Comparison of Mobile Robot State Estimation using Extended Kalman Filter, Predictive Filter and Particle Filter

Cesar Corzo
George Mason University
Department of Electrical and Computer Engineering
Email: ccorzo@gmu.edu

***Abstract*—In this paper, three filtering techniques are presented for determining the position of a mobile robot. The aim is to find the best estimate of a state vector for the mobile robot system given only position measurements. Specifically, this paper presents three simulated results for the application of these filters to the non-linear dynamics for maneuvering vehicles.**

I. INTRODUCTION

In todays Robotics world, robot localization is a common problem in mobile robotics and it has become a prerequisite for modern autonomous navigation. Robots have moved from manufacturing to missions to another planets and/or moons. The level of autonomy depends on the ability of the robot to know its position using previous sensor data. Localization is a process in which it estimates the robot position from sensor or environment information. This paper attempts to solve the problem using only sensor information for accurate robot localization.

Filtering is one of the most widely used tool in engineering and embedded systems. Many papers has been written about Kalman Filter (KF) and its usefulness in a variety of applications. The KF is optimal for linear systems only. The Filtering problem for mobile robots is considerably more difficult since its kinematic model is non-linear. The extended kalman filter (EKF) was born to accommodate the nature of non-linear systems.

EKF works pretty well in practice and has been utilized for mobile robot navigation. This technique has some well-known weaknesses: (i) The derivation of the Jacobian matrices can be nontrivial and lead to implementation difficulties (ii) As non-linearity increases, estimate errors increases (iii) It assumes white Gaussian distribution for process disturbance and measurement noise [1].

A new approach for solving the filtering problem has been developed by Crassidis and Markley, and is called Predictive Filtering [2]. This algorithm is based on an algorithm called Minimum Model Error by Mook and Junkins. This filter can be implemented near real-time and is not constricted to model error Gaussian noise [3].

Another filter technique explored in this paper is the Particle filter (PF). This algorithm implements Bayesian equations and uses a posterior density estimation for the states. The PF uses a set of particles to represent the density distribution and it does not make any assumptions about the dynamics or the density function [4]. The beauty of this algorithm is that the state-space model can be non-linear and the initial state can take any form.

The organization of this paper is as follows. First, the basic mobile robot equations are presented along with its dynamics. Then each filter technique algorithms and equations are shown and devoted one chapter per filter. Finally, the comparison results between the filters are presented.

II. MOBILE ROBOT KINEMATIC MODEL

The type of Mobile Robot explored in this paper is the one with front wheel steering. The rear wheel axis is fixed and the front wheel provides steering through an actuator.

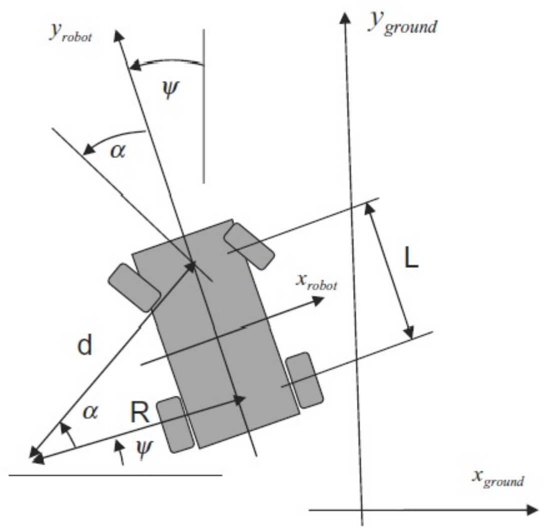

Fig. 1: Diagram of the Front-Wheel Mobile Robot [5]

In the diagram (Fig. 1), the steering angle α is measured in the counter clockwise direction. The heading angle ψ is the angle between the longitudinal axis of the robot, y_r, with y_g axis [5]. Then, the robot's kinematic equations in robot coordinate is as follow.

$$v_x = 0 \tag{1}$$

$$v_y = v_{rear\ wheel} \tag{2}$$

$$\dot{\psi} = \frac{v_{rear\ wheel}}{L} \tan \alpha \tag{3}$$

Simplifying robot speed $v_{rear\ wheel}$ into v and transforming these equations into ground coordinates [5], these become.

$$\dot{x} = -v \sin \psi \tag{4}$$
$$\dot{y} = v \cos \psi \tag{5}$$
$$\dot{\psi} = \frac{v}{L} \tan \alpha \tag{6}$$

The dynamic equations of the robot are simple but are nonlinear. Notice that α and v are control inputs. It is also convenient to represent these equations in discrete form.

$$x_k = x_{k-1} - \delta t \cdot v \sin \psi_{k-1} \tag{7}$$
$$y_k = y_{k-1} + \delta t \cdot v \cos \psi_{k-1} \tag{8}$$
$$\psi_k = \psi_{k-1} + \delta t \cdot \frac{v}{L} \tan \alpha \tag{9}$$

Where (δt) is the sample time.

A. Steering Control

There are several steering controllers for the mobile robot. In this paper, ψ_{des} is used as a desired heading and it represents the desired direction from the current position to the destination point (x_{des}, y_{des}). The formula is expressed as follows.

$$\psi_{des} = -\tan \left(\frac{x_{des} - x}{y_{des} - y} \right) \tag{10}$$

This document assumes a desired heading has been chosen, and the steering algorithm used in the simulations are presented in Cook [5]. The steering control logic used in this paper is as follows.

```
if $K\,|\psi_{des} - \psi| > \frac{\pi}{4}$ then
    $\alpha = \frac{\pi}{4} sign(\psi_{des} - \psi)$
else
    $\alpha = K(\psi_{des} - \psi)$
end if
```

This algorithm drives the robot in the proper direction and the heading error diminishes as a ramp if the heading error is greater than $\frac{\pi}{4}$ radians. It uses linear control if the error in heading is less than $\frac{\pi}{4}$ radians with the heading error decaying exponentially.

B. System Model

In controls and robotics, the state space representation is a mathematical model of a physical entity. It includes inputs(sensor data or controls), outputs and state variables. The state variables are expressed as a vector of size n. This paper is interested in tracking robot's position in a plane along with its heading. In this document, these states ($n = 3$) will be represented by the vector x. The output vector y represents the measurement model of the system. The simulations will assume the system has only position readings ($m = 2$) from sensors . The state representation of the robot is as follow.

$$\dot{x}(t) = f(x(t), u(t), w(t), t) \tag{11}$$
$$y(t) = h(x(t), t) + v(t) \tag{12}$$

The function f is nonlinear dependent on the state and input. $w(t)$ and $v(t)$ are the process and measurement disturbances, respectively. The simulation will assume constant velocity v and the steering control input α will be computed from equation (10) and the steering control logic presented before.

III. Extended Kalman Filter

The Extended Kalman Filter is a suboptimal algorithm based on Bayesian estimation. It works by making an assumption that the prior density function has Gaussian form. Let's start by describing the nonlinear system in discrete time from equation (11).

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}) \tag{13}$$

where x_k is the state vector at sample time k. The observable equation can be represented in discrete time as follows.

$$\tilde{y}_k = h(x_k) + v_k \tag{14}$$

Since the robot dynamics are non-linear, the EKF makes local linearization of the function by computing its Jacobi matrix [6]. Let F_k and H_k be the local linearization around x of f and h respectively, and G_k the one of f with respect to α.

$$F_k = \frac{\partial f(\cdot)}{\partial x} \tag{15}$$
$$G_k = \frac{\partial f(\cdot)}{\partial \alpha} \tag{16}$$
$$H_k = \frac{\partial h(\cdot)}{\partial x} \tag{17}$$

The nonlinear function f is represented by a vector of equations (7), (8), and (9), F_k then is as follows.

$$F_k = \begin{bmatrix} 1 & 0 & -\delta t \cdot v \cos \psi_{k-1} \\ 0 & 1 & -\delta t \cdot v \sin \psi_{k-1} \\ 0 & 0 & 1 \end{bmatrix} \tag{18}$$

$$G_k = \begin{bmatrix} 0 \\ 0 \\ \dfrac{\delta t \cdot v}{L} \left(\dfrac{1}{\cos^2 \alpha} \right) \end{bmatrix} \tag{19}$$

Since this paper assumes we have direct measurements of positions, the function $h(t)$ described in equation (12) is linear and can be represented by a scalar matrix H_k [5]. Thus, equation (17) does not necessarily need to be computed.

$$H_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{20}$$

The Kalman estimation equations and steps are described in table I. These equations use the linearized matrix versions described above.

IV. Predictive Filter

In this section, a short introduction to Predictive Filter is shown along with the state and output estimate equations [2].

$$\dot{\hat{x}}(t) = f(\hat{x}(t), u(t), t) + G(t)d(t) \tag{21}$$
$$\hat{y}(t) = H(\hat{x}(t), t) \tag{22}$$

Where f is a model vector of size n, $\hat{x}(t), t)$ is a n-size state estimate vector, $d(t)$ is a model error vector of size q, $G(t)$ is a n by q model-error distribution matrix which can

State Estimate Propagation	$\hat{x}_k^{pred} = f(\hat{x}_k, u_{k-1}, w_{k-1})$
Error Covariance Propagation	$P_k(-) = F_k P_k(+) F_k^T + G_k Q_k G_k^T$
Error Covariance Update	$P_{k+1}(+) = [I - K_k H_K] P_k(-) [I - K_k H]^T + K_k R K_k^T$
Kalman Gain Matrix	$K_k = P_k(-) H_k^T [H_k P_k(-) H_k^T + R]^{-1}$
State Estimate Update	$\hat{x}_{k+1} = \hat{x}_k^{pred} + K_k(\tilde{y_k} - H_k \hat{x}_k)$

TABLE I: EKF Algorithm [7]

include external disturbances, $H(t)$ is a m by n measurement matrix, and $\hat{y}$ is a estimated output vector of size m. State-observable measurements in discrete form have the following representation.

$$\tilde{y}(t_k) = H(\hat{x_t}(t_k), t_k) + v(t_k) \tag{23}$$

Where $\tilde{y}(t)$ is a mx1 measurement vector, $x(t)$ is the true state vector, and $v(t)$ is a mx1 measurement noise vector which is assumed to be a zero-mean, Gaussian white-noise distributed process.

A cost function consisting on measurement minus estimate residual plus the model correction vector is minimized and given by

$$\begin{aligned} J(t) = & \frac{1}{2}\left[\tilde{y}(t+\delta t) - \hat{y}(t+\delta t)\right]^T R^{-1} \\ & \left[\tilde{y}(t+\delta t) - \hat{y}(t+\delta t)\right] + \frac{1}{2} d(t) W d(t) \end{aligned} \tag{24}$$

where δt is the sampling interval and W is a 2x2 positive-definite matrix. This leads to the next expression for the model error vector equation.

$$\begin{aligned} d(t) = & -\left\{\left[\Lambda(\delta t) S(\hat{x})\right]^T R^{-1} \Lambda(\delta t) S(\hat{x}) + W\right\}^{-1} \\ & \left[\Lambda(\delta t) S(\hat{x})\right]^T R^{-1} \left[z(\hat{x}, \delta t) - \tilde{y}(t+\delta t) + \hat{y}(t+\delta t)\right] \end{aligned} \tag{25}$$

where the i^{th} element of $z[\hat{x}(t), \delta t]$ is given by

$$z_i[\hat{x}(t), \delta t] = \sum_{j=1}^{p_i} \frac{\delta t^j}{j!} L_f^j(H_i) \tag{26}$$

where p_i is the $H_i[\hat{x}(t)]$ lowest order derivative in which any term of $d(t)$ first appears due to successive differentiation and substitution for $\hat{x_i}(t)$ on the right side [2].

$L_f^j(H_i)$ is the j^{th}-order Lie derivative defined by

$$L_f^j(H_i) = \begin{cases} H_i, & \text{for } j = 0 \\ \dfrac{\partial L_f^{j-1}(H_i)}{\partial \hat{x}} f, & \text{for } j > 0 \end{cases} \tag{27}$$

$\Lambda(\delta t)$ is a diagonal matrix with elements given by

$$\lambda_{ii} = \frac{\delta t^{p_i}}{p_i!} \tag{28}$$

$S(\hat{x})$ is the sensitive matrix for nonlinear system and each i^{th} row is defined by

$$s_i(\hat{x}) = \left\{L_{g_1}\left[L_f^j(H_i)\right], ..., L_{g_q}\left[L_f^j(H_i)\right]\right\} \tag{29}$$

where the Lie derivative of L_{g_j} in equation (29) is

$$L_{g_j}[L_f^{p_i-1}(H_i)] = \frac{\partial L_f^{p_i-1}(H_i)}{\partial \hat{x}} g_j, \ j = 1, 2, ...q \tag{30}$$

Equation (25) is used to propagate the state estimates to time t_{k+1}. The W matrix serves as a tuning parameter. When W decreases, the estimates more closely follow the measurements since more model error is added. If W increases, the estimates more closely follow the propagated model [2].

V. Particle Filter

The goal of a particle filter is to keep track of the states of a given dynamic system using the sensor data (observation variables). The main advantages of PF are that it can handle non-Gaussian probability density function (pdf) and does not perform linearization. Consider the dynamic equation,

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}) \tag{31}$$

where x_k is the state vector at sample time k. The goal is to track the states from measurements

$$\tilde{y}_k = H_k(x_k, v_k) \tag{32}$$

This technique calculate the estimates of the state vector x at time k, given the sensor data $\tilde{y}_{1:k}$. This leads to the calculation of the pdf $p(x_k|\tilde{y}_{1:k})$ which can be computed recursively in two steps. In the prediction step, $p(x_k|\tilde{y}_{1:k-1})$ is calculated using equation (31) to obtain $p(x_k|x_{k-1})$ and $p(x_{k-1}|\tilde{y}_{1:k-1})$ is given from the previous calculation of the pdf.

$$p(x_k|\tilde{y}_{1:k-1}) = \int p(x_k|x_{k-1}) p(x_{k-1}|\tilde{y}_{1:k-1}) \ dx_{k-1} \tag{33}$$

The update state involves taking the prior distribution $p(x_k|\tilde{y}_{1:k-1})$ and updated with the new $\tilde{y}_k$ measurement.

$$p(x_k|\tilde{y}_{1:k}) = \frac{p(\tilde{y}_k|x_k) p(x_k|\tilde{y}_{1:k-1})}{p(\tilde{y}_k|\tilde{y}_{1:k-1})} \tag{34}$$

These equations (33, 34) represent the optimal Bayesian solution. This is a conceptual solution and generally cannot be carried out analytically. Thus, the need for approximate methods such as Monte Carlo sampling [1].

A. *Sequential Importance Sampling*

This method is the most fundamental Monte Carlo sampling technique and form the basis for others. The main idea in SIS is to approximate the posterior distribution with a weighted set of samples called *particles*. $\left\{x_{0:k}^i, w_k^i\right\}_{i=1}^M$ represents a random measure where $\left\{x_{0:k}^i, i = 0, ..., M\right\}$ are the support particles with associated weights $\{w_k^i, i = 1, ..., M\}$ and $x_{0:k}$ is the set of all states up to time k. Also, each particle $x^i$is a sample of a proposed $q(\cdot)$ called *Importance density* [1]. This density is assumed as.

$$q(x_{0:k}|\tilde{y}_{1:k}) = q(x_k|x_{0:k-1}, \tilde{y}_{1:k})q(x_{0:k-1}|\tilde{y}_{1:k-1}) \tag{35}$$

The samples, $x_{0:k}^i$, are chosen from the importance density, $q(x_{0:k}|\tilde{y}_{1:k})$, then the weights, w_k^i, can be approximated by the following.

$$w_k^i \propto \frac{p(x_{0:k}^i|\tilde{y}_{1:k})}{q(x_{0:k}^i|\tilde{y}_{1:k})} \tag{36}$$

The derivation given in [], reduces equation (36) into the next approximation.

$$w_k^i \propto w_{k-1}^i \frac{p(\tilde{y}_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i, \tilde{y}_k)} \tag{37}$$

Then SIS yields.

$$p(x_k|\tilde{y}_{1:k}) \approx \sum_{i=1}^{M} w_k^i \delta(x_{0:k} - x_{0:k}^i) \tag{38}$$

The SIS method suffers from degeneracy which occurs when all but one particle has negligible weights. One of the approach to solve this weakness is to increment the number of particles. This will take a lot of computational power and can be impractical. To overcome this deficit, particle filters rely on a good choice of Importance density and resampling [1].

B. Choice of Importance Density

There are many methods which can be used to create importance density. There are few cases where an optimal importance density can be analytically created. For many system this is impractical, and thus rely in suboptimal importance density. For simplicity, many authors choose the prior as an importance density [1].

$$q(x_k|x_{k-1}^i, \tilde{y}_k) = p(x_k|x_{k-1}^i) \tag{39}$$

Replacing equation (39) into (37) yields the following.

$$w_k^i \propto w_{k-1}^i p(\tilde{y}_k|x_k^i) \tag{40}$$

This is an approximation widely used since it is simple to program and intuitive. This is only an example of many other suboptimal densities. This choice of importance density will be used in the simulation of the mobile robot.

C. Resampling

Another way to deal with degeneracy is the resampling method. The main goal of resampling is to decimate particles with low weight and concentrate on particles with larger weights. It involves generating a new set of particles by resampling with replacement. In this paper, systematic resampling method is used due to its simplicity and computing time is linear. The algorithm is described below.

Algorithm 1 Resampling

```
Require: A set of particles $x_k^i$ and normalized weights $w_k^i$.
  $CDF[1] = 0$                          ▷ Initialize CDF
  for $i = 2 \rightarrow M$ do          ▷ Build the CDF
    $CDF[i] = CDF[i-1] + w_k^i$
  end for
  for $i = 1 \rightarrow M$ do
    $u = rand$
    $j = 0$
    while $u > CDF[j]$ do
      $j = j + 1$
    end while
    ${x_k^i}^* = x_k^j$                 ▷ Assign to the new set of particles
    $w_k^i = 1/M$                       ▷ All weights are reset
  end for
```

D. Sampling Importance Resampling

The SIR algorithm can be thought of a modified SIS algorithm with a good choice of importance density function (V-B) and resampling (V-C) at every time interval. Since resampling is applied at each time step, all particles weights w_k^i have the same value, $1/M$. Thus, equation (40) becomes

$$w_k^i \propto p(\tilde{y}_k|x_k^i). \tag{41}$$

The weights are directly proportional to the density function and they are normalized before going to the resampling stage [4]. Evaluating the weights is straight forward and the importance density easily sampled. This is the algorithm of choice for this paper due to its simplicity and good performance. The final estimate of the robot position is the mean of the particle states. The estimate can be calculated using other kind of metric from the re-sampled particles such as median, variance, etc.

Algorithm 2 SIR

```
Require: The prior set of particles $x_{k-1}^i$ and weights $w_{k-1}^i$,
  and $\tilde{y}_k$.
  ▷ Propagate the particles and calculate the weights.
  for $i = 1 \rightarrow M$ do
    $x_k^i = f(x_{k-1}^i, u_{k-1})$
    $w_k^i \sim p(\tilde{y}_k|x_k^i)$
  end for
  ▷ Normalize the weights.
  for $i = 1 \rightarrow M$ do
    $w_k^i = \frac{w_k^i}{\sum_{i=1}^{M} w_k^i}$
  end for
  ▷ Resample using algorithm 1.
```

VI. Results

In this section of the paper, I ran the Particle filter and Predictive filter against the Extended Kalman Filter and compare the results. The simulation was run with measurements covariance of 0.004 for both x and y positions and a process disturbance covariance of 0.0025.

A. Predictive Filter vs Extended Kalman Filter

The EKF and the Predictive filter were run side by side and the resulting robot's trajectory is shown in Figure 2 and the robot heading is shown in Figure 3. In the presence of high nonlinearity trajectories, the EKF performs a little better than the Predictive Filter. The predictive filter performs pretty close than the EKF and the difference is not noticeable despite having less information of the system. The mean square error average are shown in Table II.

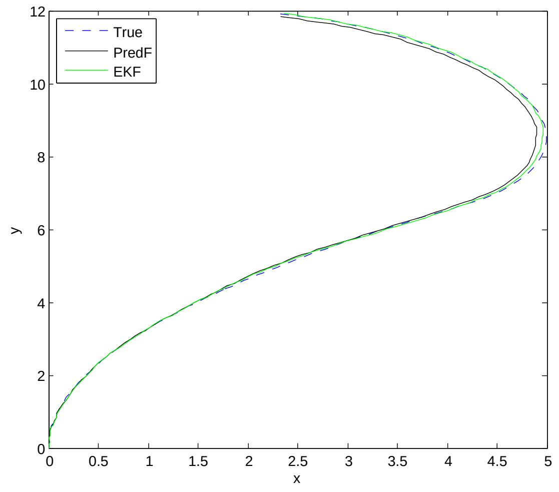

Fig. 2: Robot Trajectory

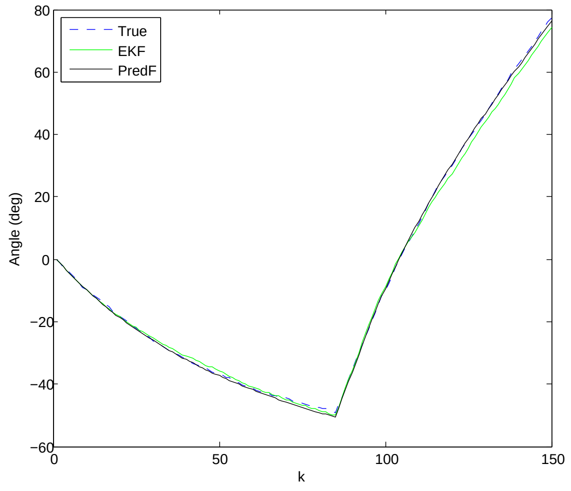

Fig. 3: Robot Heading

Figure 6 shows the absolute value of the estimate minus truth error for one simulation run.

B. Particle Filter vs Extended Kalman Filter

Similarly, the simulation was run side by side between these two filters and the results are shown in Figure 4 and Figure 5.

	EKF	Predictive Filter
$MSE(x)$[m]	0.0037	0.0031
$MSE(y)$[m]	0.0058	0.0062
$MSE(\Psi)$[rad]	0.0006	0.0007

TABLE II: Average MSE on 1000 simulation runs

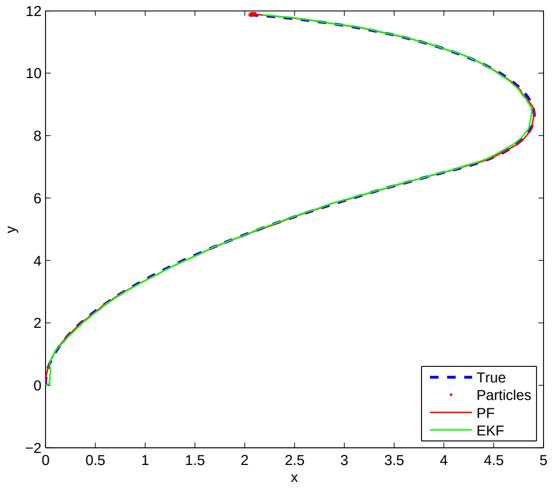

Fig. 4: Robot Trajectory

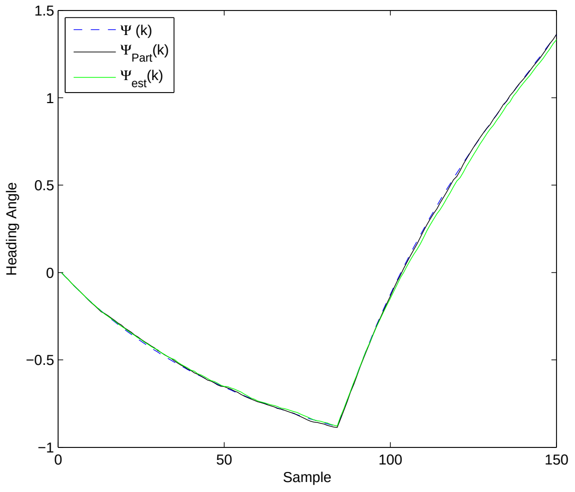

Fig. 5: Robot Heading

The particle Filter was run using 500 particles and have better performance than the EKF. On the bad side, it is computationally expensive since each particle is propagated and its weight computed. In addition, the resampling step adds more time delay. As the number of partciles increases so is the computation time. The mean square error average are shown in Table III.

Figure 7 shows the absolute value of the estimate minus

	EKF	Particle Filter
$MSE(x)$[m]	0.0035	0.0001507
$MSE(y)$[m]	0.0057	0.0000715
$MSE(\Psi)$[rad]	0.0004	0.0000381

TABLE III: Average MSE on 1000 simulation runs

truth error for one simulation run.

VII. Conclusion

In this paper, the Extended Kalman Filter was ran against the Predictive Filter and the Particle Filter. The results show that the particle filter outperforms the EKF but with a huge computational cost. The predictive Filter performs similar to the EKF with similar running time.

References

[1] M. Sanjeev Arulampalam, Simon Maskell, and Neil Gordon. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2002.

[2] John L. Crassidis and F. Landis Markley. Predictive filtering for nonlinear systems. *Journal of Guidance, Control, and Dynamics*, 20:566–572, 1997.

[3] D J Mook and J L Junkins. Minimum model error estimation for poorly modeled dynamic systems. *Journal of Guidance, Control, and Dynamics*, 11:256–261, 1988.

[4] Ioanis M. Rekleitis. A particle filter tutorial for mobile robot localization tr-cim-04-02. 2002.

[5] Gerald Cook. *Mobile Robots: Navigation, Control and Remote Sensing*. Wiley-IEEE Press, June 2011. ISBN 9780470630211.

[6] Mohinder S. Grewal and Angus P. Andrews. *Kalman Filtering Theory and Practice Using Matlab*. 3rd edition, September 2008. ISBN 9780470173664.

[7] Arthur Gelb, Joseph F. Kasper, Raymond A. Nash, Charles F. Price, and Arthur A. Sutherland. *Applied Optimal Estimation*. MIT Press, Cambridge, MA, 1974. ISBN 9780262570480.

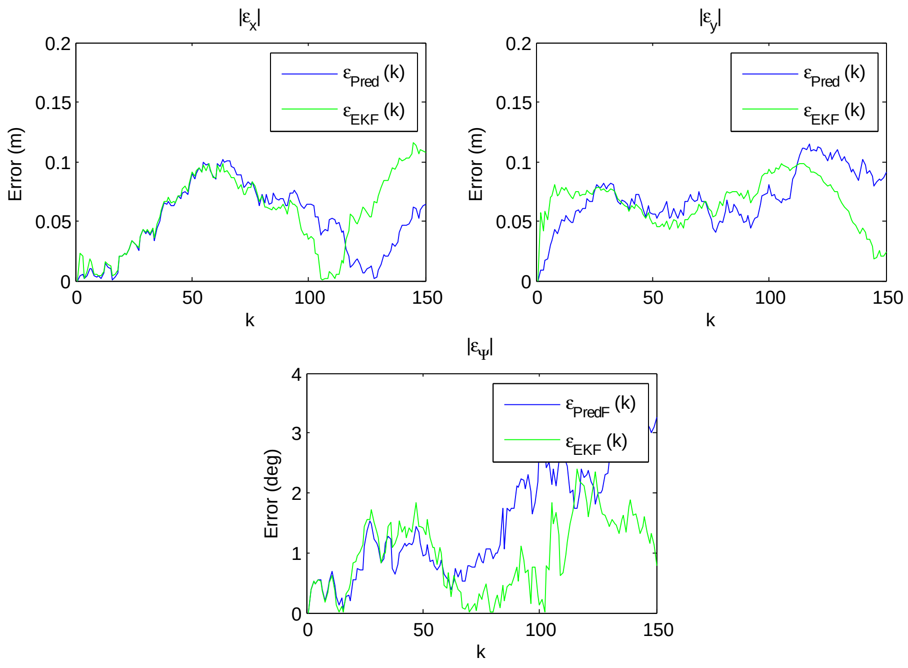

Fig. 6: EKF vs Predictive F. Error

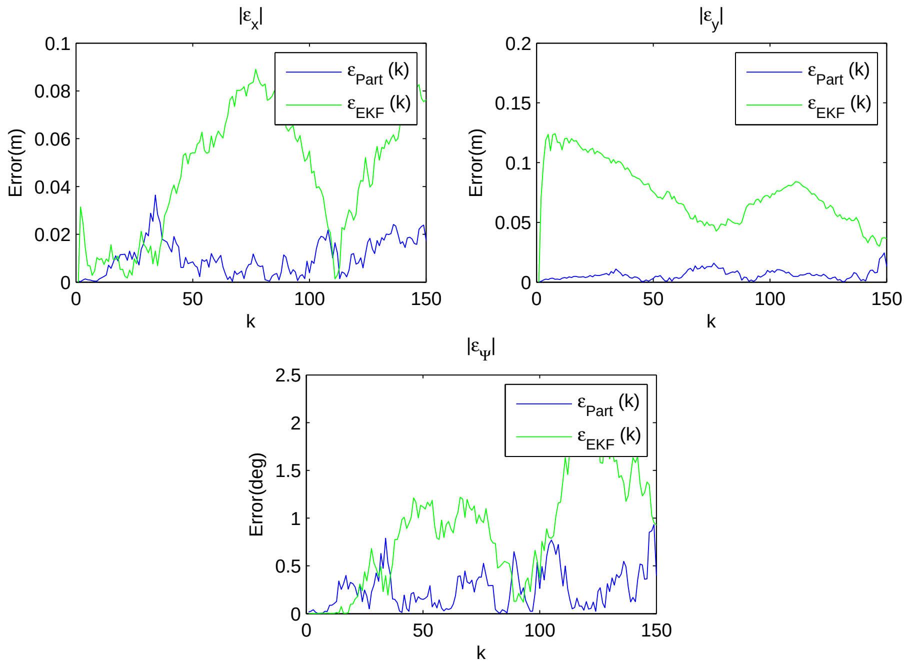

Fig. 7: EKF vs Particle F. Error