Udacity project: Where am I?

Lukasz Zmudzinski
University of Warmia and Mazury in Olsztyn
lukasz@zmudzinski.me

Abstract—This paper focuses on describing the process of creating the **Where am I?** Udacity project using Monte Carlo Localization. Moreover localization basics along with Kalman Filters are described. Data on model configuration is then presented with test results.

Index Terms—monte carlo localization, particle filters, localization,

1 Introduction

Localization is one of the main problems when dealing with mobile robots. What for a human seems pretty easy, can get quite complicated for an intelligent agent. Therefore localization algorithms were created, to help in getting over the problem.

In robotics we have three main localization problems: position tracking, global localization and the kidnappend robot problem. We can tackle these challenges using Kalman Filters and Particle Filter localization algorithm (otherwise called Monte Carlo Localization).

This project focuses on creating a system, that will localize a robot inside the Robot Operating System environment using Gazebo and RViz simulation. The project is divided into three tasks:

- Building a mobile robot for simulated tasks,
- Creating a ROS package that launches a custom robot model in a Gazebo world and utilizes packages like AMCL and the Navigation Stack,
- Exploring, adding, and tuning specific parameters corresponding to each package to achieve the best possible localization results.

2 Background

Localization is connected with determining the robot's pose in a mapped environment. It's one of the main challenges in mobile robotics. It is usually done by applying a probabilistic filter to sensor data (almost always containing noise) to track the robot's position and orientation. Localization doesn't come without challenges, we can divide them into three categories: position tracking, global localization and the kidnapped robot problem. There are four popular localization algorithms used nowadays that solve the issues:

- Extended Kalman Filter,
- Markov Localization,
- Grid Localization,
- Monte Carlo Localization.

2.1 Kalman filters

Kalman filters are used because of the fact, that they can process data with a lot of uncertainty or noise and give accurate estimates of the real value quickly. Compared to other algorithms, you don't need a big amount of data, to calculate an accurate estimate. There are three types of Kalman Filters:

- Kalman Filter (KF) - linear,
- Extended Kalman Filter (EKF) - nonlinear,

Fig. 1. Comparison of EKF and MCL parameters.

Parameter	MCL	EKF
Measurements	Raw	Landmarks
Measurement Noise	Any	Gaussian
Posterior	Particles	Gaussian
Memory efficiency	+	++
Time efficiency	+	++
Implementation	++	+
Resolution	+	++
Robustness	++	+
Memory and Resolution Control	Yes	No
Global localization	Yes	No
State Space	Multimodel Discrete	Unimodal Continuous

- Unscented Kalman Filter (UKF) - highly nonlinear.

Kalman filters allow the use of sensor fusion. What it means, is that you can collect data from various onboard sensors, which will give a more accurate value of the estimation.

The Kalman Filter cannot be used on a nonlinear function. Instead, a local linear approximation is used to update the covariance of the estimate. The linear approximation is made using the first terms of the Taylor Series, which includes the first derivative of the function.

2.2 Monte Carlo Localization

Monte Carlo Localization, otherwise called Particle Filter localization algorithm, is a popular algorithm for localization among roboticists. The name comes from the fact, that the algorithm uses particles, that reassamble the robot. Each particle has 4 values: x position, y position, direction (angle) and weight. The MCL algorithm is limited to global and local localizations only (cannot be applied to kidnapped robot problem). MCL takes raw measurements (as compared to landmarks in KF) and can take any type of noise, so it isn't limited to gaussian.

The powerful Monte Carlo localization algorithm estimates the posterior distribution of a robot's position and orientation based on sensory information. This process is known as a recursive Bayes filter.

To understand mobile robot localization, one must know the following definitions:

- **Dynamical system** - the mobile robot and it's environment,
- **State** - The robot's pose, including position and orientation,
- **Measurements** - Perception and odometry data.

The goal of Bayes filtering is to estimate a probability density over the state space conditioned on the measurements. The probability density, or also known as posterior is called the belief and is denoted as $Bel(X_t) = P(X_t|Z_{1...t})$, where X_t is the state at time t and $Z_{1...t}$ are the measurements from time 1 to t.

The MCL algorithm has five steps needed to work:

1) Taking the previous belief of the robot position,
2) Updating the motion data,
3) Updating the measurement,
4) Resampling data,
5) Creating a new belief.

The differences between Extended Kalman filer and Monte Carlo Localization as in Fig. 2.

3 Model configuration

4 Results

5 Discussion

6 Future work

The problem was run into simulation only. Future work could include deploying the system on a mobile robot like Roomba or custom build Raspberry Pi robots and test the algorithms in real life environments like: home or work buildings.

Moreover, more localization algorithms could be tried (or even variations presented in the Background part of the writeup).

Finally, more tests with model configuration parameters could be run, to pick the optimal result for the problem.