Particle Filter Implementation for Localization

Ignacio Lloret

October 31, 2024

1 Introduction

Localization is a fundamental problem in mobile robotics. This project implements a Particle Filter-based localization algorithm to achieve accurate localization and evaluates its performance against the Adaptive Monte Carlo Localization (AMCL) available in ROS 2.

2 Algorithm Design

2.1 Particle Filter Overview

The Particle Filter algorithm uses a set of particles to represent the probability distribution of the robot's location. Each particle is evaluated through the given function `self.sensor_model.get_weight` this will return a rational value which a score of how likely is the particle the real position.

2.1.1 Motion Update

In this case the model was uncalibrated as each movement alters too much the expected position of the robot which makes the algorithm unusable when moving. Thus the `distance_travelled` variable was divided by 50 to match the map resolution.

2.1.2 Initial Sampling

To sample the initial sample of the particle filter algorithm we used a normal distribution with σ equal to 0.5. This was selected because of its low variance compared to other distributions. Moreover we used 1000 particles that was the maximum before ros2 overran.

2.1.3 Resampling

After each update cycle, additional random points with a t-student distribution (because of its greater volatily) are added (to handle the kidnapped robot problem). Particles are resampled with to ensure that those with higher weights (i.e., better matches to the observed data) are more likely to survive, while others are discarded. As the difference between weights is actually small, maximum values around 8 and low values around 4 this difference is accentuated by using the exponent of this weight, after that normalization is done with the sum of every exponentialed weight.

2.2 Single Pose Estimate

To estimate a single pose from the particle set at each timestep, we use the clustering algorithm called DBSCAN. DBSCAN creates multiple clusters using the resampled data points. We pick the biggest cluster and we compute the mean of the points inside the cluster for the position and orientation.

2.3 Handling the Kidnapped Robot Problem

The kidnapped robot problem is addressed by using the mixture of normal distribution and t-student distribution which creates random points very far away from the initial pose although this creates unstability so in the end couldn't be used. Some other approaches were tried but none had success.

3 Experimental Analysis

3.1 Setup

The experiments were conducted on the simulated environment using the `socspioneer` package and the provided map. The localization accuracy and robustness of the Particle Filter were compared to the AMCL implementation in ROS 2.

3.2 Results

- Localization Accuracy: The accuracy of the algorithm is not great when getting the initialpose near the true pose it has struggles to identify exactly where is it and the orientation.

  The AMCL in contrast does not update from its initial pose so in that sense is better the particle filter algorithm.

- Robustness to Kidnapping: The robustness against the kidnapped robot is not great either. The intend was to add a t-student distribution in resampling but that becames very unstable.

  The AMCL does not prevent the kidnapp either.

- Computational Efficiency: The program was efficient enough to run 100 particles increasing that would prevent the localisation algorithm to not finish in time to predict in real time. Is hard to compare with the AMCL as it works like a black box.

4 Conclusion

In this report, we have implemented and analyzed a Particle Filter for localization, comparing its performance with AMCL. The Particle Filter algorithm was not able to handle the dynamic environment and localization challenges such as the kidnapped robot problem. Some work that should be done and couldn't be done was ensuring all points are inside the map, prevent inestability in resampling.

References

- ROS Documentation: `https://docs.ros.org/`