Probabilistic Location Estimation

Raihan Seraj

***Abstract*—In this work we perform robot localisation with visual imagery.A Recursive Bayesian Filtering has been used in order to track the robot's belief. In particular a particle filter has been implemented and a detailed analysis is provided regarding the use of such filters.**

I. Introduction

Bayesian Filtering techniques provide a powerful statistical tool to help manage measurements and perform multisensor fusion and identity estimation. Bayesian Filters for Robot localization task involve estimation the position of the robot with respect to the world map and estimates robot's dynamic state from noisy observations. In this work, a particle filter has been implemented in order to estimate the location of the aqua bot from its noisy observation as measured by the odometer. The robot starts from a known initial position and its position is tracked.The performance of the filter is tested using different velocity of the aqua bot and the results are analyzed. In the following section, we address the notion of using a particle filter and the different steps involved in implementing a particle filter.

II. Methodology

A. Particle Filter

Particle filters are extensively used in Bayesian statistical inference problems. The filtering problem consists of estimating the internal states of dynamical systems when partial observations are made.This filtering technique uses genetic-mutation selection approach with a set of particles in order to represent the posterior distribution of some stochastic processes.For the localization task we use a particle filter consisting of 400 particles. When the localization method starts execution, all the particles are randomly spread over the environment and the robots location is unknown. With the noisy sensory measurements being received from the robot's odometer, the weights of each of the particles are adjusted by comparing the actual position of the robot with that of its predicted sensor measurement. Hence the particle with the most weight has a greater chance of being the robots position. The generic particle filtering algorithm has three steps.

1) **Prediction Step:** Each particle is moved by sampling from the motion model of the system
2) **Weighting Step:** The particles are weighted based on how well the sensory measurements corresponds to the expected location of the robot.
3) **Resampling:** The particles are re sampled based on the weights assigned to them in the Weighting step. Particles having smaller weights have less chance of being replicated in the same position, and are thus removed and particle with more weights have a higher chance of being replicated at that particular position and is thus retained.

Advisor: Dipl.–Ing. Firstname Lastname, Lehrstuhl für Nachrichtentechnik, TUM, WS 2050/2051.

B. Algorithm

We further provide an algorithm encompassing the all the three steps.

Particle Filter Algorithm $(\mathcal{X}_{t-1}, u_t, z_t)$;
Input : Three values state $\mathcal{X}$,control u and observation z
$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
for $m = 1$ *to M* **do**
  sample $x_t^{[m]} \sim p(x_t | u_t, x_{t-1}^{[m]})$
  $w_t^{[m]} = p(z_t | x_t^{[m]}$
  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + (x_t^{[m]}, w_t^{[m]})$
end
for $m = 1$ *to M* **do**
  draw i with probability $\propto w_t^{[i]}$
  add $x_t^{[i]}$ to $\mathcal{X}_t$
end
return $\mathcal{X}_t$

Algorithm 1: Particle Filter

III. Implementation and Experiments

The simulation and the experiments are performed using ROS kinetic and Gazebo for the simulator.The turtle bot can be controlled with the keys with different speeds and the corresponding estimate of the location is observed compared with the ground truth. We also compare a variety of turn patterns including circle, eight. For constructing the nodes in ROS we use random from C++ 11. The particles were initialized with equal weights, with states (x,y and yaw) approximately equal to the known starting state for the tracking case. We also performed the experiments of using a particle filter when the robot starts from an initial position which is unknown. In that case the particles were approximately evenly distributed in equidistant cells throughout the map. For the implementation, the motion model was propagated each time a new motion is received. The particles were re sampled at each observation which was a result of a motion from the previous one.

A detailed description of the code base is as follows:

- **Motion Model**: Each of the particles are updated based on the motion callback command with an addition of gaussian noise with zero mean and variance 1.
- **Observation Model**:For each of the particles the conditional probability of observing a particular image from the

camera given the current state is computed as a geometric mean of percent matches of each pixel of the received camera image that would have been received if the robot were at that particles state. The weights of each of the pixel is set as to be inversely proportional to its distance from the center of the image. We compute the percent match between two pixels by the following equation.

$$\frac{e^{(r-r_0)^2+(g-g_0)^2+(b-b_0)^2}}{\sigma^2} \tag{1}$$

Here r,g,b represents the red green and blue pixel values which are of 8 bit with a chosen variance σ^2. The following section addresses the experimental results for the implementation.

IV. Results

The turtle bot provided and the environment is given by the following figure

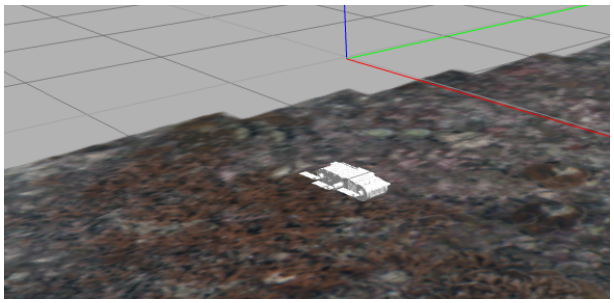
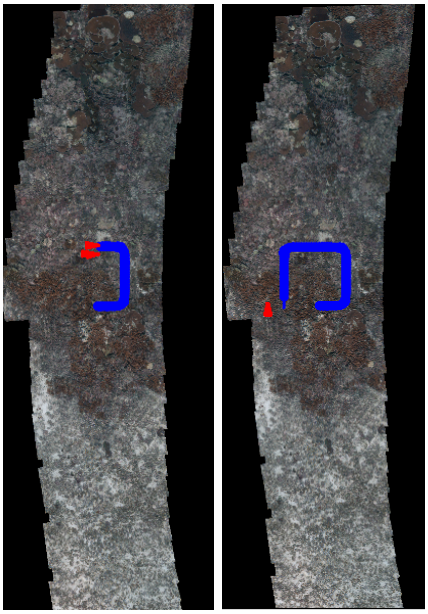


Fig. 1. Gazebo Simulator Environment

We obtained the following results for different trajectory shapes of the robot. The image shows the ground truth values in blue and the location estimated with the particle filter.

Fig. 2. Initialization of Particles

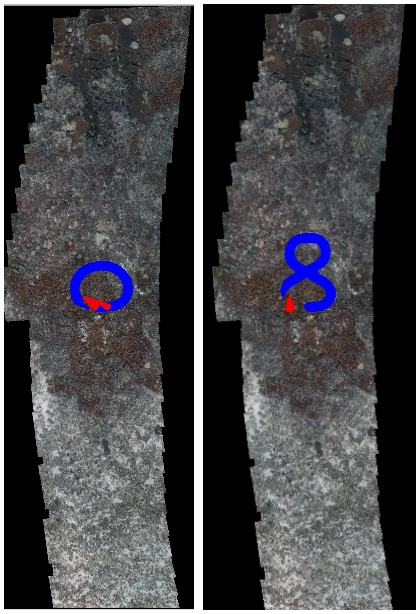
Figure 2 shows the initialization of Particles in the Environment. We start with 400 particles.



(a) (b)

Fig. 3. Particles (in red) tracking the motion of the robot moving in a square pattern

We also try different shapes including circles and the shape of the digit 8 using the ros bag provided. We therefore obtained the following figures



(a) Particles tracking circular motion (b) Particles tracking 'eight' shaped motion

Fig. 4. Particles tracking the motion of the robot moving in a circular and eight pattern

V. Discussion

For this report we have dealt with the problem of keeping a large number of particles without putting a lot of load in the machine. This is done by considering a small central portion of the received camera image. In this work we have assumed that the robot actually turned at a constant rate with an initial delay

which is quite different in practice. Hence if a more robust and suitable translation can be found from a given motion command of the robot. Apart from that in order to get suitable values for the parameters involved in the motion model and the observation model several trial and error methods were used until we reached a suitable value.

In this work we ensured that the particles did not degenerate to undesirable ones by adding noise to the particles initial state in the tracking case. Our simulation shows that the particles' estimation of the robot's location is most erroneous when the robot moves in an 'eight' pattern, this is probably due to the fact that since the turns are sharper and not in a constant manner the particles therefore produces the erroneous result.