Politecnico di Torino

Department of Electronics and Telecommunications

Master’s degree in Mechatronic Engineering

2024/2025

Sensors, Embedded Systems and Algorithms for Service Robotics

Report 2

Date 16/12/2024

Group n° 1

Gabriel Cunha – S335404

Baptiste Hooghe – S334817

Thiago Levin – S334838

Paul Novoa – S339335

1 Objective

This laboratory activity aims to develop practical skills and deepen the theoretical understanding of robot localization using the particle filter e filter and ROS 2. Localization is a fundamental task in robotics that involves estimating a robot's position and orientation in its environment. This is critical for enabling autonomous navigation and decision-making. The difficulty depends on environment complexity and this report will show several differences between simulation and real environment testing.

One common and effective method for localization is the Particle Filter (PF), a probabilistic approach that estimates the robot's pose using a set of weighted particles representing possible states. Each particle is updated iteratively based on motion and sensor measurements, resulting in a robust estimation even in noisy environments.

In this lab, we developed a ROS 2 package to implement the Particle Filter for robot localization. The system leverages a landmark measurement model, where the robot localizes itself using distance and bearing measurements to known landmarks in its environment. The project is divided into two key tasks:

1. Simulation-Based Development and Testing: The Particle Filter localization system was designed, implemented, and validated in a controlled virtual environment using the Gazebo simulation. This simulation phase allows for iterative testing and debugging without the constraints of physical hardware.
2. Real-World Validation: The localization system was deployed on a real robot in a lab environment. Trajectory data recorded during robot movement was used to evaluate the system's performance under realistic conditions. A rosbag file of the experiments is recorded if time constraints limit live experimentation.

The main objectives of this lab are:

- To design and implement a ROS 2 package that realizes Particle Filter-based localization.
- To evaluate the system in both simulated (Gazebo) and real-world environments, analyzing its effectiveness in tracking the robot's pose accurately.

This report outlines the methodology, implementation details, and experimental results, providing insights into the system's performance.

2 Design

2.1 Task 1

The ROS 2 program that has been developed implements a particle filter-based localization system for a robot. It tracks the robot's position and orientation using motion commands, landmarks detected by sensors, and probabilistic models using the following elements.

The primary node in this program is ‘pf_node’, which implements the core logic for particle filtering, including initialization, prediction, and update steps. The node uses three publishers. The first publishes the estimated robot position as an Odometry message to the /pf_home topic. The second publishes a MarkerArray to the /particles_home topic to visualize the particle distribution in RViz. Lastly, the node publishes a MarkerArray to the /landmark_markers topic to visualize landmarks in RViz.

Two key subscribers are also part of the program. The node subscribes to the /cmd_vel topic to receive the input velocities to the robot as Twist messages, which are used to predict particle motion through a velocity motion model. The node also subscribes to the /landmarks topic to receive LandmarkArray messages, which tell which landmarks the robot can see, at what distance they are, and what the bearing to that landmark is. These measurements are used to update the particle filter estimation.

The particle filter logic consists of three main steps. The prediction step updates particle positions using velocity commands and a motion model with noise. The update step adjusts particle weights based on landmark observations and resamples particles if the effective sample size is low. Finally, the estimation step computes the weighted mean of the particles to estimate the robot's current state. This probabilistic framework allows the robot to maintain a robust belief about its position, even in the presence of noise and uncertainty.

2.2 Task 2

For task 2, the particle filter implementation is the same, but the robot receives landmark information through a camera that publishes the landmarks on the topic /camera/landmarks.

3 Experiments

3.1 Simulation

In the experiments, the Particle Filter (PF) was implemented in a Gazebo simulation environment to estimate the robot's x-y position and yaw angle. The evaluation involved comparing the performance of the PF against the robot's ground truth odometry data (`/odom`). To analyze the impact of different resampling strategies on the PF's performance, four resampling techniques were tested: systematic, simple, stratified, and residual resampling.

Each resampling strategy was simulated and evaluated. During these runs, the performance of the PF was assessed by calculating two metrics: the root mean square error (RMSE) and the mean absolute error (MAE) between the estimated state from `/pf` and the ground truth from `/odom`. These metrics were computed for both position and orientation.

To ensure a fair comparison, the experiments were designed to vary the parameters specific to each resampling strategy. However, it is important to note that the `/odom` performance varied between runs, introducing some variability in the comparisons. The results of these experiments are presented in the results section, where RMSE and MAE values highlight the accuracy and consistency of the different resampling methods. This methodology allowed for a comprehensive analysis of the trade-offs between resampling strategies in the context of particle filter-based localization.

Problems due to the symmetry of the map were encountered, leading sometimes to a wrong pose estimation. This could be fixed by tuning the noise covariance matrices. To find an ideal set of sensor and model noise covariance parameters for the particle filter, a systematic method was used. The covariance matrices for the measurement and sensor signals were initialized to $0.1 * I$ and I, respectively. Then, the parameters were tuned in steps 0.01 and 0.1, respectively, to find a good set of parameters. The good set of parameters ended up being [0.01,0.4] for the measurement noise covariance and [0.5,0.5] for the model noise covariance.

3.2 Real Robot

After successful simulation validation, the particle filter package was deployed on a physical robot in a controlled lab environment. In real-world tests, a camera replaced the simulated /landmarks topic, providing landmark data. The robot was manually controlled using the teleop_key function. Only the systematic resampling strategy was used for this experiment due to a lack of sufficient time to test all the different resampling strategies. The focus was to tune to parameters to achieve good performance for the systematic resampling strategy.

The particle filter tuning process involved several critical adjustments to achieve accurate and reliable state estimation. Initially, the number of particles (N) was tested, ranging from 2000 to 6000. Although high number of particles increased initial convergence speed, N=2000 was chosen because the computer was encountering issues with the high number of particles.

Model noise covariance (σ_u) was systematically increased from [0.1,0.1] by steps of 0.1 to improve particle spread, with [2,2] yielding the best results. Measurement noise covariance (σ_z) was tuned by systematically lowering the range parameter from 0.2 to 0.05 in while keeping the bearing parameter at 0.2. This significantly improved the performance of the filter during turning operations.

A significant issue was the robot's initialization, which included offsets in both yaw and x-y position. These issues were corrected by updating the robot model to start at x=0 m, y = 0 m, and yaw = 0 rad. Additionally, the particle filter's state boundaries were adjusted to x ∈ $[-0.9,2.7]$, y ∈ $[-1.7,1.2]$, and yaw ∈ $[-\pi,\pi]$. This ensured the particles operated within the robot's realistic workspace. These refinements collectively resolved issues with particle clustering, position inaccuracies, and orientation errors, leading to a well-tuned particle filter. The final parameter values are shown below.

$$N = 2000$$

$$\sigma_u = [2,2]$$

$$\sigma_z = [0.05,0.2]$$

4 Results and Discussion

4.1 Task 1

In Task 1, the PF was implemented to estimate the x-y position as well as the yaw angle of the robot in a Gazebo simulation. 4 different resampling strategies were used and compared, namely, the systematic, simple, stratified, and residual resampling strategies. Tables 1 and 2 present the root mean square error (RMSE) and mean absolute error (MAE), respectively, for /pf vs /odom relative to /ground_truth. Note that the sampling strategies were tested on different runs, so the /odom performance varies. It is clear from the tables that the best resampling strategy is the so-called residual resampling strategy, as it has the best performance metrics relative to the performance metrics of the /odom topic. For this reason, the results presented in this section will use the data from the run using the residual resampling strategy.

Sampling Strategy	X-Y Root Mean Square Error (m) pf/odom	Yaw Root Mean Square Error (rad) pf/odom
Systematic	0.21/0.00	0.95/0.20
Simple	0.25/0.00	0.50/0.13
Stratified	0.13/0.00	0.50/0.18
Residual	0.17/0.00	0.20/0.15

Table 1. RMSE of Each Resampling Strategy for /pf and /odom

Sampling Strategy	X-Y Mean Absolute Square Error (m) pf/odom	Yaw Mean Absolute Square Error (rad) pf/odom
Systematic	0.11/0.00	0.43/0.01
Simple	0.19/0.00	0.36/0.00
Stratified	0.11/0.00	0.36/0.01
Residual	0.11/0.00	0.17/0.00

Table 2. MAE of Each Sampling Strategy for /pf and /odom

The plots for the estimated x-position, y-position, yaw angle, and x-y position are shown in Figures 1-4, respectively. At the beginning of the simulation, the particle filter estimated position is off by 25 cm in the Y position, and a little error on the yaw angle of about 0.1 rad. This may be due to the uniform initialization of the particles on the map. However, as the simulation continues and the particle filter can use more landmark data to correct the estimation, it converges to the correct pose. By the end of the simulation the pose estimation error is very clearly close to zero. A reason the particle filter takes a relatively long time to converge to the right pose may be the map. As seen in Figure 4, the map consists of landmarks set up as a square. Additionally, although not shown in the figure, the walls surrounding the map form a hexagon. The inherent symmetry of the map can pose some difficulties to the particle filter, as the environment the robot is sensing can translate to multiple possible positions on the map. This is because having multiple possible locations on the map can create multiple convergence points for the particles.

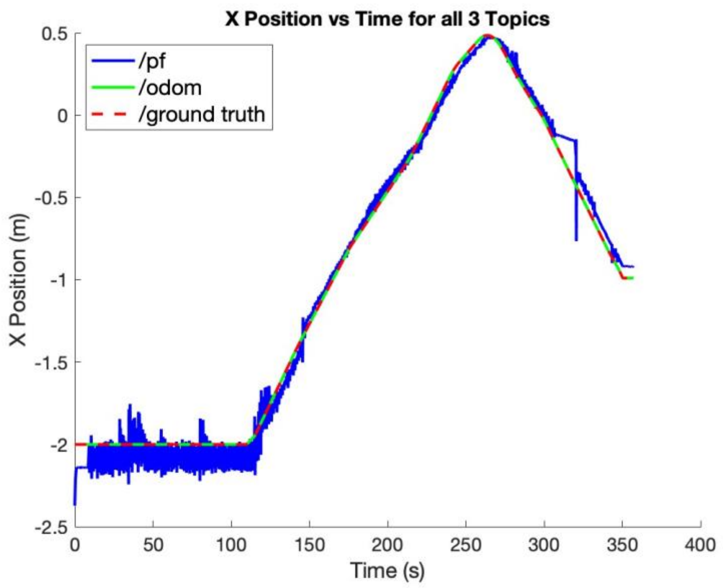

Figure 1. Estimated X-Position vs True Position and On-Board Odometry

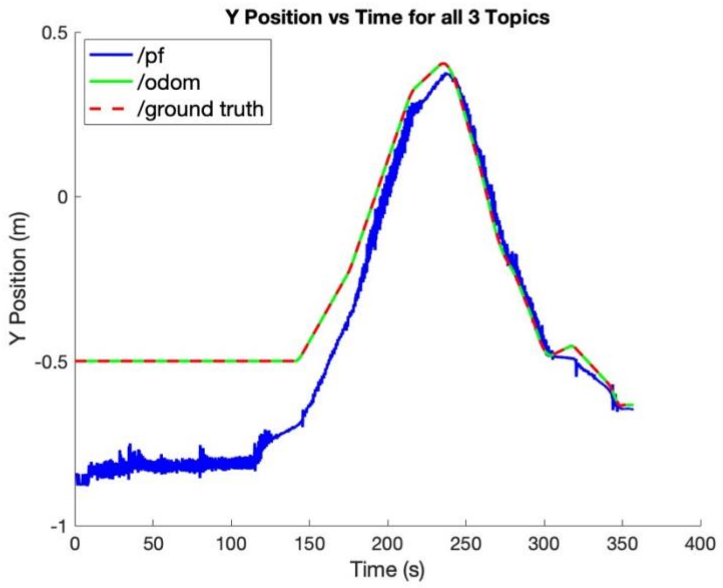

Figure 2. Estimated Y-Position vs True Position and On-Board Odometry

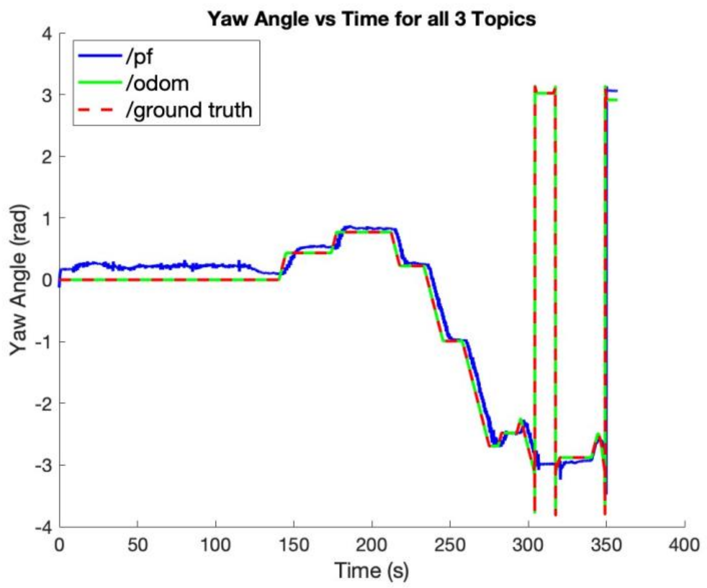

Figure 3. Estimated Yaw Angle vs True Yaw Angle and On-Board Odometry

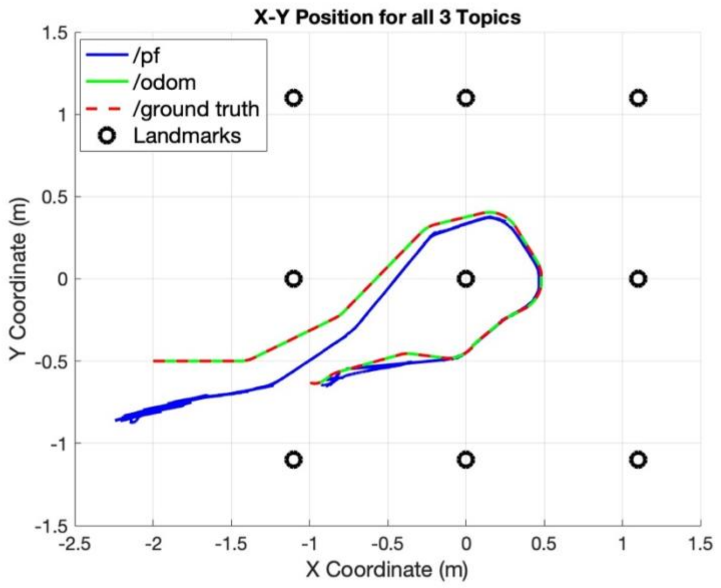

Figure 4. Estimated X-Y Position vs True Position and On-Board Odometry

4.2 TASK 2

In task 2, the particle filter was deployed to estimate the x-y position and the yaw angle of a real robot. The systematic resampling strategy was used and tuned to obtain the best results possible, as described in the previous section. Figures 5-8 show the x position, y position, yaw angle, and x-y position signals of the on-board odometry and the particle filter. Note that the odometry signal was shifted to match the particle filter's initial position because the odometry can only measure relative displacements.

As you can see in the plots, there are certain times when the particle filter does a good job estimating the x-y position of the robot, and certain times when the estimation strays away from the actual x-y position. We can also see from the plots that the filter does a relatively poor job estimating the yaw angle in real time. It seems like the yaw angle estimated by the particle filter is lagging the yaw angle estimated by the odometry by a few seconds.

One reason for these inaccuracies is that the robot is not always able to see the landmarks. This means that the particle filter has no way to correct its position prediction from the motion model with real life data and thus it must rely solely on the motion model and the given inputs to make the estimation. The problem with this is that there are a lot of environmental disturbances that the model does not account for. For example, there may be slippage between the wheels and the floor. There could also be small time delays between the velocity command signal and when the robot reaches that velocity. However, once the camera can detect one of the landmarks, the particle filter is able to converge on the true position once again.

Another reason for the discrepancies between the estimation and the true position is that the landmarks may not be in the exact position the particle filter believes them to be. The landmarks were placed manually and then the position was measured manually as well. In addition to this, they may have moved between the moment where they were placed and when the experiment takes place. This possible inaccurate information is fed to the particle filter which can cause it to make inaccurate estimations.

A further potential cause for the discrepancies could be related to the tuning process of the particle filter. While the filter can estimate the robot's trajectory with reasonable accuracy using the current tuned parameters, further testing and optimization over time could yield optimal values that minimize the error as much as possible.

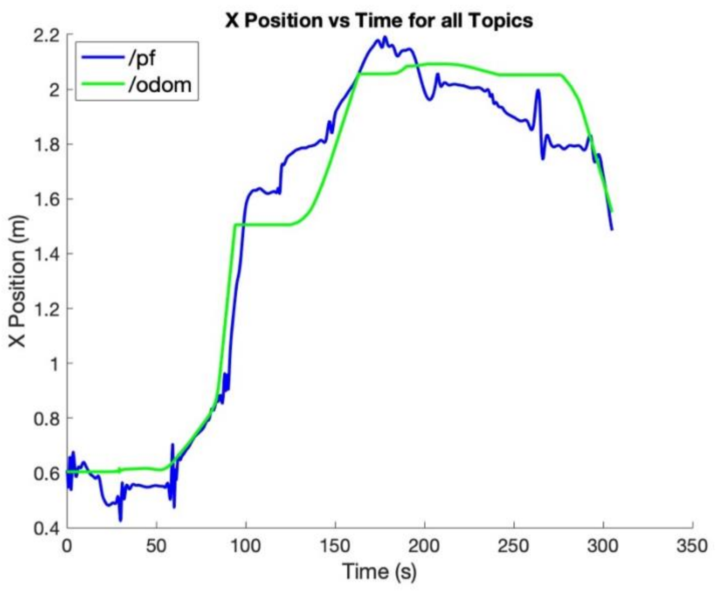

Figure 5. Estimated X-Position vs True Position and On-Board Odometry

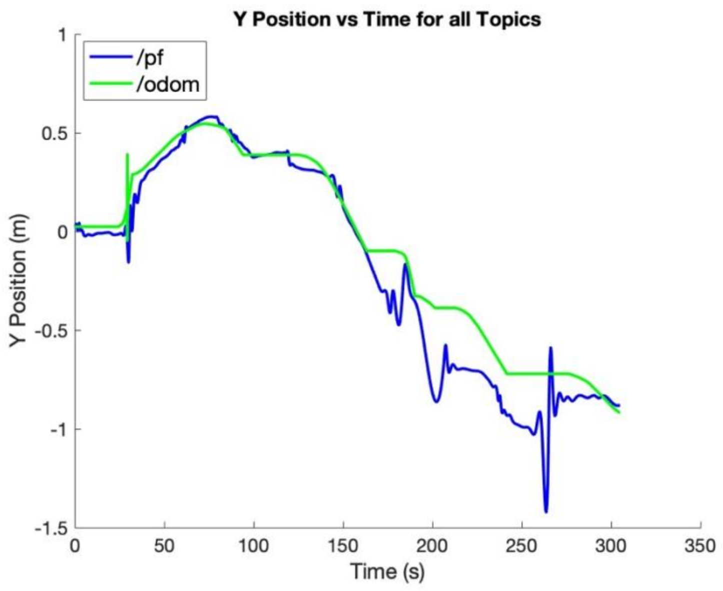

Figure 6. Estimated Y-Position vs On-Board Odometry

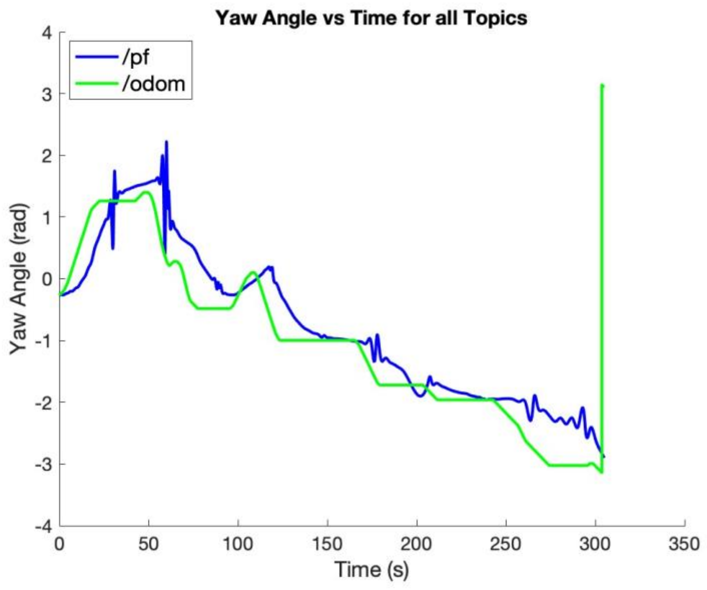

Figure 7. Estimated Yaw Angle vs On-Board Odometry

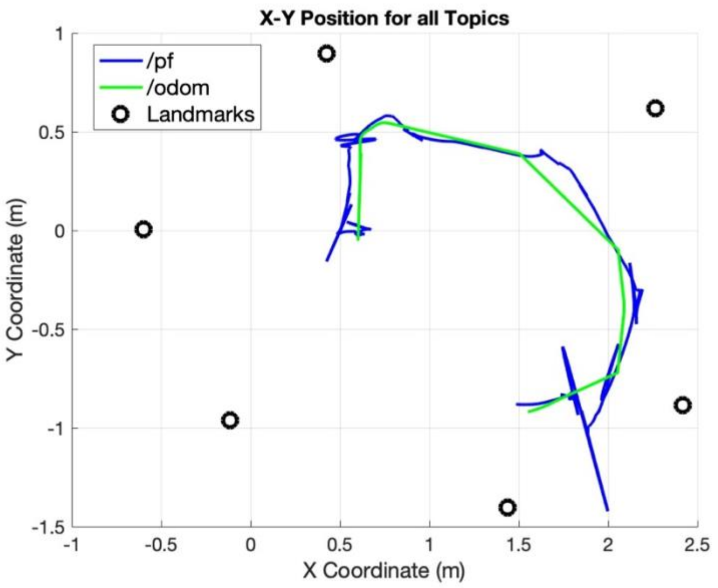

Figure 8. Estimated X-Y Position vs On-Board Odometry