2D Robot Motion Estimation using Particle Filter

1st Ali Babaei
Robotics and Computer vision Lab
Ryerson University
Toronto, Canada
ali.babaei@ryerson.ca

***Abstract*—Here is the report of implementation the Particle Filter for a 2D robot motion estimation. Unlike previous reports this document is mostly focused on practical considerations of implementing a particle filter for kinematics of a 2D robot.**

***Index Terms*—Particle Filter, motion estimation, 2D , robot pose**

I. Introduction

he purpose of this assignment is to simulate a robot moving in a circle around a landmark using the Particle Filter algorithm. The same setup is used from previous assignment. The system was simulated using Python.

When dealing non-linearity, one might pass the probability density function through linearized model. The Extended Kalman Filter - EKF is one well-known method to do so. Although the EKF algorithm is effective, but is it is only valid when the distribution of the models and/or noise are approximately Gaussian. However this requires computation of Jacobian and recursive computation which is not feasible in many cases.

Using a finite number of randomly sampled points to compute a result is called a Monte Carlo (MC) method. The idea is simple, which is to generate enough points to get a representative sample of the problem, run the points through the system you are modeling, and then compute the results on the transformed points. In a nutshell this is what particle filtering does.

II. Particle filter

The Particle Filer is an state estimation method that uses the Monte Carlo sampling method. Its main advantages/disadvantages are characterized by the Monte Carlo method itself, in that it's very accurate but the most computationally expensive, so parameters, such as the number of samples, is determined to get the best accuracy/timing tradeoff. The basic PF algorithm can be explained in the following steps:

- **Randomly generate a bunch of particles:** Particles can have position, heading, and/or whatever other state variable you need to estimate. Each has a weight (probability) indicating how likely it matches the actual state of the system. Initialize each with the same weight.
- **Predict next state of the particles:** Move the particles based on how you predict the real system is behaving.
- **Update:** Update the weighting of the particles based on the measurement. Particles that closely match the measurements are weighted higher than particles which don't match the measurements very well.
- **Resample:** Discard highly improbable particle and replace them with copies of the more probable particles.
- **Compute Estimate:** Compute weighted mean and covariance of the set of particles to get a state estimate.

III. Problem setting

We have a two-wheeled robot moving in 2D space. The goal is to circle around a landmark located at [10,10] in Cartesian coordinate system. The robot starts from [15,10] and 90 degrees heading angle.

To calculate the control signal we need to dive deeper into the robot kinematics. The robot model shown in figure 1 is based on ICC - Instantaneous Center of Curvature. ICC is the point around which the robot turns or it can be considered as the center around which the robot moves to form a circle. This point changes with the motion of the robot, hence called as an instantaneous point, true to that specific moment of time. From the figure 1, R is the distance of ICC from the center of the robot. ω is the angular velocity. When the robot is moving in straight line, R will be infinite and will be zero . These parameters will be defined when the robot takes any curvy path, that is, when the left and right wheels have different velocities.

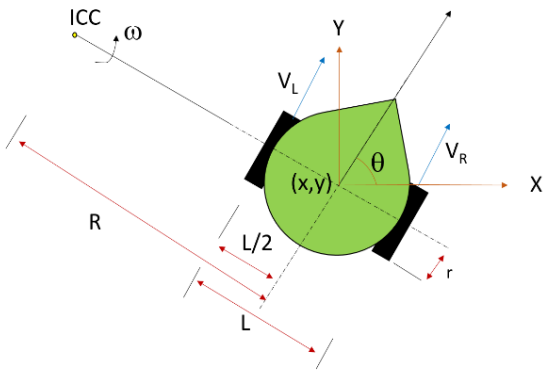

Fig. 1. The ICC concept

Kinematic equations for the mobile robot as per the ICC concept are:

$$V_r = \omega(R + \frac{L}{2})$$

$$V_l = \omega(R - \frac{L}{2})$$

Using the above equations, the individual wheel velocities to the robot, based on the radius equal to 5 and angular velocity equal to $\pi/30$, were calculated. The distance between robot wheels, **L**, also considered to be 0.2 unit. By controlling the angle of the robot, any arc or semi-circle can also be drawn,in the reference plane.

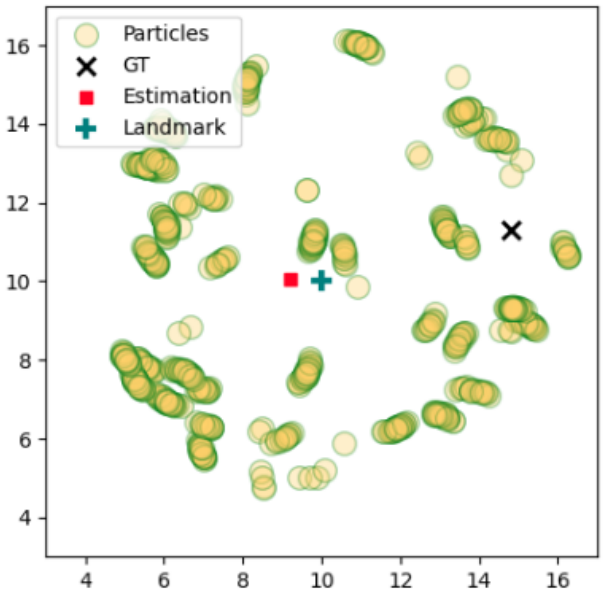

Fig. 2. Estimated 2D pose Using Particle Filter

the implementation and more graphics of the results are available at *https://github.com/AliBabaei0/Estate-Estimation-for-Robotics-self-study*.

The main reference of this work is [1].

REFERENCES

[1] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2017.

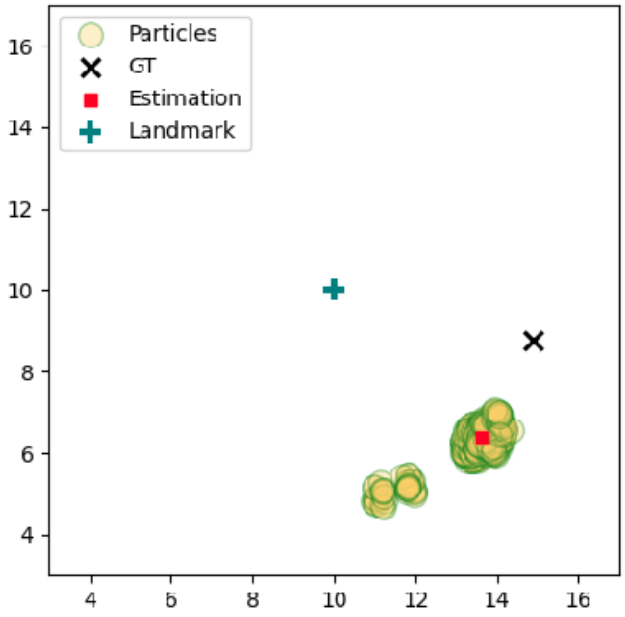

Fig. 3. Estimated 2D pose Using Particle Filter

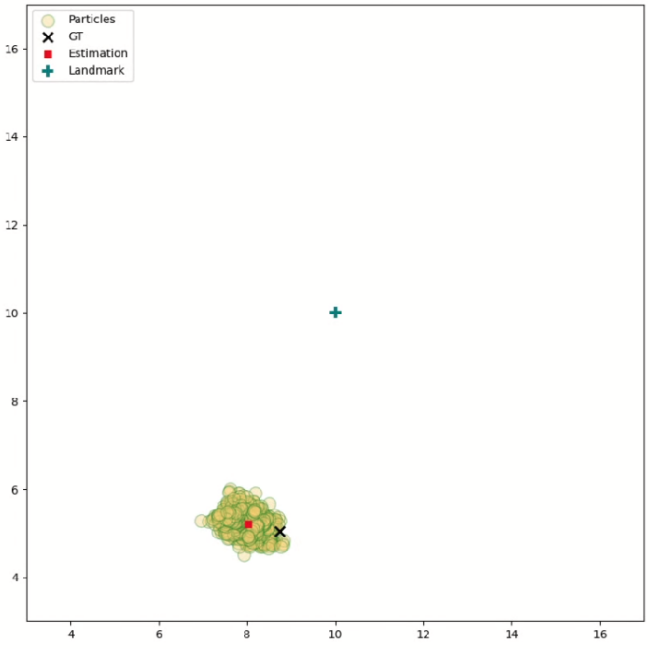

Fig. 4. Estimated 2D pose Using Particle Filter