Localization Problem of a Real Omni Platform

Perception in Robotics Final Project Report

Aleksandr Kashirin
Space and Engineering Systems
Skoltech
Moscow, Russia
Aleksandr.Kashirin@skoltech.ru

Maksim Katerishich
Space and Engineering Systems
Skoltech
Moscow, Russia
Maksim.Katerishich@skoltech.ru

Mert Alper
Space and Engineering Systems
Skoltech
Moscow, Russia
Mert.Alper@skoltech.ru

Sausar Karaf
Space and Engineering Systems
Skoltech
Moscow, Russia
Sausar.Karaf@skoltech.ru

Vladimir Gunyavoy
Space and Engineering Systems
Skoltech
Moscow, Russia
Vladimir.Gunyavoy@skoltech.ru

***Abstract*—This work describes implementation of Extended Kalman filter (EKF) and Particle filter (PF) algorithms for the localization problem of the real omnidirectional robot platform. Localization is a common problem which occurs in the tasks where environment is known preliminary. In this work environment or map is described by the set of landmarks. The goal of the project is a decision on localization algorithm for 'Eurobot' robotics competition. Match in this competition is a match between autonomous robots of two teams, where localization problem is crucial.**

***Keywords*—Extended Kalman Filter(EKF), Particle Filter(PF), Localization problem**

I. Introduction

The goal is to perform localization of 3-wheeled omnidirectional platform in the known environment.

In this work we use robot 'Guido' by team 'RESET' that is built for competition called 'Eurobot'. Figure 1 represents its appearance. Robot has a 3-wheeled omnidirectional wheelbase. Hence, it is considered as a holonomic system in XY-plane and has an ability to move in any direction without or with rotation. 'Guido' has a Lidar 'HOKUYO UST-10LX', IMU sensor MPU6050 and 'Maxon DCX22L' motors with encoders. Thus, 'Guido' has an ability to calculate odometry from 3 different sensors. We use this feature in Particle Filter by fusing the data. 'Guido' utilizes Raspberry Pi 3 as main computer and 'ROS noetic' system.

We also use a preliminary constructed environment with a set of landmarks (beacons) that are represented in figure 2. Three beacons with known positions are located at the borders of the field. These beacons can be observed by the robot at any time. Landmarks are placed in the red circles in the picture for better visualization and understanding.

II. Motion and Sensor Model

A. Odometry Model

Odometry model of the system for readers should be familiar since it copies the odometry model described in the Lecture 5 of the course 'Perception in Robotics' by professor Gonzalo Ferrer [1] with the only difference that axes' motion is independent.

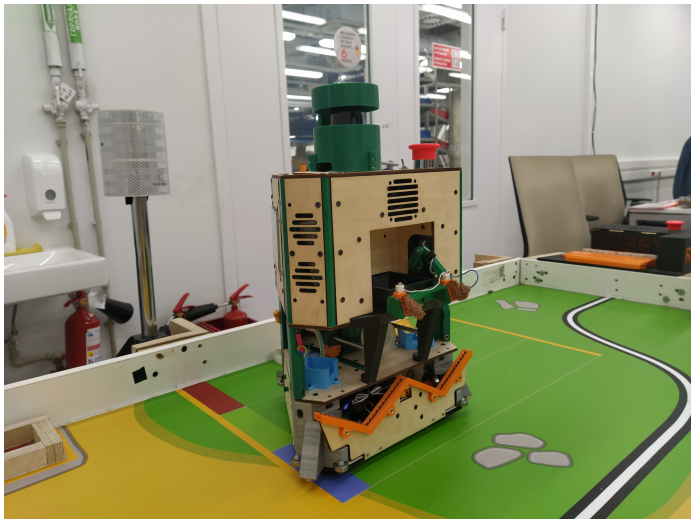

Fig. 1. Robot 'Guido' by 'RESET'

$$X = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{1}$$

Where X is a state vector composed of position x, position y, angle θ.

$$U = \begin{bmatrix} v_x \\ v_y \\ w \end{bmatrix} \cdot \Delta t \rightarrow \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{bmatrix} \tag{2}$$

Where U is a control input vector composed of position displacement Δx, position displacement Δy and angle dis-

Fig. 2. Environment (Beacons are marked in the red circles)

placement $\Delta\theta$. From these equations we can define a transition function $g(x_{t-1}, u_t)$:

$$g(x_{t-1}, u_t) = \begin{bmatrix} x_{t-1} + v_x \cdot \Delta t \\ y_{t-1} + v_y \cdot \Delta t \\ \theta_{t-1} + w \cdot \Delta t \end{bmatrix} \rightarrow \begin{bmatrix} \mu_{x_{t-1}} + \Delta x \\ \mu_{y_{t-1}} + \Delta y \\ \mu_{\theta_{t-1}} + \Delta\theta \end{bmatrix} \tag{3}$$

Where μ is the mean of the filtered coordinate and t is a time step.

In addition, for EKF we consider only wheel odometry in this paper. However, for PF we use data fusion from thress sources mentioned beforehand. Thus, we have several sources of control vector.

B. Sensor Model

In observations we have range, e. g. distance to the landmark and bearing, e. g. relative heading to the landmark measurements:

$$z_t^i = \begin{bmatrix} \phi_t^i \\ r_t^i \end{bmatrix} \tag{4}$$

Where r is a range, ϕ is a bearing, i is a landmark id and t is the time step. Observations have nonlinear nature. For the sake of simplicity a nonlinear transformation between observations and robot state is not provided in this report, but one can always obtain it in Lecture 5: 8.1 Feature-based measurement model [1].

C. Probabilistic System Model

Overall, we can derive the following probabilistic system model:

$$\begin{aligned} X_t &= X_{t-1} + U_t + R \\ Z_t &= h(X_t) + Q \end{aligned} \tag{5}$$

Where X_t is the previous state, U_t is the control input vector, Z_t is the observation vector and $h(X_t)$ is the nonlinear observation function, R is the motion Gaussian noise and Q is the measurement Gaussian noise. By running multiple tests on the real robot, we have estimated noise parameters in order to overcome underestimation and overestimation problems:

$$R = \begin{bmatrix} \alpha_x & 0 & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & \alpha_w \end{bmatrix} \tag{6}$$

where $\alpha_x = \alpha_y = \alpha_w = 1 \cdot 10^{-4}$

$$Q = \begin{bmatrix} 0.015^2 & 0 \\ 0 & 0.03^2 \end{bmatrix} \tag{7}$$

III. Filtering

A. Extended Kalman Filter (EKF)

Since we have a nonlinear measurement model, we use extended part of EKF in the update part only. Thus, we have following equations describing the algorithm:

$$\begin{gathered} \text{Predict step:} \\ 1.\ \bar{\mu}_t = \mu_{t-1} + U_t \\ 2.\ \bar{\Sigma}_t = \Sigma_{t-1} + R \\ \\ \text{Update step:} \\ 3.\ K_t = \bar{\Sigma}_t H^T (H \bar{\Sigma}_t H^T + Q)^{-1} \\ 4.\ \mu_t = \bar{\mu}_t + K_t (Z_t - h(\bar{\mu}_t)) \\ 5.\ \Sigma_t = (I - K_t H_t) \bar{\Sigma}_t \end{gathered} \tag{8}$$

Where H_t is a Jacobian, linearized observation function of the function $h(X_t)$

$$H_t = \frac{\partial h(X_t)}{\partial X_t}\bigg|_{\bar{\mu}_t}$$

$$H_t = \begin{bmatrix} \frac{-(m_{j,x} - \bar{\mu}_{t,x})}{\sqrt{(m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2}} & \frac{-(m_{j,y} - \bar{\mu}_{t,y})}{\sqrt{(m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2}} \\ 0 \\ \frac{m_{j,y} - \bar{\mu}_{t,y}}{(m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2} & \frac{-(m_{j,x} - \bar{\mu}_{t,x})}{(m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2} \\ -1 \end{bmatrix} \tag{9}$$

Where m_j, x, for instance, is the beacon with index j and coordinate x.

B. Data Association

During the implementation process, we have faced a problem of data association. For example, 'Guido' receives several measurements but we do not how do they correspond with landmark ground truth positions. In order to solve this problem, we have written a simple data association algorithm which can be described in the following manner:

1) Estimate the position of the beacon by using the lidar point cloud data and geometric parameters of the beacon.
2) Calculate distance (Euler norm) from estimated position of the beacon to each ground truth beacon location.
3) Find closest ground truth beacon.
4) Assign the id of the ground truth beacon.

In addition, receiving several measurements implies loop cycle inside the update step for each received measurement. We do

not provide the pseudo code in the report, but one can obtain it in the Lecture 6: Algorithm: EKF localization with known correspondences [2].

C. Particle Filter (PF)

PF motion model is based on data fusion of lidar odometry, wheel odometry and IMU odometry, so the control input is a combination of delta's of each measurement channel in respect to each channel weight (not the particles weight!). Propagation of the state is described by the following formula:

$$x_t^{[m]} \sim p(x_t|u_t, x_{t-1}^{[m]}) \tag{10}$$

Where $p := p(x_t|u_t^{wheel}) + p(x_t|u_t^{lidar}) + p(x_t|u_t^{imu})$

Amount of particles related to each distribution is taken by coefficients 0.4 for wheel odometry, 0.2 for lidar odometry, and 0.4 for IMU odometry. Thus we have full distribution $\int p(x) = 1$ where we have more particles of more reliable data source and less particles of less reliable data source.

To calculate the weights, we need to perform data association for beacons, we do it the same way as in EKF section; but in addition, we calculate the difference between known beacon position and each of the associated particles:

$$\begin{matrix} \hat{r}_t^{[m]} = \sqrt{m_{j,x}^2 + m_{j,y}^2} \\ \hat{\phi}_t^{[m]]} = atan2(m_{j,y} + m_{j,x}) \end{matrix} \quad \rightarrow \quad \begin{matrix} \Delta r_t^{[m]} = r_t^{[m]} - \hat{r}_t^{[m]} \\ \Delta \phi_t^{[m]} = \phi_t^{[m]} - \hat{\phi}_t^{[m]} \end{matrix} \tag{11}$$

Where j is known index. After that we use the Cauchy loss function to get weights:

$$w_t^{[m]} \sim p(ln(1 + \Delta\, r_t)|x_t, x_{t-1}^{[m]}) \cdot p(ln(1 + \Delta\, \phi_t)|x_t, x_{t-1}^{[m]}) \tag{12}$$

Then we normalize weights, and resample using low-variance re-sampler algorithm, which one can obtain from Lecture 7: 3.2: Low-variance sampling [3].

D. Ground Truth

We have used the Particle Filter method to obtain ground truth data. We passed into the Particle Filter data from lidar scans, wheel odometry, IMU data and laser odometry. The main difference between the ground truth trajectory and the trajectory given by the Particle Filter is that we have used 10000 particles to get the ground truth data and only 500 particles to localize the robot. Because of this, the program was running slowly, so we have run it offline using rosbag file.

E. Results

In order to prove estimation results, we have provided graphs of the estimation errors of the robot pose. In Fig. 9. we have plotted the estimation errors of the robot pose calculated by the Extended Kalman Filter.

Notice, that as a ground truth data we have use Particle Filter with a 10000 number of particles. By looking in the picture, a reader can ensure that at each time step an estimation error lays within 3σ interval. Which tells us that the EKF works correctly. In Fig. 10 we have depicted estimation errors provided by the PF.

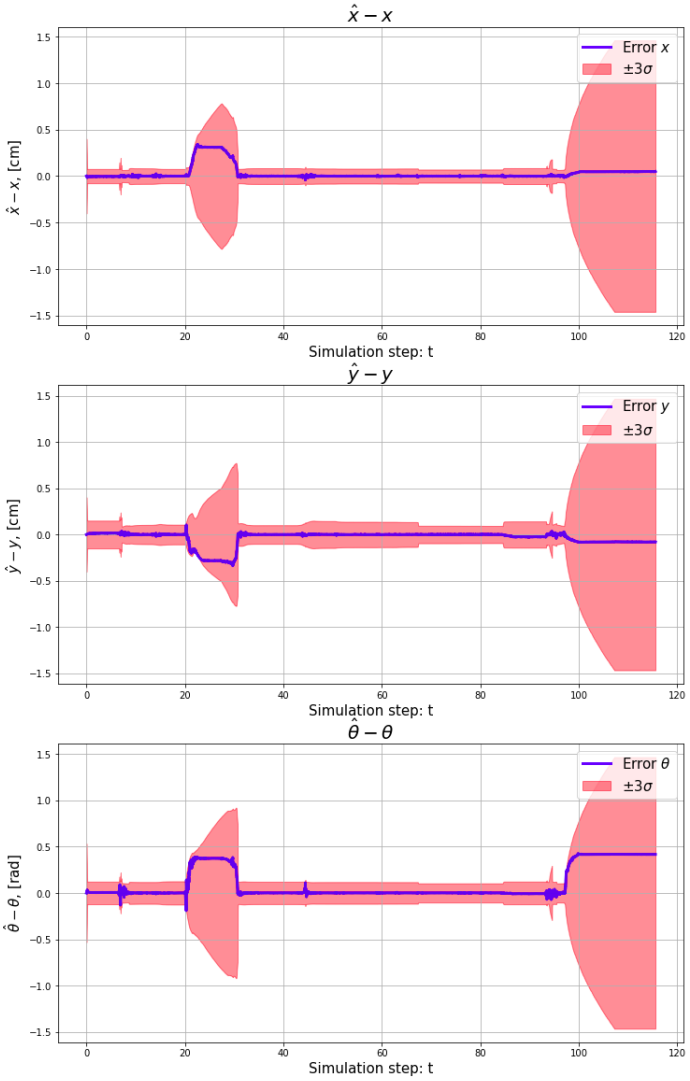

Fig. 3. Extended Kalman Filter: Errors of robot pose estimation $\hat{X} - X$

In this picture, we can notice, that 3σ interval is rather small, hence the filter is more precise.

In addition, to compare the filters we have used three trajectories: the trajectory from the Particle Filter, the trajectory from the EKF and the ground truth trajectory.

In this picture we can see that there are almost no difference between PF and the ground truth trajectory. However, there is a performance difference between the EKF and the PF. There are some intervals where filters' estimations are different. We assume that is due to the fact that the PF fuses the data among different sensors. Hence it is capable to give more precise estimation.

IV. Conclusion

The goal of this project was a research comparison of two types of filter algorithms: Extended Kalman Filter (EKF) with wheel odometry only and Particle Filter (PF) with fused odometry data. This comparison is crucial due to the fact that is used in the real omnidirectional robot 'Guido' by the team

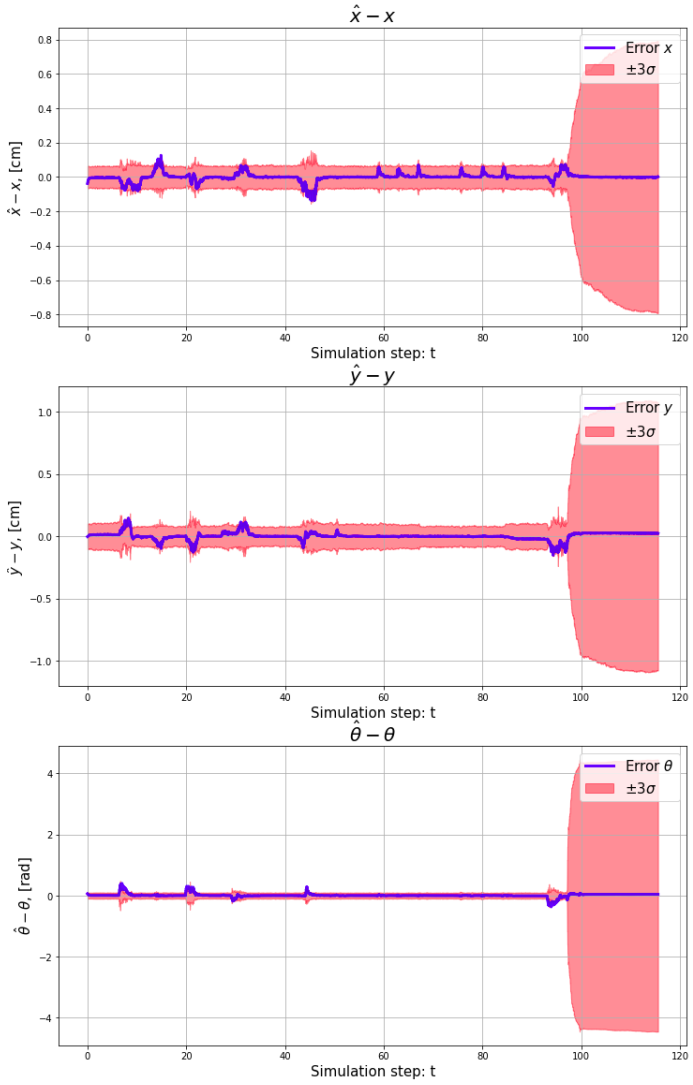

Fig. 4. Particle Filter: Errors of robot pose estimation $\hat{X} - X$

'RESET'. Localization during the match is an important point because if it malfunctions it may cause many problems that will lead to the defeat of the team during the competition. During the tests, we have concluded the PF works better and stable. However, the EKF also has it is benefits such as higher frequency rate (27Hz for the EKF and 25Hz for the PF). In conclusion, the team has decided to pick the PF algorithm for this particular localization problem.

REFERENCES

[1] G. Ferrer 'L05: Motion and sensor model' URL: https://github.com/SkoltechAI/Perception-in-Robotics-course-T3-2022-Skoltech/blob/main/lectures/L05/L05-motion-sensors.pdf

[2] G. Ferrer 'L06: EKF and Localization' URL: https://github.com/SkoltechAI/Perception-in-Robotics-course-T3-2022-Skoltech/blob/main/lectures/L06/L06-localization.pdf

[3] G. Ferrer 'L07: Particle Filter and Monte-Carlo localization' URL: https://github.com/SkoltechAI/Perception-in-Robotics-course-T3-2022-Skoltech/blob/main/lectures/L07/l07-PF.pdf

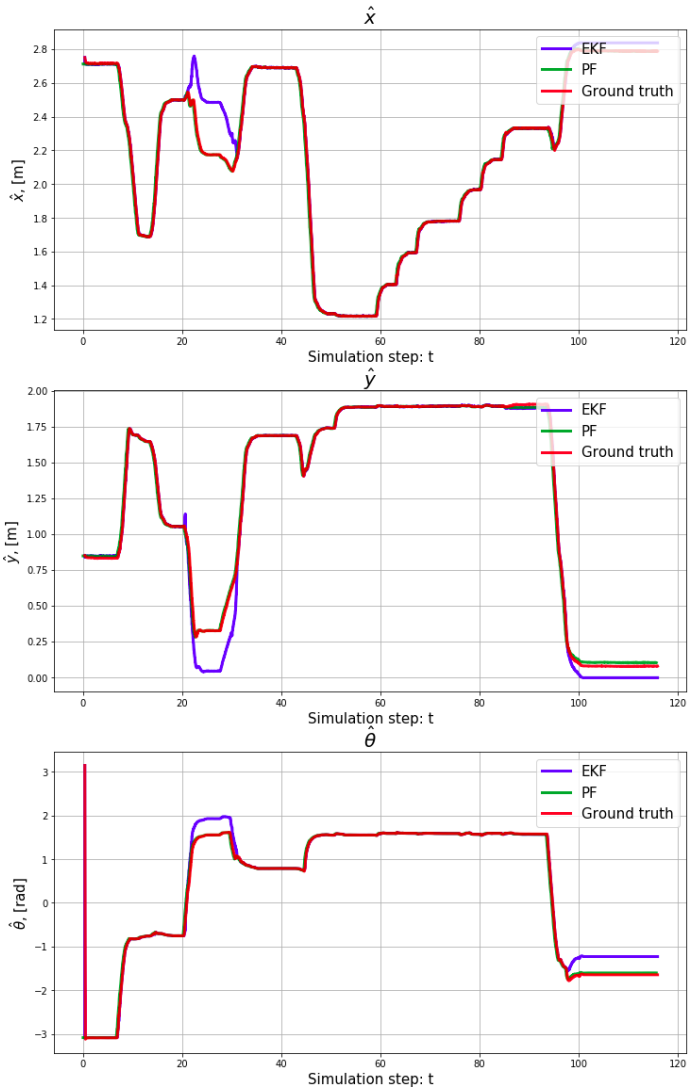

Fig. 5. Comparison of EKF and PF performances