starting location, 1000 particles ensured that one particle would be close enough to the true starting position to be highly weighted and be resampled many times. The RMSE of the estimated path to the estimated path with 1000 particles was 0.2041 m. This error is comparable to the performance of the Extended Kalman Filter in Lab 3, which achieved an RMSE of 0.20 m.