Comparison of Performance of Gaussian Filtering Algorithms for Mobile Robot Localization

Aarish Shah
University of Michigan
AEROSP 567 Final Project

I. INTRODUCTION

According to [1], "Mobile robot localization is the problem of determining the pose of a robot relative to a given map of the environment." In other words, with knowledge of the map of the environment, localization places the robot in the map, with the help of sensor data, motion models and control inputs. Localization is naturally very important for mobile robot navigation. The Gaussian filtering algorithms are one of the categories of algorithms used for mobile robot localization. For this project, I have implemented the Extended Kalman Filter (ExKF) and the Unscented Kalman Filter (UKF) algorithms for mobile robot localization and I compared their performance. Both of these algorithms are Gaussian filtering algorithms, however, ExKF is based on the idea of linearizing whereas UKF is an integration-based Gaussian filtering algorithm.

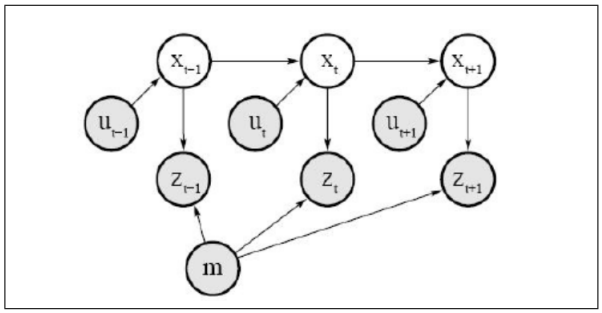


Fig. 1. Graphical model of mobile robot localization from [1]. Shaded nodes are known. x_k represents the state of the robot at time k. u_k represents the control input to the mobile robot at time k. z_k is the sensor data at time k and m is the map of the environment.

I have implemented local, passive localization in a static environment for a single robot. Local localization is the case of localization when the initial pose of the robot is known. Passive localization means that the control law of the robot's motion is not designed specifically for localization; the motion might be random. I have implemented these algorithms on the UTIAS Multi-Robot Cooperative Localization and Mapping Dataset. This dataset contains the following relevant data for 5 robots:

- Odometry: The odometry data has the forward and angular velocity of the mobile robot.
- Measurement data (range and bearing): This contains estimates of distances and angles at which the landmarks were detected by the robot.
- Accurate groundtruth data: The groundtruth data is the true pose of the robot as it moves around in the environment.
- Position and identity of landmarks: The true positions of all the landmarks is provided. Further, each landmark is linked with a unique barcode which the robot detects while obtaining the measurement data. Thus, the robot can recognise the landmark being observed and place itself on the map accordingly.

This dataset includes data for 5 robots. Since I am only performing localization for one robot, all measurement data corresponding to other robots will be ignored. This can be done because like the landmarks, all robots also have a unique barcode assigne to it, hence the robot can detect whether it is detecting a landmark or another robot.

II. DESCRIPTION OF FILTERING ALGORITHMS

Our system is given as follows:

$$X_{k+1} = \Phi(X_k) + \xi \quad \xi \sim N(0, \Sigma)$$

$$Y_k = h(X_k) + \eta \quad \eta \sim N(0, \Gamma)$$

The fundamental filtering equations for this system are:
Prediction:

$$P(X_n|\mathcal{Y}_{n-1}) = \int P(X_n|X_{n-1}P(X_{n-1}|\mathcal{Y}_{n-1})dX_{n-1}$$

Update (using Bayes rule):

$$P(X_n|\mathcal{Y}_n) \propto P(y_n|X_n)P(X_n|\mathcal{Y}_{n-1})$$

The system is non-linear in this case. We make two assumptions to make these integrals easier to compute:

- The predictive distribution is Gaussian.
- The joint distribution $P(X_k, Y_k|\mathcal{Y}_{k-1})$ is Gaussian.

For a general non-linear system, this is not true. Hence, Gaussian filtering is an approximation. Since we are assuming these distributions are Gaussian, we just need to propagate the mean and covariance of the distribution. Under these assumptions, the above equations are simplified to the Gaussian filtering equations. In this project, the Extended Kalman Filter and the Unscented Kalman Filter are discussed, implemented and compared for the mobile robot localization problem.

A. Extended Kalman Filter

ExKF is based on the idea of making a linear approximation to the Gaussian filtering equations using the Taylor series approximation. Basically, ExKF is the standard Kalman Filter applied around a linearized state trajectory. We make the following approximations:

$$\Phi(X_{k-1}) = \Phi(m_{k-1}) + A_k(X_{k-1} - m_{k-1})$$

$$A_k = \nabla\Phi(X_k)|_{X_k=m_{k-1}}$$

The higher order terms of the Taylor expansion are ignored.

Further,

$$h(X_k) = h(m_k^-) + H_k(X_k - m_k^-)$$

$$H_k = \nabla h(X_k)|_{X_k=m_k^-}$$

Here, m_{k-1} is the mean at the prior timestep and m_k^- is the predicted mean at the current timestep. Based on this approximation, the equations simplify as follows:

Prediction:

$$m_k^- = \Phi(m_{k-1})$$

$$C_k^- = A_k C_{k-1} A_k^T + Q$$

Update:

$$\mu = h(m_k^-)$$

$$U = C_k^- H_k^T$$

$$S = H_k C_k^- H_k^T + R$$

$$m_k = m_k^- + US^{-1}(y_k - \mu)$$

$$C_k = C_k^- - US^{-1}U^T$$

where y_k is the data at that timestep.

This filter is easy to implement and is also computationally cheaper compared to integration-based filtering algorithms.

B. Unscented Kalman Filter

UKF aims to solve the Gaussian filtering equations by approximating the integrals by numerical integration (quadrature rules). This usually gives better approximations of the Gaussian filtering equations. Quadrature basically works by replacing an integral with a weighted sum. The Gaussian filtering equations require integration with respect to a Gaussian measure, that is, the integrals are of this form:

$$\int g(X)N(\mu, \Sigma)dX$$

The unscented transform uses a 2d+1 point quadrature rule, where d is the dimension of the Gaussian we are integrating with respect to. UKF has three free parameters, α,β and κ which can be adjusted according to the problem's requirements. The integral is approximated as follows:

$$\int g(X)N(\mu, \Sigma)dX = \sum_{i=0}^{2d} g(\mu + \sqrt{\Sigma}\sqrt{d+\lambda})u^{(i)})w^{(i)}$$

where $u^{(i)} \in [M] \cup u^{(0)}$ as defined in the notes and $w^{(i)}$ are the weights. A unique feature of UKF is that the mean and covariance each have 2 sets of weights. When integrating to find a mean, we use a seperate set of weights. Further in that case, when i=0, we use one weight whereas for the rest of the values of i, we use another. Similarly while integrating to find a covariance, different weights are used when i=0 and otherwise. The weights for mean and covariance are also different. g is any function that we would want to implement. The entire part passed as an argument to g(X) is the ith sigma point. This rule is usually accurate upto third order integrals. Since it is integration based, this filter is more expensive than ExKF.

III. IMPLEMENTATION

A. Robot Motion Model

We have access to odometry data of the robot. The odometry data consists of the velocity and angular velocity of the robot at a given timestep. This data is used to describe the motion of the robot as follows:

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \Phi(X_k) = \begin{bmatrix} x_{k-1} + v_t cos(\theta_{k-1})dt \\ y_{k-1} + v_t sin(\theta_{k-1})dt \\ \theta_{k-1} + \omega_t dt \end{bmatrix}$$

Here, v_k and ω_k represents the velocity and angular velocity of the robot at that timestep. If we simply use the odometry data to predict the robot's motion, we get what is called the "Dead Reckoning" data. Thus, this is the $\Phi(X_{k-1})$ of the robot.

Linearising for ExKF, we get:

$$A_k = \begin{bmatrix} 1 & 0 & -vsin(\theta)dt \\ 0 & 1 & vcos(\theta)dt \\ 0 & 0 & 1 \end{bmatrix}$$

B. Observation Model

The measurement data of the robot consists of four parts:

- The time at which the measurement was taken
- A unique identifying number of the landmark detected
- Distance at which landmark was detected (range)
- Angle at which landmark was detected (bearing)

We know the true location of all the landmarks in the map. Let us assume the ith landmark was detected by the robot with range r and bearing ϕ. Let the landmark's location be denoted by $(m_{i,x}, m_{i,y})$. Let the robot's current pose be (x_r, y_r, θ_r). Then, the robot's prediction of the readings would be:

$$\hat{z} = \begin{bmatrix} \hat{r} \\ \hat{\phi} \end{bmatrix} = h(X_k) = \begin{bmatrix} \sqrt{q} \\ atan2(m_{i,y} - y_r, m_{i,x} - x_r) - \theta_r \end{bmatrix}$$

where $q = (m_{i,x} - x_r)^2 + (m_{i,y} - y_r)^2$

Linearising for ExKF, we get:

$$H_k = \begin{bmatrix} -\frac{m_{i,x}-x_r}{\sqrt{q}} & -\frac{m_{i,y}-y_r}{\sqrt{q}} & 0 \\ \frac{m_{i,y}-y_r}{q} & -\frac{m_{i,x}-x_r}{q} & -1 \end{bmatrix}$$

C. Data Incorporation

Since these filters were implemented on an actual dataset, some problems were faced while incorporating the measurement and odometry data. The timesteps at which odometry and measurement was available was not equal, and neither were the timesteps equally far apart from each other. Thus, to use this data, the following steps had to be taken:

- Check and store timestamps of odometry and measurement data.
- Use a for loop to loop through all of the odometry data. If we are in the ith loop, let's say the odometry data with index i has timestamp t_0 and the odometry data with index i+1 has timestamp t_1.
- Check the timestamps of the measurement data to see if there is any measurement data in the range $[t_0, t_1]$.
  - If not, just use the odometry data with index i to predict the robot's state at t_1.
  - If yes, use odometry data to predict state at t_d, which is the timestamp of the measurement data being considered. Then, at t_d, update the state using the data available. Then use odometry again to predict state at t_1 using the updated distribution.
- Continue this in a loop.

IV. RESULTS AND ANALYSIS

A. ExKF

TABLE I

MEAN SQUARED ERROR (MSE) AND AVERAGE STANDARD DEVIATION (STD) USING EKF

	X	Y	θ
MSE	1.7875	1.889	7.7997
STD	0.00407	0.004068	0.008549

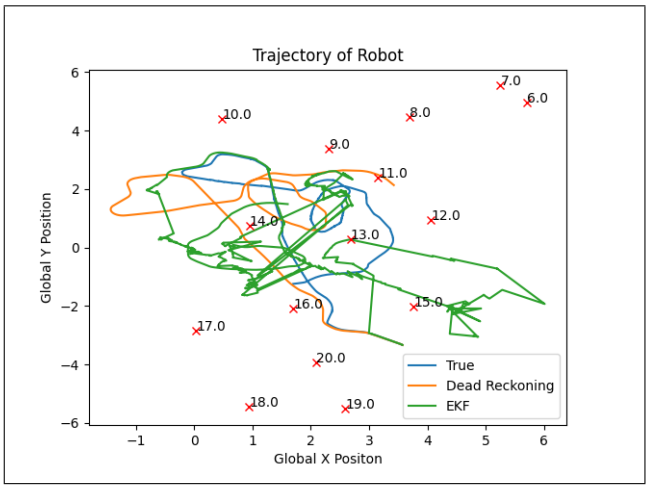


Fig. 2. Trajectory of Mobile Robot using EKF

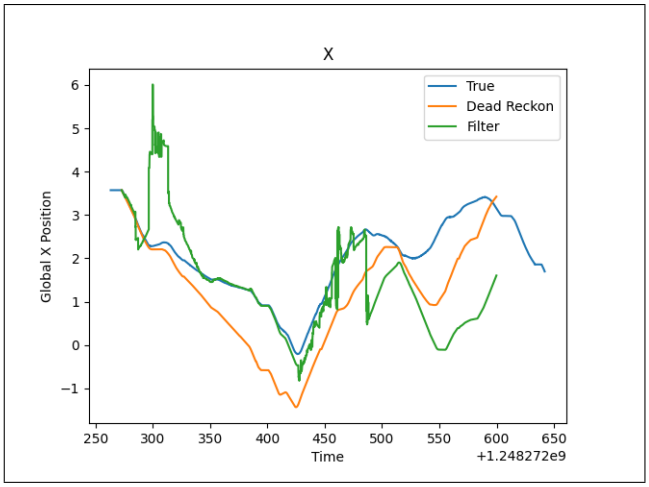


Fig. 3. Global X Position of Robot v/s Time using EKF

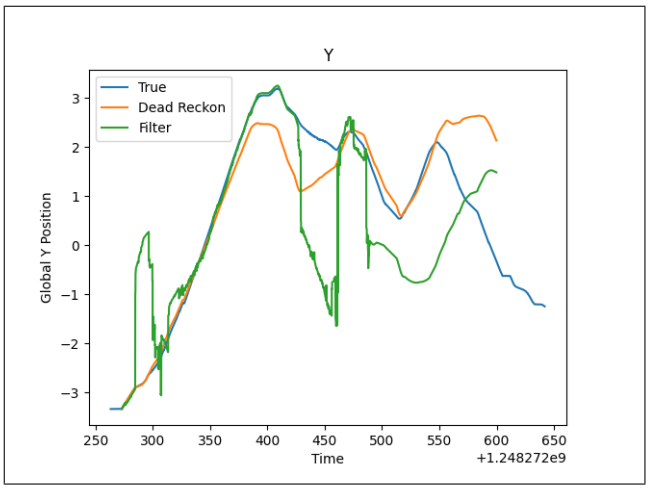
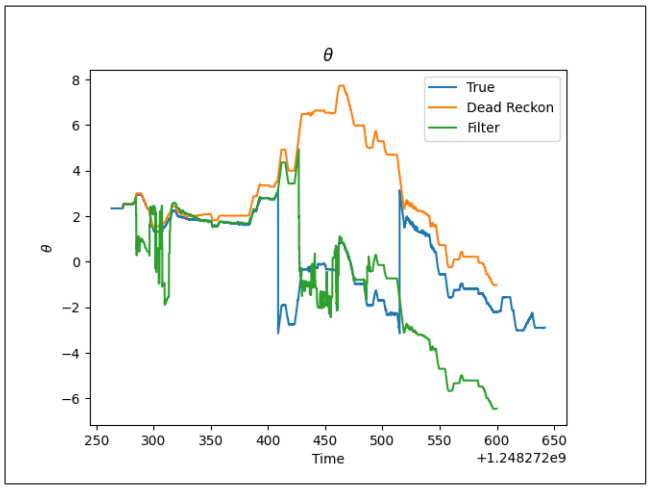


Fig. 4. Global Y Position of Robot v/s Time using EKF



It is clear from these plots that ExKF by itself might not be a suitable filtering algorithm for mobile robot localization. The tracking of the ground truth is quite poor in most places. Sometimes the update step works well and brings the filtering distribution quite close to the ground truth, but only temporarily. The overall trajectory of the filtering distribution is a mess and follows the true trajectory at a few places only. There are a lot of sudden spikes in the trajectory. Further, in the very initial steps, the trajectory almost exactly follows the groundtruth, but then it deviates in a completely wrong direction.

The cause of this might be that the two assumptions that we make for ExKF might not hold true in this case. Firstly, we assumed that filtering distribution is Gaussian, which is not necessarily true for a system like this. Further, we also assumed that the higher order terms in the Taylor series expansion of the Φ and h functions could be ignored. Possibly, these are not valid assumptions under these conditions and hence ExKF fails.

B. UKF

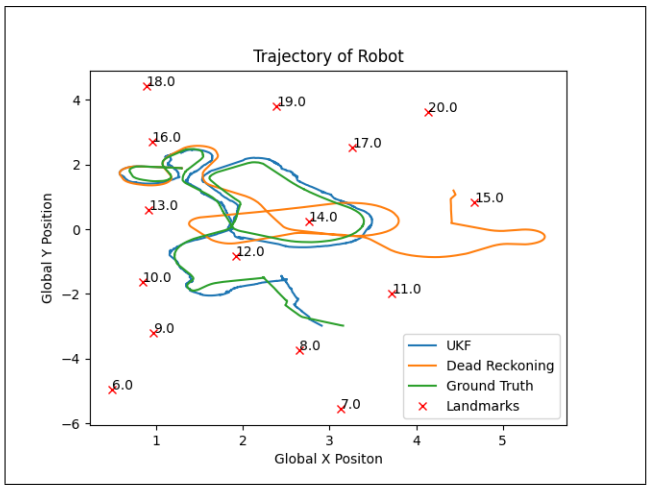


Fig. 6. Trajectory of Mobile Robot using UKF

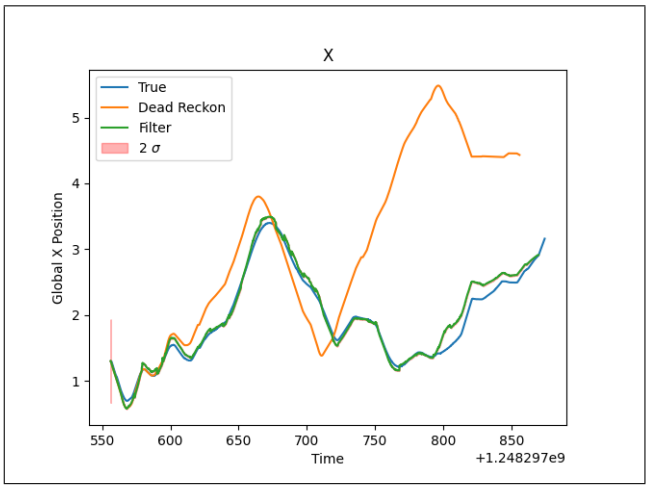


Fig. 7. Global X Position of Robot v/s Time using UKF

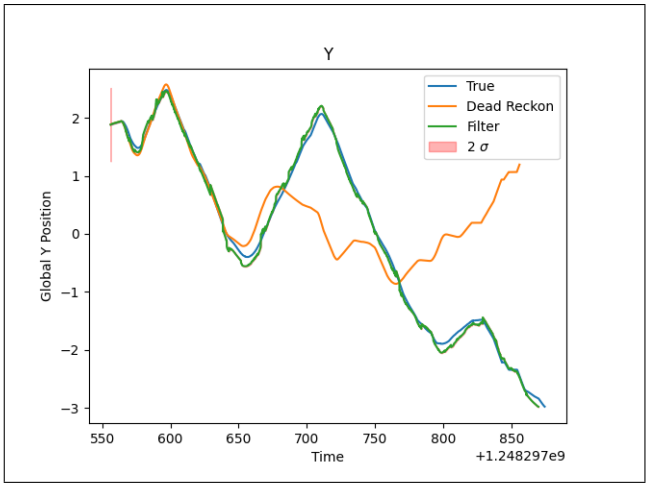


Fig. 8. Global Y Position of Robot v/s Time using UKF

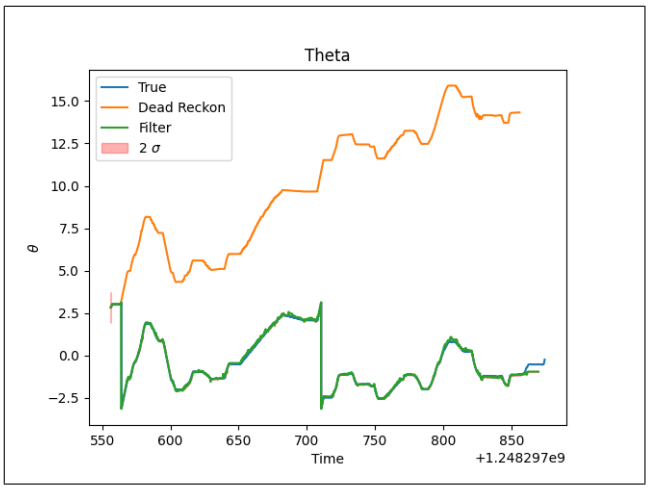


Fig. 9. θ of Robot v/s Time using UKF

TABLE II
MEAN SQUARED ERROR (MSE) AND AVERAGE STANDARD DEVIATION (STD) USING UKF

	X	Y	θ
MSE	0.0119	0.0078	0.0571
STD	0.004	0.004	0.0083

It is clearly visible from these plots that UKF does a much better job at the tracking the groundtruth than ExKF. The Dead Reckoning distribution is way off course, but using the sensor measurements, UKF corrects itself and localizes the robot fairly accurately in the map. We can see that towards the end of the path, the dead reckoning is in a completely different part of the map but the filtering distribution still tracks the groundtruth fairly accurately. The MSE in UKF is also quite low, showing that the tracking is fairly accurate.

V. CONCLUSION

The results of this project indicate ExKF by itself might not very suitable for localization of mobile robots. This concurs with [1], which states that ExKF tends to be brittle sometimes, and this can be solved by adding another layer: measurement likelihood. In this step, the likelihood of any future measurement given information of the past states and measurements is computed. Only landmarks which pass a certain threshold of likelihood are considered for the update step and the rest are rejected. Implementing this would possibly make ExKF better and useful for some localization applications.

UKF seemed to have worked quite well. The MSE was still considerable, but these levels of error might be acceptable in some scenarios, especially in cases where limitations on computational power prohibit application of the Particle Filter. Thus, we see that the Gaussian approximation is not absolutely off; UKF achieved an acceptable approximation.

REFERENCES

[1] Thrun, S., Burgard, W.,, Fox, D. (2005). Probabilistic robotics.. MIT Press. ISBN: 978-0-262-20162-9

[2] Lecture Notes

[3] https://github.com/bostoncleek/Unscented-Kalman-Filter

[4] Leung K Y K, Halpern Y, Barfoot T D, and Liu H H T. “The UTIAS Multi-Robot Cooperative Localization and Mapping Dataset”. International Journal of Robotics Research, 30(8):969–974, July 2011.