A simple implementation of EKF-SLAM in MATLAB for a simulated robot with finite Range and Bearing Sensor

Subodh Mishra[1] and Chiranjeev Ghosh[2]

***Abstract*— In this report, we have described and presented results of a very commonly used SLAM framework called the EKF-SLAM used for state estimation and map building in mobile robotics. We have drawn inspiration from an online course offered by Dr. Joan Sola and used his code framework in our work. Although the basic structure of the code is same as his implementation, we have modified the sensor model and unlike his sensor, ours has a finite range and bearing, so it cannot observe at all the landmarks simultaneously. This poses extra challenges and requires meticulous understanding of the SLAM problem and is closer to the real world than the original code frame work explained in Dr. Sola's course. We have described in succinct cogency what SLAM is, what are the different ways of doing it and the advantages and disadvantages of the different techniques. Finally, we have explained in detail the EKF-SLAM technique and presented the results obtained from our simple simulation formulation.**

I. INTRODUCTION

Simultaneous Localisation and Mapping or SLAM as it is better known in the robotics fraternity, is an important problem in the field of Mobile and Field Robotics. The SLAM problem asks an important question pertaining to build completely autonomous agents. Consider a robot left on its own in an unknown environment, will it be able to simultaneously know where it is with respect to a consistent and incrementally self built map? To know where it is, it needs to build a map and to build a map it has to know where it is, so it is like a chicken egg problem and the solution to the SLAM problem has been seen as a "holy grail" among the mobile robotics community [1].

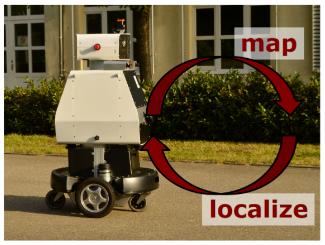

Fig. 1: Navigation using SLAM

SLAM is a problem that all living beings solve unconsciously. It is the problem of spatial exploration. When we enter a completely unknown environment, we look around and observe and then move, making a mental map of our spatial surroundings. Then we plan our actions according to this mental map we have built. For example, to reach a coffee machine, we first recall this map in our brain which contains the position of the coffee machine and we execute our motion commands(i.e. walk) until we reach the coffee machine. In cases when we don't know about the environment we explore, i.e. we search for the coffee machine and as we search we build a mental map and at the same time localise ourselves with respect to this map. For living beings this mental map is more in terms of semantics, if someone asks us where the coffee machine is, we answer in terms of its location w.r.t. another landmark and not in its Cartesian location or Longitudes and Latitudes. Although doing this mentally is quite trivial, implementing this capability to make robots completely autonomously is very complex task.

While doing semantic SLAM is a big challenge, researchers have had tremendous success in doing geometric SLAM (i.e. in terms of Cartesian Co-ordinates or Lat-Long) with considerable precision and consistency. There are different approaches of doing SLAM once it is posed in a probabilistic framework using the ubiquitous Bayes Theorem. The four major paradigms of doing SLAM are Kalman Filter based , Information Filter based, Particle Filter based and Graph based approaches. Each having their own set of advantages and disadvantages.

The first ever SLAM algorithm was written using a Kalman Filter based approach. While a general Kalman Filter(KF) can be used only with linear systems, with non linear systems an Extended Kalman Filter(EKF) is used, which may not prove to be efficacious if the amount of non linearity is large. The solution can easily diverge under such circumstances. KF based approaches have very high computational complexity which increases with the number of landmarks mapped by the robot so it cannot be used for large scale SLAM. Besides this KF approaches assume a Gaussian distribution for the state variable which may not hold good under all circumstances. The major advantage with KF based SLAM is that it is easy to formulate and implement.

Sparse Extended Information Filter (SEIF) approaches can also be used like EKF after linearizing the system about an operating point. The advantage of SEIF SLAM is that it has constant time complexity which does not depend on the number of landmarks to be mapped. Hence, unlike EKF SLAM, it can be used for large scale SLAM. The disadvantages with this approach is that like EKF it assumes

*This report was submitted for a classroon project on Estimation and Detection Theory at Texas A& M University for the course ECEN-662 offered on Spring 2018.

[1]S. Mishra is with the Department of Mechanical Engineering, Texas A & M University, College Station, USA. subodh514@tamu.edu

[2]C. Ghosh is with the Department of Electrical and Computer Engineering, Texas A & M University, College Station, USA. chiranjeev.ghosh@tamu.edu

a Gaussian distribution for the state variable and also unlike EKF it is difficult to formulate and implement.

Graph based approaches to the SLAM problem are the state of the art and primarily use the method of least squares. The complexity is linear in the number of edges which is not as high as the complexity with the EKF approach. Even this approach assumes a Gaussian distribution for its state variable but it is robust to outliers. Unlike the EKF approach where the linearisation of the system is done only once before the iteration, this approach linearises the system at each iteration, hence it does not diverge as much as EKF. It is very much suitable for large scale SLAM.

Particle Filter (PF) based approaches hold good for systems with states belonging to any kind of probability distribution and there is no need to linearize the system. The time complexity is logarithmic in the number of landmarks, hence it is not as computationally intensive as KF/EKF based approaches which makes it suitable for large scale SLAM.

In theory, SLAM is a solved problem and it has been applied in a plethora of environments, ranging from indoors to outdoor, from aerial to underwater, in mining application to survey and exploration of coral reefs. Nevertheless, many challenges still remain to be paid heed to. Owing to sensor limitations and limitation of computation power specific to robotic platforms SLAM frameworks are not generic and one SLAM pipeline which may perform jolly well underwater may not work at all on land and similarly a SLAM framework which may work like a charm indoors may not work outdoors. So, much of the problems concerning SLAM are not theoretical and rather practical in nature. Every environment poses a different challenges and a sensor which may work in one environment may not work at all in another.

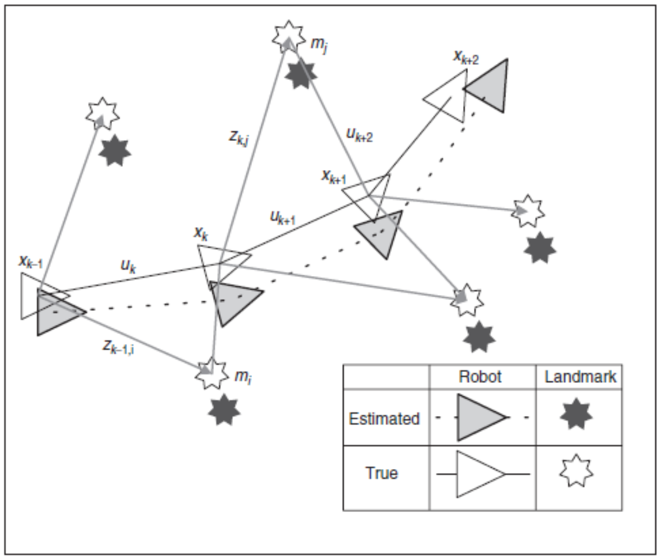

Fig. 2: The essential SLAM problem. A simultaneously estimate of both robot and landmark locations is required. The true locations are never known or measured directly. Observations are made between true robot and landmark locations. [1]

II. Probabilistic Formulation of the SLAM problem

SLAM is the process by which the mobile robot estimates it's own location with respect to a map that it incrementally builds. Both the robot trajectory and the landmark locations are estimated online without any a priori knowledge of location.

A. Notation

Let us consider a mobile robot moving in the environment taking observations using a sensor located in the robot as shown in Figure 2. At any time instance k, let us define the following quantities:

- $\mathbf{x}_k$: The state vector describing location and orientation of the vehicle.
- $\mathbf{u}_k$: The control vector applied at time $k-1$ to drive the vehicle to a state $\mathbf{x}_k$ at time k.
- $\mathbf{m}_i$: a vector describing the location of the i^{th} landmark whose true location is assumed time invariant.
- $\mathbf{z}_{ik}$: An observation taken from the vehicle of the location of the i^{th} landmark at time k. When there are multiple landmark observations at any one time or when the specific landmark is not relevant to the discussion, the observation will be written simply as $\mathbf{z}_k$.
- $\mathbf{X}_{0:k} = \{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_k\} = \{\mathbf{X}_{0:k-1}, \mathbf{x}_k\}$: the history of vehicle locations.
- $\mathbf{U}_{1:k} = \{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_k\} = \{\mathbf{U}_{1:k-1}, \mathbf{u}_k\}$: the history of control inputs.
- $\mathbf{m} = \{\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_n\}$: the set of all landmarks.
- $\mathbf{Z}_{1:k} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_k\} = \{\mathbf{Z}_{0:k-1}, \mathbf{z}_k\}$: the set of all landmark observations.

B. Probabilistic Modelling of the SLAM problem

In probabilistic framework, the SLAM problem requires a closed form solution to the following probability distribution

$$P(\mathbf{x}_k, \mathbf{m} | \mathbf{Z}_{1:k}, \mathbf{U}_{1:k}, \mathbf{x}_0) \tag{1}$$

at all time instance k. The probability distribution describes the probability that the robot is at x_k and the landmarks are at m given the history of measurements $\mathbf{Z}_{1:k}$, the history of control inputs $\mathbf{U}_{1:k}$ and the initial position $\mathbf{x}_0$. In general, every robot has two basic functions, the first being an action which it undertakes to change the state and second is a measurement which it undertakes to get an estimate of the current state. An action always increases the uncertainty associated with the state and a measurement always decreases the uncertainty associated with the state. There is no defined order in which action and measurement can occur but here for the sake of simplicity and without any lack of generality we can assume that an action occurs first and the measurement occurs next.

We now go on to define the action and the measurement probability distributions. The action model is defined as:

$$P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) \tag{2}$$

and the measurement model is defined as:

$$P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m}) \tag{3}$$

The action model is also called probability distribution of the state transition and is assumed to be a Markov process because the current state $\mathbf{x}_k$ depends only on the previous state $\mathbf{x}_{k-1}$ and the current control/action input $\mathbf{u}_k$. Similarly, the equation 3 is the probability distribution of the measurement model where $\mathbf{z}_k$ is a measurement.

The SLAM algorithm is now implemented as a standard two step recursive/sequential prediction(action) and correction(measurement) form:

- **Action:**

$$\begin{aligned} P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{1:k-1}, \mathbf{U}_{1:k}, \mathbf{x}_0) = \int P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) \\ \times P(\mathbf{x}_{k-1}, \mathbf{m}|\mathbf{Z}_{1:k-1}, \mathbf{U}_{1:k-1}, \mathbf{x}_0) d\mathbf{x}_{k-1} \end{aligned} \tag{4}$$

- **Measurement:**

$$\begin{aligned} P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{1:k}, \mathbf{U}_{1:k}, \mathbf{x}_0) = \\ \frac{P(\mathbf{z}_k|\mathbf{x}_k, m) P(\mathbf{x}_k, \mathbf{m}|\mathbf{Z}_{1:k-1}, \mathbf{U}_{1:k}, \mathbf{x}_0)}{P(\mathbf{z}_k|\mathbf{Z}_{0:k-1}, \mathbf{U}_{0:k})} \end{aligned} \tag{5}$$

C. Solution to the SLAM problem

In order to find a probabilistic solution to the SLAM problem we need to find a mathematical representation for the action/motion model 2 and the measurement model 3 and then use them in 4 and 5 to obtain 1. And these representations may vary with the type of framework we want to use to solve the SLAM problem. The EKF based approach is easy to understand and implement, so it will be the topic of discussion for the rest of the report. The EKF based approach is by far the simplest and most common representation of the SLAM problem and it uses state space model with additive noise to model the action and the measurement models.

III. Extended Kalman Filter based Simultaneous Localisation and Mapping (EKF-SLAM)

A. Preliminaries

We want to mention the fact that irrespective of the framework adopted, the procedure to solve SLAM has three basic operations:

- **The robot moves,** from $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$ by executing a motion command $\mathbf{u}_k$ at time step k. This is modelled as the *motion model*. Due to noisy control input, the motion is not perfect and it may not exactly reach where it should actually reach if the perfect noiseless control input were executed. This operation increases the system uncertainty.
- **The robot discovers new landmarks in the environment,** which are added to the state vector. Because of errors in the sensor measurements, the location of these landmarks will not be perfectly correct and there will be a certain amount of uncertainty associated with it. Moreover, as the robot location is uncertain, these two uncertainties need to be properly composed. Since the state vector contains the Cartesian coordinates of the landmark in some frame, we need a *inverse observation model* which takes the measurement and throws out the location of the landmark in cartesian coordinates. This operation augments the state vector and the corresponding covariance terms.
- **The robot re-observes** the landmarks which are already in the state vector (note that here the problem of data association comes into play, i.e. How will the robot recognise a landmark which is already there in the state vector?) and uses them to correct both its self-localisation and the localisation of all the landmarks in space. In this case, both the robot's and the landmark uncertainties decrease. Here the *direct observation model* is used to predict the value of measurement from the predicted landmark location and the robot localisation. The error between the predicted measurement and the true measurement drives the correction step. This operation reduces the uncertainty associated with the robot and the reobserved landmarks.

Using appropriate mathematical models for the above operation and a estimator, like EKF, one can solve the SLAM problem.

Mathematically the probabilistic formulation of motion model translates to an equation in the following manner,

$$P(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k) \iff \mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_k \tag{6}$$

Here $f()$ is the motion model of the robot and $\mathbf{w}_k$ is a zero mean Gaussian noise with covariance $\mathbf{Q}$. In a manner similar to the one aforementioned, the direct observation or the measurement model also translates into an equation as follows,

$$P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m}) \iff \mathbf{z}_k = \mathbf{h}(\mathbf{x}_{k-1}, \mathbf{m}) + \mathbf{v}_k \tag{7}$$

Here h() is the direct measurement model of the sensor mounted on a robot $\mathbf{v}_k$ is the measurement noise which is assumed to be zero mean with error covariance equal to $\mathbf{R}$.

B. Implementation Details

In this report, we simulate a robot in plane which takes velocity inputs. So, we use a velocity model to keep things simple. In this framework we fix the number of landmarks. The robot is endowed with a range and bearing sensor (in practice it can be a LiDAR or a stereo camera).

If the robot sees a landmark, the the measurements are in the following form:

$$\begin{bmatrix} \mathbf{d} \\ \theta \end{bmatrix} = \begin{bmatrix} \sqrt{\mathbf{x}^2 + \mathbf{y}^2} \\ \tan^{-1} \frac{\mathbf{y}}{\mathbf{x}} \end{bmatrix} \tag{8}$$

Here x and y are defined in the robot frame. Hence, to design an *observation/measurement model* one need to express the landmark position in the the robot frame and not the global frame.

The state variable of the system consists of the robot pose in the plane and the positions of the landmarks in the plane. So, if N landmarks have been mapped, the state dimension would be $3 + 2N$ because the robot pose is defined by its x, y and ϕ and each landmark has its x and y coordinates. We can define the state vector at time k is defined as $\mathbf{X}_k = [\mathbf{x}_k^{rT}, \mathbf{x}_1, \mathbf{y}_1, \mathbf{x}_2, \mathbf{y}_2, ..., \mathbf{x}_N, \mathbf{y}_N]^T$

and the state covariance can be defined as:

$$\mathbf{P}_{X_k X_k} = \begin{bmatrix} \mathbf{P}_{x_k^r x_k^r} & \mathbf{P}_{x_k^r M} \\ \mathbf{P}_{M x_k^r} & \mathbf{P}_{M_k M_k} \end{bmatrix}$$

Where M_k refers to the mapped landmarks.

1) Prediction:

$$\mathbf{x}_k^r = \mathbf{f}(\mathbf{x}_{k-1}^r, \mathbf{u}_k) \tag{9}$$
$$\mathbf{P}_{x_k^r x_k^r} = \mathbf{F}_{x_k^r} \mathbf{P}_{x_{k-1}^r x_{k-1}^r} \mathbf{F}_{x_k^r}^T + \mathbf{F}_{u_k} \mathbf{Q} \mathbf{F}_{u_k}^T \tag{10}$$
$$\mathbf{P}_{x_k^r M_k} = \mathbf{F}_{x_k^r} \mathbf{P}_{M_k x_{k-1}^r} \tag{11}$$
$$\mathbf{P}_{M_k x_k^r} = \mathbf{P}_{x_k^r M_k}^T \tag{12}$$

The prediction step is performed when a control command u_k is executed. As can be seen in equation 10, the covariance associated with the robot pose $\mathbf{P}_{x_k^r x_k^r}$ increases in this step and if one continues to execute control signals for a long time without taking any measurements, the covariance will increase drastically.

2) Observation of Mapped Landmarks: When the robot's sensor sees a landmark that it had already mapped, i.e. a landmark which is already present in the state vector, it uses this observation information to correct the robot pose prediction and also the location of the re-observed landmark using the predicted observation. The predicted observation is obtained by the robot's predicted position obtained in the aforementioned *Prediction* step and the landmark's position which is stored in the state variable.

$$\mathbf{z}_k = \mathbf{y}_k - \mathbf{h}(\mathbf{X}_k) \tag{13}$$
$$\mathbf{Z}_k = \mathbf{H}_{X_k} \mathbf{P}_{X_k X_k} \mathbf{H}_{X_k}^T + R \tag{14}$$
$$\mathbf{K} = \mathbf{P}_{X_k X_k} \mathbf{H}_{X_k}^T \mathbf{Z}_k^{-1} \tag{15}$$
$$\mathbf{X}_k = \mathbf{X}_k + \mathbf{K}\mathbf{z}_k \tag{16}$$
$$\mathbf{P}_{X_k X_k} = \mathbf{P}_{X_k X_k} - \mathbf{K}\mathbf{Z}_k\mathbf{K}^T \tag{17}$$

In equation 13 the difference between the actual measurement $\mathbf{y}_k$ and the predicted measurement $\mathbf{h}(\mathbf{X}_k)$ is taken by using the *measurement model* $\mathbf{h}()$ and the result of the *prediction step*, $\mathbf{X}_k$. This difference is also called the innovation. Here, $\mathbf{H}_{X_k}$ is the Jacobian of the *measurement model* with respect to the state vector. Then in equation 14 the innovation covariance $\mathbf{Z}_k$ is calculated which is utilised in the equation 15 to obtain the *Kalman Gain* $\mathbf{K}$. Finally, in equations 16 and 17 the state vector $\mathbf{X}_k$ and the associated state covariance matrix $\mathbf{P}_{X_k X_k}$ are updated by using the *Kalman Gain* $\mathbf{K}$. Furthur details of implementation can be found in [2].

3) Observation of a Landmark for the first time, i.e. Landmark intialisation: Landmark initialisation occurs when the robot discovers landmarks that are not yet mapped and decides to incorporate them into the state vector. So, this particular operation results in an augmentation of the state vector and the state covariance matrix. So, one must realise here that the state vector in EKF-SLAM is one with a dynamically changing size. When the robot observes a new landmark we use a *inverse observation model* to find the location of the landmark in the global coordinates using the robot's current pose $\mathbf{x}_k^r$ and the landmark measurement from the sensor $\mathbf{y}_k$. Let us denote the new landmark position as $\mathbf{L}_{N+1}$. The following equations can be written:

$$\mathbf{L}_{N+1} = \mathbf{h}^{-1}(\mathbf{x}_k^r, \mathbf{y}_k) \tag{18}$$
$$\mathbf{G}_{\mathbf{x}_k^r} = \frac{\partial \mathbf{L}_{N+1}}{\partial \mathbf{x}_k^r} \tag{19}$$
$$\mathbf{G}_{\mathbf{y}_k} = \frac{\partial \mathbf{L}_{N+1}}{\partial \mathbf{y}_k} \tag{20}$$
$$\mathbf{P}_{LL} = \mathbf{G}_{\mathbf{x}_k^r} \mathbf{P}_{x_k x_k} \mathbf{G}_{\mathbf{x}_k^r}^T + \mathbf{G}_{\mathbf{y}_k} \mathbf{R} \mathbf{G}_{\mathbf{y}_k}^T \tag{21}$$
$$\mathbf{P}_{L\mathbf{X}_k} = \mathbf{G}_{\mathbf{x}_k^r} \mathbf{P}_{\mathbf{x}_k \mathbf{X}_k} \tag{22}$$

Finally, we need to append these results to the state vector and the covariance matrix as follows:

$$\mathbf{X}_k = \begin{bmatrix} \mathbf{X}_k^T & \mathbf{L}_{N+1}^T \end{bmatrix}^T = \begin{bmatrix} \mathbf{X}_k^T & \mathbf{x}_{L_{N+1}} & \mathbf{y}_{L_{N+1}} \end{bmatrix}^T \tag{23}$$

$$\mathbf{P}_{X_k X_k} = \begin{bmatrix} \mathbf{P}_{X_k X_k} & \mathbf{P}_{L\mathbf{X}_k}^T \\ \mathbf{P}_{L\mathbf{X}_k} & \mathbf{P}_{LL} \end{bmatrix} \tag{24}$$

This concludes our discussion on the implementation of EKF-SLAM.

IV. Simulation and Results

For the purpose of simulation we use a very simple velocity model to capture the dynamics of a robot. The *motion model* of the robot is as given in equation 25.

$$\begin{bmatrix} \mathbf{x}_k \\ \mathbf{y}_k \\ \phi_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{k-1} \\ \mathbf{y}_{k-1} \\ \phi_{k-1} \end{bmatrix} + \begin{bmatrix} \cos\phi_{k-1} & 0 \\ \sin\phi_{k-1} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{dx}_k \\ \mathbf{d}\phi_k \end{bmatrix} \tag{25}$$

The *measurement model* is given in equation 8. One has to be mindful of the frame of reference in which each quantity is expressed.

In this simulation a number of analysis can be made, but due to paucity of time, we will just show the effect of the measurement/observation step in the EKF-SLAM algorithm. So, in the first run we show the results when no measurements are taken (i.e. no mapping) and the algorithm just uses the prediction step for state estimation and in the second run, measurements are taken(landmarks are mapped) in addition to the state propagation using the motion model.

The mean average errors when no measurements are made are 1.7689 units, 1.8073 units and 0.1457 rads along x, y and ϕ respectively.

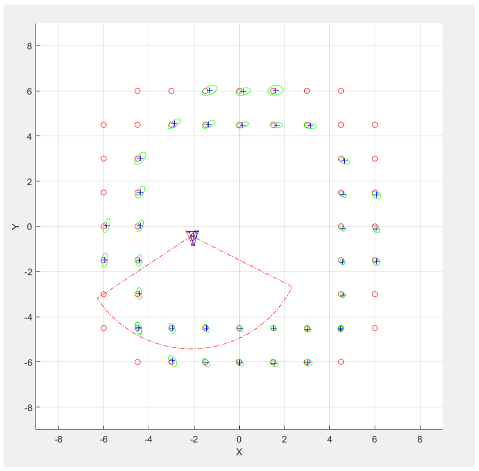

Fig. 3: EKF-SLAM Simulation environment in MATLAB: The red circles are the true position of landmarks and the small red triangle is the real robot, the blue triangle is the estimated robot and the small blue ellipse is the uncertainty associated with the robot position. The blue "+" signs are the estimated locations of the landmarks and the green ellipses around them denote the associated uncertainty. The ray like figure, emanating from the robot, with red dashed lines, is the range and bearing sensor

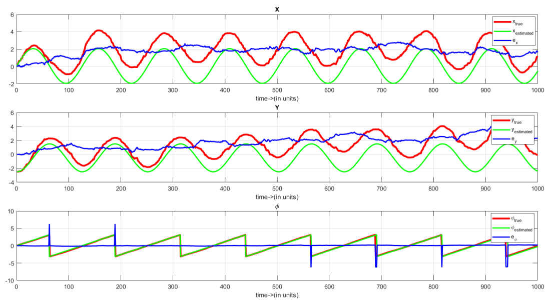

Fig. 4: Result of robot state estimation when no measurements were taken and the state estimate is equal to the predictions of motion model only. Red: Truth, Green: Estimate, Blue: Error.

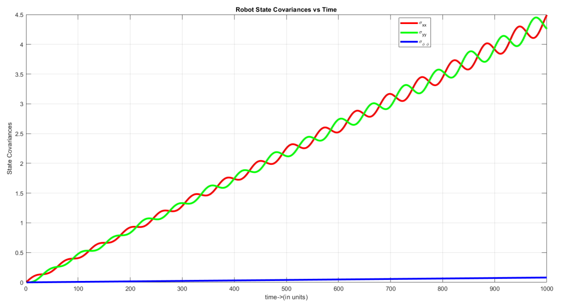

Fig. 5: Covariance associated with the Robot state when no measurements are made to support the motion model

.

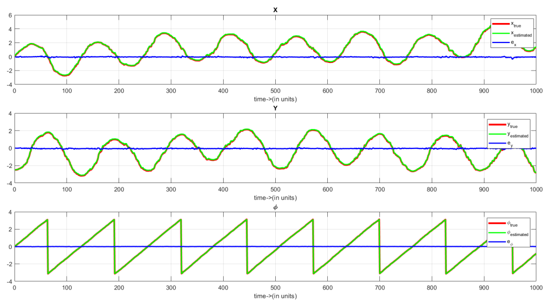

Fig. 6: Result of robot state estimation when measurements were taken and the state estimate is equal to the fusion of data both from the prediction and measurement models. Red: Truth, Green: Estimate, Blue: Error.

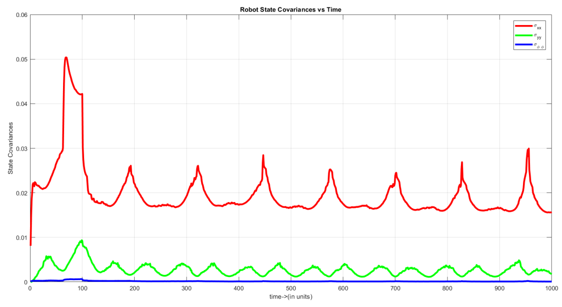

Fig. 7: Covariance associated with the Robot state when measurements are also made to support the motion model

.

The mean average errors when the measurement model was used to aid the prediction model are made are 0.0619 units, 0.0530 units and 0.0068 rads along x, y and ϕ respectively. Once can also note than in Figure 5 the covariance keeps on increasing with time but in Figure 7 the covariance is considerably lower and bounded.

V. Conclusion

In this report we have demonstrated the application of the Extended Kalman Filter to a very important estimation problem called Simultaneous Localization and Mapping. We saw how the Bayes law can be translated into standard analytical mathematical equations and how sensor measurements can help to aid state prediction. Although there are far more elegant and robust solutions to the SLAM problem, EKF still remains one of the most used paradigms of solving this important problem in the field of mobile robotics. In future we aim to use an Unscented Kalman Filter (UKF) in place of an EKF for solving the SLAM problem.

References

[1] H. Durrant Whyte and T. Bailey, "Simultaneous localization and mapping: part I" in IEEE Robotics & Automation Magazine, vol. 13, no. 2, pp. 99-110, June 2006. doi: 10.1109/MRA.2006.1638022

[2] J. Sola, "Simultaneous localization and mapping with the extended Kalman filter"