Consistency Comparisons of EKF-SLAM with Investigations into Robust EKF-SLAM

Chi Hay Tong
Division of Engineering Science
Faculty of Applied Science and Engineering
University of Toronto

Supervisor: Tim Barfoot
April 2008

Abstract

The Robust Extended Kalman Filter (REKF), presented recently in [7], is investigated as a possible improvement over the conventional Extended Kalman Filter (EKF) approach to solve the Simultaneous Localization and Mapping (SLAM) problem for mobile robotics. The limitations of the EKF are investigated compared to theoretical boundaries in the form of the Cramer-Rao Lower Bound (CRLB), and concise derivations for the EKF and REKF are presented. Simulations are conducted to obtain additional insight in the performance and failure points of the EKF-SLAM algorithm, alongside some REKF-SLAM trials.

Acknowledgements

I would like to thank Dr. Tim Barfoot for his guidance and support this year, and for giving me the opportunity to find a starting point in the area of field robotics. I have definitely been pushed hard, and have learned a great deal about this area, and more importantly, about myself and my abilities over the course of this project.

Additionally, I would like to thank my friends and family for the support and encouragement they have given me over the course of my undergraduate career.

CHI HAY TONG
April 11, 2008

Contents

List of Symbols

x_k	system state
z_k	measurement model
w_k	process noise
n_k	measurement noise
Q_k	process noise covariance / process error weight matrix
R_k	measurement noise covariance / measurement error weight matrix
Δt	time step size
u_k	system input
v	velocity input
ω	angular velocity input
r_l	landmark distance measurement
ϕ_l	landmark angle measurement
J	cost function
μ_k^-	*a priori* mean estimate
μ_k	*a posteriori* mean estimate
Σ_k^-	*a priori* covariance estimate
Σ_k	*a posteriori* covariance estimate
$\hat{z}_k$	estimated measurement
K_k	Kalman gain
F_k	Jacobian of f with respect to x
G_k	Jacobian of f with respect to u
H_k	Jacobian of h with respect to x
e_k	system error dynamics
S_k	robot pose error weight matrix
γ^2	upper bound on worst-case error
$\hat{x}_k^-$	*a priori* state estimate
$\hat{x}_k$	*a posteriori* state estimate
P_k^-	*a priori* P matrix (algorithm construct)
P_k	*a posteriori* P matrix

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

While robotics found its first widely-used application in the industrial setting, researchers worldwide have been working hard to change this concept, making robots more viable for other applications. There are promises of grand changes to society, but there are still challenges for robotics researchers before systems like entirely autonomous space rovers can be sent off for planetary exploration. With the desire that robots operate in the physical world comes inherently unpredictability, hardware limitations, and modeling approximations. A robust method to represent and reason with uncertainty will address these issues, and thus facilitate true autonomous navigation.

Simultaneous localization and mapping (SLAM) is a process by which a mobile robot can build a map and localize itself within the map at the same time. As a theoretical problem, the SLAM problem is viewed solved, but substantial issues with physical implementation currently prevent the development of truly general algorithms for scaling to large, real-world applications.

1.2 Literature Review

The initial research into this problem began with the seminal paper by Smith, Self and Cheeseman ([4]) which introduced the Stochastic Map. The Stochastic Map allows for concurrent mapping of the robot and the landmarks by augmenting the robot state vector with the landmark states, and using the Extended Kalman Filter as a tool for estimation. However, this traditional implementation of SLAM on real systems requires a number of assumptions and approximations that lead to inconsistencies in the algorithm. It has been shown by a number of papers ([2], [1]) that severe estimate degradation occurs when the Gaussian noise and linearization approximations do not hold. Two key books on the subject of statistical methods applied to mobile robotics and state estimation have recently been published, [6] and [3], of which the development of this implementation of the SLAM algorithm and the REKF derivation were influenced, respectively.

Various other filters and methods have been proposed to address this issue, including particle filters and variations on the EKF. A recent development by West and Syrmos ([7]) proposes the Robust Extended Kalman Filter (REKF) applied to SLAM, which addresses the limitations through the implementation of the H_∞ bounded norm.

1.3 Objectives

The primary purpose of this thesis is to investigate the improvements the REKF provides over the conventional EKF approach. Various aspects will be investigated, including comparisons to the theoretical limit of the Cramer-Rao Lower Bound (CRLB) (as introduced in [5]), as well as performance under Gaussian and non-Gaussian noise situations. As the REKF is a newly-proposed technique, its performance is not yet well detailed in literature, and has a number of variations to be experimented with.

In the following chapters, the simulator model and dataset generator will be introduced, as well as the underlying mathematical theory for EKF-SLAM and REKF-SLAM, culminating in the experiment results. Concise derivations are intended to provide a clean starting point for further investigations into the REKF method, with discussion and suggestions for future work to follow.

Chapter 2

Simulator

2.1 Introduction

As the goal of this thesis is to investigate the improvements of the SLAM algorithms, various other aspects were chosen to be as simple as possible, yet also general enough for the problem. The intent of the simplicity is to isolate the effects of the filtering algorithms, and avoid any possible unrelated complications. The first is the choice of the robot motion and sensor model, which is used to generate the datasets for testing. Figure 2.1 shows the model used.

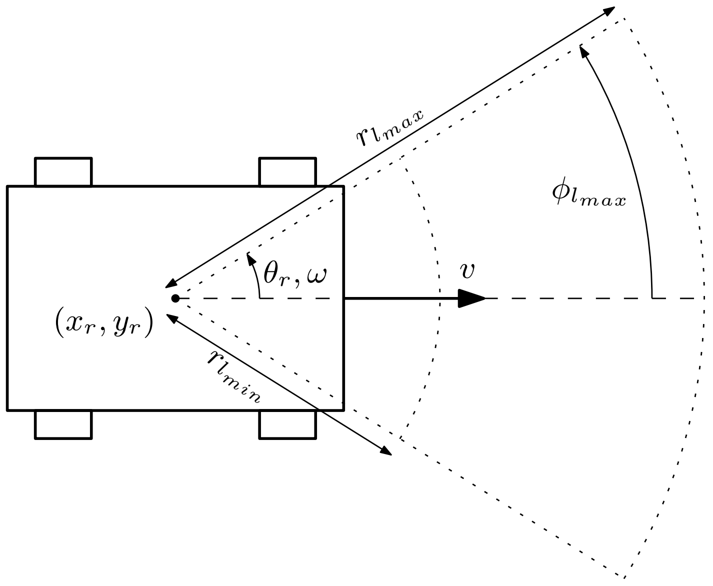

Figure 2.1: Robot motion and sensor model.

2.2 Motion Model

The motion model used is that of the 2D unicycle. This is a barebones representation of a mobile rover, with linear and angular velocities as the inputs. The robot pose is represented in two dimensions, with $x = (x_r, y_r, \theta_r)$ as the state, and the input is $u = (v, \omega)$. The equations governing motion of the rover are

$$\dot{x} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \tag{2.1}$$

To discretize this model, we will use the following numerical approximation:

$$\dot{x}(t) = \frac{x(t) - x(t-\Delta t)}{\Delta t} \tag{2.2}$$

With this approximation, our discrete state transition model is then:

$$x_k = x_{k-1} + \Delta t \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \end{bmatrix} \tag{2.3}$$

2.3 Sensor Model

The sensor model used is that of a generic range sensor, with a sensing cone defined by the maximum range r_{lmax}, minimum range r_{lmin}, and angle ϕ_{lmax}. This generalizes common rangefinding sensors such as radar, LIDAR, and computer vision. A reading is thus obtained for each landmark observation, $z_l = (r_l, \phi_l)$. Absolute identification is assumed, to simplify the algorithm.

$$r_l = \sqrt{(x_l - x_r)^2 + (y_l - y_r)^2} \tag{2.4}$$

$$\phi_l = \tan^{-1}\left(\frac{y_l - y_r}{x_l - x_r}\right) - \theta_r \tag{2.5}$$

$$r_{l_{min}} \quad |r_l| \leq r_{l_{max}} \tag{2.6}$$

$$|\phi_l| \leq \phi_{l_{max}} \tag{2.7}$$

2.4 Noise Model

To simulate uncertainty in sensor readings and motion commands, noise is injected into the datasets prior to storage. Initial tests use and vary magnitudes of i.i.d. Gaussian noise, but later ones also try to see if the filters are robust to other types - as EKF is defined under Gaussian assumptions.

$$\tilde{u} = u + \delta u, \quad \delta u \sim N(0, Q_k) \tag{2.8}$$

$$\tilde{z}_l = z_l + \delta z_l, \quad \delta z_l \sim N(0, R_k) \tag{2.9}$$

2.5 Simulation

For the datasets, v and ω are chosen to be constant, so that the robot travels in a circle. This makes it easy to see the ground truth deviation of the estimates. As well, the landmarks were randomly generated in a annular distribution around the path of the robot, and the noisy control and noisy sensor readings were stored to file as input for the filtering algorithm.

2.6 Implementation

For implementation, a single file is used to define the simulation initializations and input commands, and a random map is generated from it. These parameters are fed into the simulator, generating a noisy dataset storing the ground truth, noisy input, noisy sensor readings, map, and various model properties. These can be used to generate an animation for viewing, as well as fed to the SLAM algorithms. Consistency plots can then be generated from the estimation results. Figure 2.2 below summarizes the code file relations.

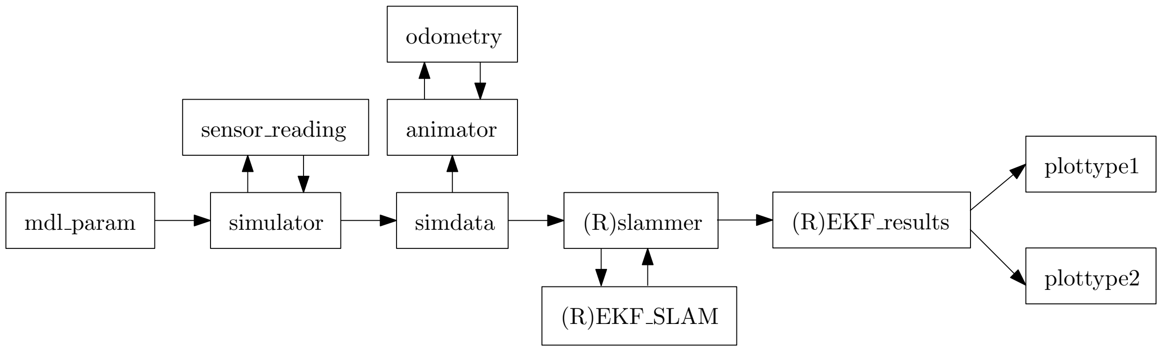

Figure 2.2: File relation flowchart.

Chapter 3

Kalman Filter (KF)

3.1 Introduction

The Kalman Filter (KF) is a well-studied recursive state estimator for stochastic systems. Based on Bayes' Rule, and a number of assumptions, the Kalman Filter estimates the state of a system by incorporating a series of noisy measurements. These assumptions include the Markov properties, linear state transitions and measurements, a normally distributed initial belief, and zero-mean, Gaussian noise characteristics. Alternately, one can also view the system as a Markov chain with linear operators perturbed by Gaussian noise. While the Kalman Filter is a discrete-time method, it is still highly applicable given the discrete approximations that we apply anyway when implementing algorithms on computers today. As a result of the derivations, the Kalman Filter is also a minimum mean-square error estimator (MMSE). The linear state transition and measurement models are given below, along with the Gaussian noise assumptions factored directly into the process and measurement equations.

$$x_k = A_k x_{k-1} + B_k u_k + w_k, \quad w_k \sim N(0, Q_k) \tag{3.1}$$

$$z_k = C_k x_k + n_k, \quad n_k \sim N(0, R_k) \tag{3.2}$$

3.2 Algorithm

The algorithm can be generally viewed in two steps, as a predictor-corrector. The Filter maintains the first and second moments of the state distribution - the mean and covariance, keeping with the Gaussian assumption. These are represented by μ_k^- and Σ_k^- regarded as the predictive step, and μ_k and Σ_k the corrective step. For the robotic rover, the motion model is applied first as a predictor, and predictions are made as to the location of the landmarks. These predictions are compared against the actual measurements, and the difference (known as the innovation) is used alongside the Kalman gain, K_k, as a correction to the state estimate. The following set of equations summarizes the overall algorithm at each update step.

$$\begin{aligned}
\mu_k^- &= A_k\mu_{k-1} + B_k u_k && (3.3)\\
\Sigma_k^- &= A_k\Sigma_{k-1}A_k^T + Q_k && (3.4)\\
K_k &= \Sigma_k^- C_k^T \left(C_k\Sigma_k^- C_k^T + R_k\right)^{-1} && (3.5)\\
\mu_k &= \mu_k^- + K_k\left(z_k - C_k\mu_k^-\right) && (3.6)\\
\Sigma_k &= (I - K_kC_k)\Sigma_k^- && (3.7)
\end{aligned}$$

3.3 Derivation

In this derivation, we will split the algorithm into its two component steps, the predictive and corrective elements. We will begin by assuming a form for the filter:

$$\begin{aligned}
\mu_k^- &:= A_k\mu_{k-1} + B_k u_k && (3.8)\\
\mu_k &:= \mu_k^- + K_k\left(z_k - \hat{z}_k\right) && (3.9)\\
\hat{z}_k &:= C_k\mu_k^- && (3.10)
\end{aligned}$$

The choice for μ_k^- can be motivated by seeing that with the linear operators of the state transition equation in Equation 3.1, our best initial estimate would be to use our best corrected estimate generated in the prior step. For the estimate μ_k, it is based on the previous estimate and the new measurement z_k. K_k is a gain matrix to be determined, and the quantity $(z_k - \hat{z}_k)$ is known as the innovation, a corrective term to the estimate.

Alongside these estimators, we can define the *a priori* and the *a posteriori* estimate-errors:

$$\begin{aligned}
e_k^- &:= x_k - \mu_k^- && (3.11)\\
e_k &:= x_k - \mu_k && (3.12)
\end{aligned}$$

Alongside these, we can define the respective *a priori* and *a posteriori* estimate-errors covariances:

$$\begin{aligned}
\Sigma_k^- &:= E\left[e_k^- e_k^{-T}\right] && (3.13)\\
\Sigma_k &:= E\left[e_k e_k^T\right] && (3.14)
\end{aligned}$$

With these definitions and the system dynamics, we can complete the expression for e_k^-:

$$\begin{aligned}
e_k^- &= x_k - \mu_k^-\\
&= A_k x_{k-1} + B_k u_k + w_k - A_k\mu_{k-1} - B_k u_k\\
&= A_k e_k + w_k && (3.15)
\end{aligned}$$

Next, for Σ_k^-:

$$\begin{aligned}
\Sigma_k^- &= E\left[e_k^- e_k^{-T}\right] \\
&= E\left[\left(A_k e_k + w_k\right)\left(A_k e_k + w_k\right)^T\right] \\
&= E\left[A_k e_k e_k^T A_k^T + w_k w_k^T\right] \\
&= A_k E\left[e_k e_k^T\right] A_k^T + E\left[w_k w_k^T\right] \\
&= A_k \Sigma_{k-1} A_k^T + Q_k
\end{aligned} \tag{3.16}$$

Similarly, for e_k:

$$\begin{aligned}
e_k &= x_k - \mu_k \\
&= x_k - \mu_k^- - K_k\left(z_k - C_k \mu_k^-\right) \\
&= e_k^- - K_k\left(C_k x_k + n_k - C_k \mu_k^-\right) \\
&= e_k^- - K_k\left(C_k e_k^- + n_k\right) \\
&= \left(I - K_k C_k\right) e_k^- - K_k n_k
\end{aligned} \tag{3.17}$$

And then for Σ_k:

$$\begin{aligned}
\Sigma_k &= E\left[\left(\left(I - K_k C_k\right) e_k^- - K_k n_k\right)\left(\left(I - K_k C_k\right) e_k^- - K_k n_k\right)^T\right] \\
&= E\left[\left(I - K_k C_k\right) e_k^- e_k^{-T}\left(I - K_k C_k\right)^T - \left(I - K_k C_k\right) e_k^- n_k^T K_k^T - K_k n_k e_k^{-T}\left(I - K_k C_k\right) + K_k n_k n_k^T K_k^T\right] \\
&= \left(I - K_k C_k\right) E\left[e_k^- e_k^{-T}\right]\left(I - K_k C_k\right)^T - \left(I - K_k C_k\right) E\left[e_k^- n_k^T\right] K_k^T \\
&\quad - K_k E\left[n_k e_k^{-T}\right]\left(I - K_k C_k\right)^T + K_k E\left[n_k n_k^T\right] K_k^T \\
&= \left(I - K_k C_k\right) \Sigma_k^- \left(I - K_k C_k\right)^T + K_k R_k K_k^T
\end{aligned} \tag{3.18}$$

As can be seen, for the last line of simplification, $E\left[e_k^- n_k^T\right] = 0$ and $E\left[n_k e_k^{-T}\right] = 0$. That is, the *a posterior* estimate-error is assumed to be uncorrelated with n_k, the measurement noise of the current step (and unaffected by values in previous steps since n_k is i.i.d.).

Now, to find K_k, the optimal gain, we must define a cost function to minimize. We will choose to minimize the sum of the squares of the estimation-error of each component of x_k at every time step, resulting in the MMSE property of the Kalman Filter.

In this case, we can minimize the trace of Σ_k, using the matrix calculus identity $\frac{\partial \mathrm{tr}\left(ABA^T\right)}{\partial A} = 2AB$.

$$\frac{\partial \mathrm{tr}\Sigma_k}{\partial K_k} = 2\left(I - K_k C_k\right)\Sigma_{k-1}^-\left(-C_k^T\right) + 2K_k R_k \tag{3.19}$$

Setting the derivative to 0 and solving for K_k, we get:

$$\begin{aligned}
K_k R_k &= \left(I - K_k C_k\right) \Sigma_{k-1}^- C_k^T \\
K_k R_k &= \Sigma_{k-1}^- C_k^T - K_k C_k \Sigma_{k-1}^- C_k^T \\
K_k &= \Sigma_{k-1}^- C_k^T \left(C_k \Sigma_{k-1}^- C_k^T + R_k\right)^{-1} \qquad (3.20)
\end{aligned}$$

With this result, we have the entirety of the algorithm. However, we can take these results and further simplify the expressions for ease of implementation and computational efficiency by combining equations 3.18 and 3.20. Starting with Σ_k, and inserting the expression for K_k, we get:

$$\begin{aligned}
\Sigma_k &= \left(I - \Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} C_k\right) \Sigma_k^- \left(I - \Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1}\right)^T \\
&\quad + \Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} R_k \left(\Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1}\right)^T \qquad (3.21)
\end{aligned}$$

Expanding out the bracketed terms:

$$\begin{aligned}
\Sigma_k &= \Sigma_k^- - \Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} C_k \Sigma_k^- \\
&\quad - \Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} C_k \Sigma_k^- \\
&\quad + \Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} C_k \Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} C_k \Sigma_k^- \\
&\quad + \Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} R_k C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} C_k \Sigma_k^- \\
&= \Sigma_k^- - 2\Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} C_k \Sigma_k^- \\
&\quad + \Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} C_k \Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} C_k \Sigma_k^- \\
&\quad + \Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} R_k C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} C_k \Sigma_k^- \qquad (3.22)
\end{aligned}$$

Combining the last two terms:

$$\begin{aligned}
\Sigma_k &= \Sigma_k^- - 2\Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} C_k \Sigma_k^- \\
&\quad + \Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} \left(C_k \Sigma_k^- C_k^T + R_k\right) \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} C_k \Sigma_k^- \\
&= \Sigma_k^- - 2\Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} C_k \Sigma_k^- + \Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} C_k \Sigma_k^- \\
&= \Sigma_k^- - \Sigma_k^- C_k^T \left(C_k \Sigma_k^- C_k^T + R_k\right)^{-1} C_k \Sigma_k^- \qquad (3.23)
\end{aligned}$$

From here, we can reinsert the expression for K_k, and complete the derivation.

$$\begin{aligned}\Sigma_k &= \Sigma_k^- - K_k C_k \Sigma_k^- \\ &= (I - K_k C_k)\Sigma_k^- \qquad (3.24)\end{aligned}$$

Chapter 4

Extended Kalman Filter (EKF)

4.1 Introduction

One of the basic assumptions of the Kalman Filter is linearity. Unfortunately, in the real world, linear systems do not exist, even for our very simple robot model. One approach to address nonlinearity is to linearize the state transition and measurement models, resulting in the Extended Kalman Filter (EKF). For this, we consider the generalized form of our nonlinear system model, where the noise again is factored directly into the process and measurement equations.

$$x_k = f\left(x_{k-1}, u_k\right) + w_k, \;\; w_k \sim N(0, Q_k) \tag{4.1}$$
$$z_k = h\left(x_k\right) + n_k, \;\; n_k \sim N(0, R_k) \tag{4.2}$$

4.2 Algorithm

After the linearization steps are applied, the algorithm largely resembles that of the regular Kalman Filter.

$$\mu_k^- = f\left(\mu_{k-1}, u_k\right) \tag{4.3}$$
$$\Sigma_k^- = F_k \Sigma_{k-1} F_k^T + Q_k \tag{4.4}$$
$$K_k = \Sigma_k^- H_k^T \left(H_k \Sigma_k^- H_k^T + R_k\right)^{-1} \tag{4.5}$$
$$\mu_k = \mu_k^- + K_k\left(z_k - h\left(\mu_k^-\right)\right) \tag{4.6}$$
$$\Sigma_k = \left(I - K_k H_k\right)\Sigma_k^- \tag{4.7}$$

4.3 Derivation

The derivation for the EKF also resembles that of the KF, after the linearization step. As such, we first begin by linearizing the state transition model using a first order Taylor series expansion around μ_{k-1}.

$$\begin{aligned}
f\left(x_{k-1}, u_k\right) &= f\left(\mu_{k-1}, u_k\right) + \Delta x_k && (4.8)\\
\Delta x_k &= \left.\frac{\partial f}{\partial x}\right|_{\mu_{k-1}} \delta x_{k-1} + H.O.T. && (4.9)\\
\Delta x_k &\approx F_k \delta x_{k-1} && (4.10)
\end{aligned}$$

Where F_k is the Jacobian of f with respect to x. Then, as motivated by the KF derivation, we find the mean prediction, as the form is no longer as simple to assume. However, the result of taking the expected value of both sides of the state transition equation is quite intuitive:

$$\begin{aligned}
\mu_k^- &= E\left[x_k\right]\\
&= E\left[f\left(\mu_{k-1}, u_k\right) + \Delta x_k + w_k\right]\\
&= E\left[f\left(x_{k-1}, u_k\right)\right] + E\left[\Delta x_k\right] + E\left[w_k\right]\\
&= f\left(\mu_{k-1}, u_k\right) && (4.11)
\end{aligned}$$

Next, for the covariance prediction:

$$\begin{aligned}
\Sigma_k^- &= E\left[\left(x_k - \mu_k^-\right)\left(x_k - \mu_k^-\right)^T\right]\\
&= E\left[\left(f\left(\mu_{k-1}, u_k\right) + \Delta x_k + w_k - \mu_k^-\right)\left(\ldots\right)^T\right]\\
&= E\left[\left(\mu_k^- + \Delta x_k + w_k - \mu_k^-\right)\left(\ldots\right)^T\right]\\
&= E\left[\left(\Delta x_k + w_k\right)\left(\ldots\right)^T\right]\\
&= E\left[\left(F_k \delta x_{k-1} + w_k\right)\left(\ldots\right)^T\right]\\
&= F_k E\left[\delta x_{k-1} \delta x_{k-1}^T\right] F_k^T + E\left[w_k w_k^T\right]\\
&= F_k \Sigma_{k-1} F_k^T + Q_k && (4.12)
\end{aligned}$$

We now continue on to the correction step, with the Taylor series expansion of the measurement model around μ_k^-:

$$\begin{aligned}
h\left(x_k\right) &= h\left(\mu_k^-\right) + \Delta z_k && (4.13)\\
\Delta z_k &= \left.\frac{\partial f}{\partial x}\right|_{\mu_k^-} \delta x_k + H.O.T. && (4.14)\\
\Delta z_k &\approx H_k \delta x_k && (4.15)
\end{aligned}$$

Choosing an appropriate estimate, $\hat{z}_k$:

$$\begin{aligned}\hat{z}_k &= E\left[z_k\right] \\ &= E\left[h\left(\mu_k^-\right) + \Delta z_k + n_k\right] \\ &= E\left[h\left(\mu_k^-\right)\right] + E\left[\Delta z_k\right] + E\left[n_k\right] \\ &= h\left(\mu_k^-\right) \qquad (4.16)\end{aligned}$$

Then, defining the innovation term:

$$\begin{aligned}z_k - \hat{z}_k &= h\left(x_k\right) + n_k - h\left(\mu_k^-\right) \\ &= h\left(\mu_k^-\right) + \Delta z_k + n_k - h\left(\mu_k^-\right) \\ &= \Delta z_k + n_k \\ &\approx H_k \delta x_k + n_k \qquad (4.17)\end{aligned}$$

As this is of the same form as the measurement model in the KF derivation, the correction step derivation gives a very similar result.

$$K_k = \Sigma_k^- H_k^T \left(H_k \Sigma_k^- H_k^T + R_k\right)^{-1} \qquad (4.18)$$

$$\mu_k = \mu_k^- + K_k \left(z_k - h\left(\mu_k^-\right)\right) \qquad (4.19)$$

$$\Sigma_k = \left(I - K_k H_k\right) \Sigma_k^- \qquad (4.20)$$

Chapter 5

Robust Kalman Filter (RKF)

5.1 Introduction

While the Kalman Filter is presented as a mathematically elegant tool for state estimation, a number of issues arise in actual implementation on real-world systems. The two main issues are the fundamental assumptions of linearity and zero-mean, Gaussian noise, and the possibility of inaccuracies in system modeling. The linearization in the EKF fundamentally violates the assumptions, and as such, the algorithm fails to perform properly.

To address these issues, a new filter was developed for robustness to these modeling inaccuracies, addressing estimation errors bounded by the H_∞ norm. As a result, the filter limits the worst-case error, but allows the average overall error to be bigger. This filter is known as the H_∞ Filter, or, as it will soon become apparent, the Robust Kalman Filter (RKF).

To begin, we will retain the linear model for the state transition and measurement models, and introduce two additive parameters, w_k, and n_k.

$$x_{k+1} = A_k x_k + B_k u_k + w_k \tag{5.1}$$

$$z_k = C_k x_k + n_k \tag{5.2}$$

While these parameters were used to represent noise in the previous filters, these parameters will now also capture the unmodeled system dynamics, as well as additive noise. As such, no assumptions are made at all about the distribution of these parameters.

5.2 Cost Function

For this derivation of the H$_\infty$ Filter, we will take the game theory approach, as it provides some intuition as well as a clean derivation. For the estimates at each timestep from 0 to K-1, we define the cost function as:

$$\tilde{J} := \frac{\sum_{k=0}^{K-1} ||x_k - \hat{x}_k^-||^2_{S_k}}{||x_0 - \hat{x}_0^-||^2_{P_0^{-1}} + \sum_{k=0}^{K-1}\left(||w_k||^2_{Q_k^{-1}} + ||n_k||^2_{R_k^{-1}}\right)} \tag{5.3}$$

Defining S_k, P_0^-, Q_k, and R_k as weights (symmetric, positive-definite matrices), the cost function is defined as the weighted L_2-norms of the estimation error over the weighted L_2-norms of the noise. It will be seen later that the values of P_0^-, Q_k, and R_k can be taken as analogous to those in the Kalman Filter, representing initial uncertainty, and noise parameters, if they are known. However, since w_k and n_k are supposed to capture the unmodeled system dynamics along with the additive noise, the magnitudes of the weighting matrices may need to be increased. S_k does not have an analogy in the Kalman Filter, but it is a user-specified matrix with relative weighting to which components of the error in the robot pose estimate the user deems more important.

Viewing this as a game, our goal is to minimize the estimation error, while our adversary, Nature, assumed to be perverse, attempts to maximize the error. We define $\tilde{J}$ with the noise variables and initial conditions in the denominator to mitigate the use of of large magnitudes, necessitating clever choices to increase the estimation error instead of brute force. As such, we can view this rivalry as a minimax problem, wherein the optimal strategies are played at each timestep, k:

$$\min_{\hat{x}_k^-} \max_{w_k, n_k, x_0} \tilde{J} \tag{5.4}$$

It is analytically intractable to solve for the optimal estimator to minimize the cost function, so instead we can guarantee a bound, γ^2, to the function:

$$\tilde{J} < \gamma^2 \tag{5.5}$$

This user-specified upper bound to the cost function is the H$_\infty$ bound that we consider in the H$_\infty$ filter. As will be seen, this parameter will need to be determined experimentally, which may or may not be chosen to be its minimum value due to various performance specifications.

5.3 Algorithm

Again, the resulting algorithm is in the form of a predictor-corrector. In fact, it largely resembles the form of the Kalman Filter, but with robustness to noise and modeling uncertainties (explaining the name RKF):

$$\begin{aligned}
\hat{x}_{k+1}^{-} &= A_k\hat{x}_k + B_k u_k &(5.6)\\
P_k^{-} &= A_k P_k A_k^T + Q_k &(5.7)\\
\tilde{P}_k &= \left(I - \frac{1}{\gamma^2}S_k P_k^{-}\right)^{-1}\left(2I - \frac{1}{\gamma^2}S_k P_k^{-}\right)P_k^{-}\left(I - \frac{1}{\gamma^2}S_k P_k^{-}\right)^{-1} &(5.8)\\
K_k &= \tilde{P}_k C_k^T\left(C_k\tilde{P}_k C_k^T + R_k\right)^{-1} &(5.9)\\
\hat{x}_k &= \hat{x}_k^{-} + K_k\left(z_k - C_k\hat{x}_k^{-}\right) &(5.10)\\
P_k &= \left(I - K_k C_k\right)\tilde{P}_k &(5.11)
\end{aligned}$$

One of the particular differences is that we are no longer estimating μ_k and Σ_k, but now $\hat{x}_k$ and P_k, respectively. The reason for this is that without the Gaussian noise assumption, we can no longer maintain a Gaussian representation for our state estimates. $\hat{x}_k$ is our best estimate of the state, and P_k is a parameter defined for the sake of the calculation of K_k. In light of that, we can no longer view P_k as the certainty of the estimate, as it is just a value that evolves through time alongside the $\hat{x}_k$.

5.4 Derivation

To begin this derivation, we will first need to start with a couple of definitions. We will assume a form for the estimator, defined as:

$$\begin{aligned}
\hat{x}_{k+1}^{-} &:= A_k\hat{x}_k + B_k u_k &(5.12)\\
\hat{x}_k &:= \hat{x}_k^{-} + K_k\left(z_k - C_k\hat{x}_k^{-}\right) &(5.13)
\end{aligned}$$

In addition, we will consider the estimation-error dynamics, and how they propagate:

$$\begin{aligned}
e_k &:= x_k - \hat{x}_k^{-} &(5.14)\\
e_{k+1} &= x_{k+1} - \hat{x}_{k+1}^{-}\\
&= \left(A_k x_k + B_k u_k + w_k\right) - \left(A_k\hat{x}_k + B_k u_k\right)\\
&= A_k e_k + w_k - A_k K_k\left(C_k x_k + n_k - C_k\hat{x}_k^{-}\right)\\
&= A_k\left(I - K_k C_k\right)e_k + w_k - A_k K_k n_k &(5.15)
\end{aligned}$$

With these definitions, we can manipulate the cost function into a more useful form.

$$\tilde{J} = \frac{\sum_{k=0}^{K-1} e_k^T S_k e_k}{e_0^T {P_0^{-}}^{-1} e_0 + \sum_{k=0}^{K-1}\left(w_k^T Q_k^{-1} w_k + n_k^T R_k^{-1} n_k\right)} < \gamma^2 \quad (5.16)$$

Bringing the denominator to the other side of the inequality:

$$\sum_{k=0}^{K-1} e_k^T S_k e_k < \gamma^2 \left(e_0^T P_0^{-^{-1}} e_0 + \sum_{k=0}^{K-1} \left(w_k^T Q_k^{-1} w_k + n_k^T R_k^{-1} n_k \right) \right) \tag{5.17}$$

Dividing both sides by $2\gamma^2$, and moving the terms all to one side:

$$\sum_{k=0}^{K-1} \frac{1}{2\gamma^2} e_k^T S_k e_k - \frac{1}{2} e_0^T P_0^{-^{-1}} e_0 - \sum_{k=0}^{K-1} \left(\frac{1}{2} w_k^T Q_k^{-1} w_k + \frac{1}{2} n_k^T R_k^{-1} n_k \right) < 0 \tag{5.18}$$

For use in the rest of the derivation, we will define J as our cost function, which includes the Lagrange multiplier terms λ_k to maintain the error dynamic relations as constraints in the optimization.

$$\begin{aligned} J \quad := \quad & \sum_{k=0}^{K-1} \left[\frac{1}{2} e_k^T \left(\frac{1}{\gamma^2} S_k \right) e_k - \frac{1}{2} w_k^T Q_k^{-1} w_k - \frac{1}{2} n_k^T R_k^{-1} n_k \right. \\ & \left. + \lambda_{k+1}^T \left(A_k \left(I - K_k C_k \right) e_k + w_k - A_k K_k n_k - e_{k+1} \right) \right] - \frac{1}{2} e_0^T P_0^{-^{-1}} e_0 \end{aligned} \tag{5.19}$$

This cost function is guaranteed to be constrained by the bound, and we will attempt to find an analytical solution to the minimax problem using it.

For Nature's influence, we will maximize the cost function with respect to the two noise parameters, as well as the initial condition. The maximal parameters will be found by obtaining the partial derivatives of J with respect to the parameters, setting them to 0, and solving:

$$\frac{\partial J}{\partial w_k}^T = -Q_k^{-1} w_k + \lambda_{k+1} = 0 \Rightarrow w_k = Q_k \lambda_{k+1} \tag{5.20}$$

$$\frac{\partial J}{\partial n_k}^T = -R_k^{-1} n_k - K_k^T A_k^T \lambda_{k+1} = 0 \Rightarrow n_k = -R_k K_k^T A_k^T \lambda_{k+1} \tag{5.21}$$

$$\begin{aligned} \frac{\partial J}{\partial x_0}^T &= \frac{1}{\gamma^2} S_0 e_0 + \left(I - K_0 C_0 \right)^T A_0^T \lambda_1 - P_0^{-^{-1}} e_0 = 0 \\ & \Rightarrow e_0 = \left(P_0^{-^{-1}} - \frac{1}{\gamma^2} S_0 \right)^{-1} \left(I - K_0 C_0 \right)^T A_0^T \lambda_1 \end{aligned} \tag{5.22}$$

For cleanliness, let us define λ_0, so to simplify the expression for e_0:

$$\lambda_0 := \left(I - \frac{1}{\gamma^2} S_0 P_0^- \right)^{-1} \left(I - K_0 C_0 \right))^T A_0^T \lambda_1 \tag{5.23}$$

So that:

$$e_0 = P_0^- \lambda_0 \tag{5.24}$$

Plugging these expressions into J, and since Q_k, R_k are symmetric matrices, we get:

$$\begin{aligned} J &= \sum_{k=0}^{K-1} \Bigg[\frac{1}{2\gamma^2} e_k^T S_k e_k - \frac{1}{2}\lambda_{k+1}^T Q_k^T Q_k^{-1} Q_k \lambda_{k+1} - \frac{1}{2}\lambda_{k+1}^T A_k K_k R_k^T R_k^{-1} R_k K_k^T A_k^T \lambda_{k+1} \\ &\quad + \lambda_{k+1}^T \left(A_k \left(I - K_k C_k \right) e_k + Q_k \lambda_{k+1} + A_k K_k R_k K_k^T A_k^T \lambda_{k+1} - e_{k+1} \right) \Bigg] - \frac{1}{2}\lambda_0^T P_0^- {P_0^-}^{-1} P_0^- \lambda_0 \\ &= \sum_{k=0}^{K-1} \left[\frac{1}{2\gamma^2} e_k^T S_k e_k + \lambda_{k+1}^T \left(A_k \left(I - K_k C_k \right) e_k + \frac{1}{2} Q_k \lambda_{k+1} + \frac{1}{2} A_k K_k R_k K_k^T A_k^T \lambda_{k+1} - e_{k+1} \right) \right] \\ &\quad - \frac{1}{2}\lambda_0^T P_0^- \lambda_0 \end{aligned} \tag{5.25}$$

Now, we will minimize the cost function with respect to the estimation error. Finding the partial derivative of J with respect to e_k:

$$\frac{\partial J}{\partial e_k}^T = \frac{1}{\gamma^2} S_k e_k + \left(I - K_k C_k \right)^T A_k^T \lambda_{k+1} - \lambda_k \tag{5.26}$$

We will now define a matrix P_k^-, that relates e_k to λ_k:

$$e_k := P_k^- \lambda_k \tag{5.27}$$

The actual value of P_k^- will be determined later. Plugging this expression into our partial derivative, setting it to 0, and solving for λ_k, we get:

$$0 = \frac{1}{\gamma^2} S_k P_k^- \lambda_k + \left(I - K_k C_k \right)^T A_k^T \lambda_{k+1} - \lambda_k \tag{5.28}$$

$$\lambda_k = \left(I - \frac{1}{\gamma^2} S_k P_k^- \right)^{-1} \left(I - K_k C_k \right)^T A_k^T \lambda_{k+1} \tag{5.29}$$

Re-expressing J using this value, we get:

$$\begin{aligned}
J &= \sum_{k=0}^{K-1}\left[\frac{1}{2\gamma^2}\lambda_k^T P_k^- S_k P_k^- \lambda_k + \lambda_{k+1}^T\left(A_k\left(I-K_kC_k\right)P_k^-\lambda_k + \frac{1}{2}Q_k\lambda_{k+1} + \frac{1}{2}A_kK_kR_kK_k^TA_k^T\lambda_{k+1} - e_{k+1}\right)\right] \\
&\quad -\frac{1}{2}\lambda_0^T P_0^- \lambda_0 \\
&= \sum_{k=0}^{K-1}\left[\frac{1}{2\gamma^2}\lambda_{k+1}^T A_k\left(I-K_kC_k\right)\left(I-\frac{1}{\gamma^2}S_kP_k^-\right)^{-1}P_k^-S_kP_k^-\left(I-\frac{1}{\gamma^2}S_kP_k^-\right)^{-1}\left(I-K_kC_k\right)^TA_k^T\lambda_{k+1}\right. \\
&\quad +\lambda_{k+1}^T\left(A_k\left(I-K_kC_k\right)P_k^-\left(I-\frac{1}{\gamma^2}S_kP_k^-\right)^{-1}\left(I-K_kC_k\right)^TA_k^T\lambda_{k+1} + \frac{1}{2}Q_k\lambda_{k+1}\right. \\
&\quad \left.\left.+\frac{1}{2}A_kK_kR_kK_k^TA_k^T\lambda_{k+1} - P_{k+1}^-\lambda_{k+1}\right)\right] - \frac{1}{2}\lambda_0^TP_0^-\lambda_0 \qquad (5.30)
\end{aligned}$$

Grouping common terms, we can further simplify the expression:

$$\begin{aligned}
J &= \sum_{k=0}^{K-1}\left[\frac{1}{2}\lambda_{k+1}^T A_k\left(I-K_kC_k\right)\left(I-\frac{1}{\gamma^2}S_kP_k^-\right)^{-1}\left(\frac{1}{\gamma^2}S_kP_k^- + 2\left(I-\frac{1}{\gamma^2}S_kP_k^-\right)\right)P_k^-\right. \\
&\quad \times\left(I-\frac{1}{\gamma^2}S_kP_k^-\right)^{-1}\left(I-K_kC_k\right)^TA_k^T\lambda_{k+1} + \frac{1}{2}\lambda_{k+1}^TQ_k\lambda_{k+1} + \frac{1}{2}\lambda_{k+1}^TA_kK_kR_kK_k^TA_k^T\lambda_{k+1} \\
&\quad \left.-\lambda_{k+1}^TP_{k+1}^-\lambda_{k+1}\right] - \frac{1}{2}\lambda_0^TP_0^-\lambda_0 \\
&= \sum_{k=0}^{K-1}\left[\frac{1}{2}\lambda_{k+1}^T A_k\left(I-K_kC_k\right)\left(I-\frac{1}{\gamma^2}S_kP_k^-\right)^{-1}\left(2I-\frac{1}{\gamma^2}S_kP_k^-\right)P_k^-\left(I-\frac{1}{\gamma^2}S_kP_k^-\right)^{-1}\right. \\
&\quad \times\left(I-K_kC_k\right)^TA_k^T\lambda_{k+1} + \frac{1}{2}\lambda_{k+1}^TQ_k\lambda_{k+1} + \frac{1}{2}\lambda_{k+1}^TA_kK_kR_kK_k^TA_k^T\lambda_{k+1} \\
&\quad \left.-\lambda_{k+1}^TP_{k+1}^-\lambda_{k+1}\right] - \frac{1}{2}\lambda_0^TP_0^-\lambda_0 \qquad (5.31)
\end{aligned}$$

For simplicity in further derivation, we will define $\tilde{P}_k$:

$$\tilde{P}_k := \left(I-\frac{1}{\gamma^2}S_kP_k^-\right)^{-1}\left(2I-\frac{1}{\gamma^2}S_kP_k^-\right)P_k^-\left(I-\frac{1}{\gamma^2}S_kP_k^-\right)^{-1} \qquad (5.32)$$

This simplifies the expression for J into:

$$\begin{aligned} J &= \sum_{k=0}^{K-1} \left[\frac{1}{2}\lambda_{k+1}^T A_k \left(I - K_k C_k\right) \tilde{P}_k \left(I - K_k C_k\right)^T A_k^T \lambda_{k+1} + \frac{1}{2}\lambda_{k+1}^T Q_k \lambda_{k+1} + \frac{1}{2}\lambda_{k+1}^T A_k K_k R_k K_k^T A_k^T \lambda_{k+1} \right. \\ &\qquad \left. - \lambda_{k+1}^T P_{k+1}^- \lambda_{k+1} \right] - \frac{1}{2}\lambda_0^T P_0^- \lambda_0 \end{aligned} \tag{5.33}$$

Now, to find the optimal gain, K_k, we can minimize J directly:

$$\frac{\partial J}{\partial K_k^T A_k^T \lambda_{k+1}}^T = -C_k \tilde{P}_k \left(I - K_k C_k\right)^T A_k^T \lambda_{k+1} + R_k K_k^T A_k^T \lambda_{k+1} = 0 \tag{5.34}$$

Canceling the $A_k^T \lambda_{k+1}$ terms (they are non-zero), and rearranging to solve for K_k:

$$\begin{aligned} 0 &= -C_k \tilde{P}_k \left(I - K_k C_k\right)^T + R_k K_k^T \\ 0 &= -C_k \tilde{P}_k + C_k \tilde{P}_k C_k^T K_k^T + R_k K_k^T \\ K_k^T &= \left(C_k \tilde{P}_k C_k^T + R_k\right)^{-1} C_k \tilde{P}_k \\ K_k &= \tilde{P}_k C_k^T \left(C_k \tilde{P}_k C_k^T + R_k\right)^{-1} \end{aligned} \tag{5.35}$$

As can be seen, K_k is of the same form as the Kalman gain in the Kalman Filter. Next, we can optimize J with respect to λ_{k+1}:

$$\frac{\partial J}{\partial \lambda_{k+1}}^T = A_k \left(I - K_k C_k\right) \tilde{P}_k \left(I - K_k C_k\right)^T A_k^T \lambda_{k+1} + Q_k \lambda_{k+1} + A_k K_k R_k K_k^T A_k^T \lambda_{k+1} - P_{k+1}^- \lambda_{k+1} = 0 \tag{5.36}$$

Canceling the λ_{k+1} terms and solving for P_{k+1}^-:

$$\begin{aligned} 0 &= A_k \left(I - K_k C_k\right) \tilde{P}_k \left(I - K_k C_k\right)^T A_k^T + Q_k + A_k K_k R_k K_k^T A_k^T - P_{k+1}^- \\ P_{k+1}^- &= A_k \left(I - K_k C_k\right) \tilde{P}_k \left(I - K_k C_k\right)^T A_k^T + Q_k + A_k K_k R_k K_k^T A_k^T \end{aligned} \tag{5.37}$$

Gathering common terms:

$$P_{k+1}^- = A_k \left[\left(I - K_k C_k\right) \tilde{P}_k \left(I - K_k C_k\right)^T + K_k R_k K_k^T\right] A_k^T + Q_k \tag{5.38}$$

If we define P_k based on the above equation, and use it to further simplify P_{k+1}^{-}:

$$\begin{aligned} P_k &:= \left(I - K_k C_k\right) \tilde{P}_k \left(I - K_k C_k\right)^T + K_k R_k K_k^T && (5.39) \\ P_{k+1}^{-} &= A_k P_k A_k^T + Q_k && (5.40) \end{aligned}$$

As can be seen, P_{k+1}^{-} is in a familiar form to the covariance prediction step in the Kalman Filter. In addition, P_k is of the same form as equation 3.18, so we can apply the same manipulation to simplify it even further:

$$P_k = \left(I - K_k C_k\right) \tilde{P}_k \tag{5.41}$$

Thus, all the expressions in the RKF are found. One additional aspect needs to be considered, which to ensure that the extremizing value of K_k is a minimum of the cost function. To determine the type of a stationary point, the second derivative must be considered. For minima, the expression must be positive definite. In our case:

$$\frac{\partial^2 J}{\partial \left(K_k^T A_k^T \lambda_{k+1}\right)^2}^T = C_k \tilde{P}_k C_k^T + R_k > 0 \tag{5.42}$$

Since R_k is defined as positive definite, and C_k and K_k are just linear operators, this requires that P_k is also positive definite. This assertion is the limitation on the value of γ^2.

With this final condition, the derivation is complete.

Chapter 6

Robust Extended Kalman Filter (REKF)

6.1 Introduction

As the final section to the set of derivations, we will apply the RKF to nonlinear systems, constructing the Robust Extended Kalman Filter (REKF). The same approach (as th EKF) to linearization using Taylor series expansions will be applied, and (unsurprisingly), the resulting algorithm will be a familiar form. The general nonlinear form of the system model is as follows:

$$x_{k+1} = f\left(x_k, u_k\right) + \tilde{w}_k \tag{6.1}$$

$$z_k = h\left(x_k\right) + \tilde{n}_k \tag{6.2}$$

Where $\tilde{w}_k$ and $\tilde{n}_k$ are the additive noise and unmodeled system dynamics.

6.2 Algorithm

The algorithm largely resembles the form of the EKF:

$$\hat{x}_{k+1}^- = f\left(\hat{x}_k, u_k\right) \tag{6.3}$$

$$P_k^- = F_k P_k F_k^T + Q_k \tag{6.4}$$

$$\tilde{P}_k = \frac{1}{2}\left(I - \frac{1}{\gamma^2} S_k P_k^-\right)^{-1}\left(2I - \frac{1}{\gamma^2} S_k P_k^-\right) P_k^- \left(I - \frac{1}{\gamma^2} S_k P_k^-\right)^{-1} \tag{6.5}$$

$$K_k = \tilde{P}_k H_k^T \left(H_k \tilde{P}_k H_k^T + R_k\right)^{-1} \tag{6.6}$$

$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - h\left(\hat{x}_k^-\right)\right) \tag{6.7}$$

$$P_k = \left(I - K_k H_k\right) \tilde{P}_k \tag{6.8}$$

Again, as $\gamma^2 \to \infty$, the REKF reduces to the EKF. Just like the EKF, a combination of nonlinear estimates and linear approximations are used for the various predictor-corrector steps. While Q_k and R_k can be taken as analogous to their counterparts in the EKF, in the REKF they capture the additive noise as well as unmodeled system dynamics (and linearization errors), so they may need to be inflated to account for those effects.

6.3 Derivation

Again, we will need to begin by assuming a form for the estimator:

$$\hat{x}_{k+1}^{-} := f\left(\hat{x}_k, u_k\right) \tag{6.9}$$

$$\hat{x}_k := \hat{x}_k^{-} + K_k\left(z_k - h\left(\hat{x}_k^{-}\right)\right) \tag{6.10}$$

As there are nonlinear functions, we will conduct a linearization step. First, we obtain a Taylor series expansion of the state transition equation around $\hat{x}_k$:

$$f\left(x_k, u_k\right) = f\left(\hat{x}_k, u_k\right) + \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_k}\left(x_k - \hat{x}_k\right) + H.O.T \tag{6.11}$$

$$= f\left(\hat{x}_k, u_k\right) + F_k\left(x_k - \hat{x}_k\right) + \Delta_f\left(x_k\right) \tag{6.12}$$

Where F_k is the Jacobian of f with respect to x, and $\Delta_f\left(x_k\right)$ is the higher-order terms of the Taylor series expansion. Now, for the measurement equation, we expand around $\hat{x}_k^{-}$:

$$h\left(x_k\right) = h\left(\hat{x}_k^{-}\right) + \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_k^{-}}\left(x_k - \hat{x}_k^{-}\right) + H.O.T \tag{6.13}$$

$$= h\left(\hat{x}_k^{-}\right) + H_k\left(x_k - \hat{x}_k^{-}\right) + \Delta_h\left(x_k\right) \tag{6.14}$$

Similarly, H_k is the Jacobian of h with respect to x, and $\Delta_h\left(x_k\right)$ represents the higher-order terms. With these expansions, we can consider the error dynamics:

$$e_{k+1} = x_{k+1} - \hat{x}_{k+1}^{-} \tag{6.15}$$

$$\begin{aligned} &= f\left(x_k, u_k\right) + \tilde{w}_k - f\left(\hat{x}_k, u_k\right) \\ &= f\left(\hat{x}_k, u_k\right) + F_k\left(x_k - \hat{x}_k\right) + \Delta_f x_k + \tilde{w}_k - f\left(\hat{x}_k, u_k\right) \end{aligned}$$

$$= F_k x_k - F_k \hat{x}_k + \Delta_f x_k + \tilde{w}_k \tag{6.16}$$

Using the assumed estimator for $\hat{x}_k$:

$$\begin{aligned} e_{k+1} &= F_k x_k - F_k \left(\hat{x}_k^- + K_k \left(z_k - h\left(\hat{x}_k^-\right)\right)\right) + \Delta_f\left(x_k\right) + \tilde{w}_k \\ &= F_k x_k - F_k \hat{x}_k^- - F_k K_k \left(h\left(x_k\right) + \tilde{n}_k - h\left(\hat{x}_k^-\right)\right) + \Delta_f\left(x_k\right) + \tilde{w}_k \\ &= F_k e_k - F_k K_k \left(h\left(\hat{x}_k^-\right) + H_k\left(x_k - \hat{x}_k^-\right) + \Delta_h\left(x_k\right) + \tilde{n}_k - h\left(\hat{x}_k^-\right)\right) + \Delta_f\left(x_k\right) + \tilde{w}_k \\ &= F_k e_k - F_k K_k \left(H_k e_k + \Delta_h\left(x_k\right) + \tilde{n}_k\right) + \Delta_f\left(x_k\right) + \tilde{w}_k \end{aligned} \tag{6.17}$$

If we define w_k and n_k as below:

$$w_k := \tilde{w}_k + \Delta_f\left(x_k\right) \tag{6.18}$$

$$n_k := \tilde{n}_k + \Delta_h\left(x_k\right) \tag{6.19}$$

The variables capture the additive noise and unmodeled system dynamics, along with the higher order terms of linearization approximations (which, in spirit, can be lumped into the unmodeled system dynamics group). With these definitions, we obtain:

$$\begin{aligned} e_{k+1} &= F_k e_k - F_k K_k \left(H_k e_k + n_k\right) + w_k \\ &= F_k \left(I - K_k H_k\right) e_k + w_k - F_k K_k n_k \end{aligned} \tag{6.20}$$

This is of the exact same form as the error dynamics relation in the RKF case, and as such, the completion of the derivation follows the exact same steps.

Chapter 7

Simultaneous Localization and Mapping (SLAM)

7.1 Introduction

The Simultaneous Localization and Mapping (SLAM) problem is the process by which a mobile robot can build a map and localize itself within the map at the same time. The solution to this problem would provide a large component of the ability for a robot to autonomously navigate through any unknown locale, and as such, the solution is viewed as a holy grail of robotics. Various solutions have been proposed, each with their own sets of assumptions, as well as limitations (often in terms of scaling or consistency). For a baseline comparison, we will start with one of the earliest proposed solutions to the SLAM problem, and compare it to this newly postulated approach.

7.2 EKF-SLAM

7.2.1 Introduction

The EKF-SLAM solution augments the robot pose state with all the landmark locations, creating a large state to estimate and update with the Extended Kalman Filter. It has been implemented with a good degree of success in some applications, but the linearization and Gaussian noise assumptions lead to inconsistencies that are unacceptable in the general case. These inconsistencies tend to be very noticeable in the loop-closure scenario, where the algorithm proves to be overconfident in its estimates of the landmark locations. However, it is the most basic of the SLAM solutions, so it provides a good initial attempt.

7.2.2 Linearization

Starting with the state transition and measurement models in equations 2.3, 2.4, and 2.5, we can apply Taylor series expansions to get the relations in equations 4.10 and 4.15:

$$\begin{aligned} \Delta x_k &= F_k \delta x_{k-1} && (7.1) \\ \Delta z_k &= H_k \delta x_k && (7.2) \end{aligned}$$

Calculating out the Jacobian for the motion update, we get:

$$F_k := \left.\frac{\partial f}{\partial x}\right|_{\mu_{k-1}} = I + \left.\begin{bmatrix} 0 & 0 & -v_k \Delta t \sin\theta_{r_{k-1}} \\ 0 & 0 & v_k \Delta t \cos\theta_{r_{k-1}} \\ 0 & 0 & 0 \end{bmatrix}\right|_{\mu_{k-1}} \tag{7.3}$$

In addition, for this particular implementation, we will find the Jacobian of f with respect to u:

$$G_k := \left.\frac{\partial f}{\partial u}\right|_{\mu_{k-1}} = \left.\begin{bmatrix} \Delta t \cos\theta_{r_{k-1}} & 0 \\ \Delta t \sin\theta_{r_{k-1}} & 0 \\ 0 & \Delta t \end{bmatrix}\right|_{\mu_{k-1}} \tag{7.4}$$

This G_k term will be used for in the covariance update step due to how the noise parameters were defined.

For the measurement model, we calculate the Jacobian with respect to the robot pose as well as the landmark location. For conciseness, the general form is as follows:

$$\begin{aligned} \delta_x &= x_l - x_r && (7.5) \\ \delta_y &= y_l - y_r && (7.6) \\ q &= \left(\delta_x^2 + \delta_y^2\right) && (7.7) \\ H_k &:= \left.\frac{\partial h}{\partial x}\right|_{\mu_k^-} = \left.\begin{bmatrix} -\frac{\delta_x}{\sqrt{q}} & -\frac{\delta_y}{\sqrt{q}} & 0 & \dots & \frac{\delta_x}{\sqrt{q}} & \frac{\delta_y}{\sqrt{q}} \\ \frac{\delta_y}{q} & -\frac{\delta_x}{q} & -1 & \dots & -\frac{\delta_y}{q} & \frac{\delta_x}{q} \end{bmatrix}\right|_{\mu_k^-} && (7.8) \end{aligned}$$

7.2.3 Implementation

The Jacobians derived above are used in the algorithm. In terms of implementation peculiarities, projection matrices are used to isolate the portions of the state that it is desired to modify, and newly observed landmarks are initialized to the measurement (as there is no other information). The algorithm is initialized with large landmark covariances (to indicate no knowledge), and the initial pose with very low uncertainty. For this implementation, the process noise is factored directly into the input (as defined in Section 2.4). To maintain generality, $\tilde{Q}_k$ is defined, making use of the G_k term to achieve the appropriate dimensions. In addition, for simplicity, identification of observed landmarks and the total number are known a priori, so we can isolate the effects of the various filter implementations. In real life implementation, these concerns must also be integrated into the algorithm. The following page details the entire implementation for each time step.

Algorithm 1 EKF-SLAM $(\mu_{k-1}, \Sigma_{k-1}, u_k, z_k, Q_k, R_k)$

```
1 :  $P = \begin{bmatrix} 1 & 0 & 0 & 0\ldots0 \\ 0 & 1 & 0 & 0\ldots0 \\ 0 & 0 & 1 & \underbrace{0\ldots0}_{2N} \end{bmatrix}$
2 :  $\mu_k^- = \mu_{k-1} + P^T \begin{bmatrix} v_k \Delta t \cos(\mu_{k-1,\theta}) \\ v_k \Delta t \sin(\mu_{k-1,\theta}) \\ \omega_k \Delta t \end{bmatrix}$
3 :  $F_k = I + P^T \begin{bmatrix} 0 & 0 & -v_k \Delta t \sin(\mu_{k-1,\theta}) \\ 0 & 0 & v_k \Delta t \cos(\mu_{k-1,\theta}) \\ 0 & 0 & 1 \end{bmatrix} P$
4 :  $G_k = \begin{bmatrix} \Delta t \cos(\mu_{k-1,\theta}) & 0 \\ \Delta t \sin(\mu_{k-1,\theta}) & 0 \\ 0 & \Delta t \end{bmatrix}$
5 :  $\tilde{Q}_k = G_k Q_k G_k^T$
6 :  $\Sigma_k^- = F_k \Sigma_{k-1} F_k^T + P^T \tilde{Q}_k P$
7 :  for all observed features $z_k^i = \left(r_k^i, \phi_k^i\right)^T$ do:
8 :    if landmark i never seen before:
9 :      $\begin{bmatrix} \mu_{i,x}^- \\ \mu_{i,y}^- \end{bmatrix} = \begin{bmatrix} \mu_{k,x}^- \\ \mu_{k,y}^- \end{bmatrix} + \begin{bmatrix} r_k^i \cos\left(\phi_k^i + \mu_{k,\theta}^-\right) \\ r_k^i \sin\left(\phi_k^i + \mu_{k,\theta}^-\right) \end{bmatrix}$
10 :   endif
11 :   $\delta = \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = \begin{bmatrix} \mu_{i,x}^- - \mu_{k,x}^- \\ \mu_{i,y}^- - \mu_{k,y}^- \end{bmatrix}$
12 :   $q = \delta^T \delta$
13 :   $\hat{z}_k^i = \begin{bmatrix} \sqrt{q} \\ \text{atan2}(\delta_y, \delta_x) \end{bmatrix}$
14 :   $P = \begin{bmatrix} 1 & 0 & 0 & 0\ldots0 & 0 & 0 & 0\ldots0 \\ 0 & 1 & 0 & 0\ldots0 & 0 & 0 & 0\ldots0 \\ 0 & 0 & 1 & 0\ldots0 & 0 & 0 & 0\ldots0 \\ 0 & 0 & 0 & 0\ldots0 & 1 & 0 & 0\ldots0 \\ 0 & 0 & 0 & \underbrace{0\ldots0}_{2i-2} & 0 & 1 & \underbrace{0\ldots0}_{2N-2i} \end{bmatrix}$
15 :   $H_k^i = \frac{1}{q} \begin{bmatrix} -\delta_x\sqrt{q} & -\delta_y\sqrt{q} & 0 & \delta_x\sqrt{q} & \delta_x\sqrt{q} \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{bmatrix} P$
16 :   $K_k^i = \Sigma_k^- H_k^{iT} \left(H_k^i \Sigma_k^- H_k^{iT} + R_k\right)^{-1}$
17 :   $\mu_k^- = \mu_k^- + K_k^i \left(z_k^i - \hat{z}_k^i\right)$
18 :   $\Sigma_k^- = \left(I - K_k^i H_k^i\right) \Sigma_k^-$
19 : endfor
20 : $\mu_k = \mu_k^-$
21 : $\Sigma_k = \Sigma_k^-$
22 : return $\mu_k, \Sigma_k$
```

7.3 REKF-SLAM

7.3.1 Introduction

The REKF-SLAM solution is very similar to the EKF-SLAM, as can be seen in the relations between the two underlying algorithms. The state vector is again the augmented pose and landmark locations, and the REKF applied on it. It is expected that using a filter designed with robustness in mind will result in better estimates and a more consistent filter performance.

Historically, the standard approach for improved robustness in EKF-SLAM implementation often is an artificial inflation of Q_k and R_k, the noise covariances, determined through experimentation and heuristics. This results in an inflation of the estimation covariance, improving consistency. However, as can be seen in the algorithm, the REKF defines this inflation of the estimation covariance more methodically, and is thusly preferred, if it proves to have an advantageous performance.

While the algorithm explicitly removes experimentation in some steps, unfortunately, the smallest value of γ^2 will need to be determined iteratively, asserting that P_k be positive definite at every time step.

7.3.2 Linearization

The exact same linearization approach is applied in the REKF as the EKF, except that the expansions are around $\hat{x}_k$ instead of μ_k (minor change in notation, the expressions are effectively the same).

$$F_k := \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1}} \tag{7.9}$$

$$G_k := \left.\frac{\partial f}{\partial u}\right|_{\hat{x}_{k-1}} \tag{7.10}$$

$$H_k := \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_k^-} \tag{7.11}$$

7.3.3 Implementation

The implementation of REKF-SLAM uses the Jacobians defined above, and resembles EKF-SLAM, with the use of projection matrices, and $\tilde{Q}_k$. The modifications include the additions of S_k and γ^2, as well the use of the function *LOCAL*$\tilde{P}$, which modifies the relevant sections of $\tilde{P}_k$ as per the REKF algorithm at each time step.

Additionally, the $\tilde{P}_k$ expression was modified to include a $\frac{1}{2}$ coefficient. In other works, such as [3] and [7], it is said that the EKF is recovered when $\gamma^2 \to \infty$, as the optimal result without a bound on the cost function should return to the Kalman Filter's goal of reducing the average overall error. This also should give an insight to the choice of γ^2, as it represents a balance between the KF and RKF's estimation goals. Unfortunately, in the derivation presented in the earlier chapter for the RKF, this would not be the case. Upon implementation, it was found that the coefficient was necessary to obtain working results. This modification will be discussed further in the following chapters.

Algorithm 2 REKF-SLAM $\left(\hat{x}_{k-1}, P_{k-1}, u_k, z_k, Q_k, R_k, S_k, \gamma^2\right)$

```
1 :  $P = \begin{bmatrix} 1 & 0 & 0 & 0\dots0 \\ 0 & 1 & 0 & 0\dots0 \\ 0 & 0 & 1 & \underbrace{0\dots0}_{2N} \end{bmatrix}$
2 :  $\hat{x}_k^- = \hat{x}_{k-1} + P^T \begin{bmatrix} v_k\Delta t\cos(\hat{x}_{k-1,\theta}) \\ v_k\Delta t\sin(\hat{x}_{k-1,\theta}) \\ \omega_k\Delta t \end{bmatrix}$
3 :  $F_k = I + P^T \begin{bmatrix} 0 & 0 & -v_k\Delta t\sin(\hat{x}_{k-1,\theta}) \\ 0 & 0 & v_k\Delta t\cos(\hat{x}_{k-1,\theta}) \\ 0 & 0 & 1 \end{bmatrix} P$
4 :  $G_k = \begin{bmatrix} \Delta t\cos(\hat{x}_{k-1,\theta}) & 0 \\ \Delta t\sin(\hat{x}_{k-1,\theta}) & 0 \\ 0 & \Delta t \end{bmatrix}$
5 :  $\tilde{Q}_k = G_k Q_k G_k^T$
6 :  $P_k^- = F_k P_{k-1} F_k^T + P^T \tilde{Q}_k P$
7 :  for all observed features $z_k^i = \left(r_k^i, \phi_k^i\right)^T$ do:
8 :    if landmark i never seen before:
9 :      $\begin{bmatrix} \hat{x}_{i,x}^- \\ \hat{x}_{i,y}^- \end{bmatrix} = \begin{bmatrix} \hat{x}_{k,x}^- \\ \hat{x}_{k,y}^- \end{bmatrix} + \begin{bmatrix} r_k^i\cos\left(\phi_k^i + \hat{x}_{k,\theta}^-\right) \\ r_k^i\sin\left(\phi_k^i + \hat{x}_{k,\theta}^-\right) \end{bmatrix}$
10 :   endif
11 :   $\delta = \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = \begin{bmatrix} \hat{x}_{i,x}^- - \hat{x}_{k,x}^- \\ \hat{x}_{i,y}^- - \hat{x}_{k,y}^- \end{bmatrix}$
12 :   $q = \delta^T\delta$
13 :   $\hat{z}_k^i = \begin{bmatrix} \sqrt{q} \\ \text{atan2}(\delta_y, \delta_x) \end{bmatrix}$
14 :   $P = \begin{bmatrix} 1 & 0 & 0 & 0\dots0 & 0 & 0 & 0\dots0 \\ 0 & 1 & 0 & 0\dots0 & 0 & 0 & 0\dots0 \\ 0 & 0 & 1 & 0\dots0 & 0 & 0 & 0\dots0 \\ 0 & 0 & 0 & 0\dots0 & 1 & 0 & 0\dots0 \\ 0 & 0 & 0 & \underbrace{0\dots0}_{2i-2} & 0 & 1 & \underbrace{0\dots0}_{2N-2i} \end{bmatrix}$
15 :   $H_k^i = \frac{1}{q}\begin{bmatrix} -\delta_x\sqrt{q} & -\delta_y\sqrt{q} & 0 & \delta_x\sqrt{q} & \delta_x\sqrt{q} \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{bmatrix} P$
16 :   $\tilde{P}_k^i = \textit{LOCAL}\tilde{P}\left(\tilde{P}_k, i, S_k, \gamma^2\right)$
17 :   $K_k^i = \tilde{P}_k^i H_k^{iT}\left(H_k^i\tilde{P}_k^i H_k^{iT} + R_k\right)^{-1}$
18 :   $\hat{x}_k^- = \hat{x}_k^- + K_k^i\left(z_k^i - \hat{z}_k^i\right)$
19 :   $\tilde{P}_k = \left(I - K_k^i H_k^i\right)\tilde{P}_k^i$
20 :  endfor
21 :  $\hat{x}_k = \hat{x}_k^-$
22 :  $P_k = \tilde{P}_k$
23 :  return $\hat{x}_k, P_k$
```

Algorithm 3 LOCAL$\tilde{P}$ $\left(\tilde{P}_k, i, S_k, \gamma^2\right)$

```
1 :  $\bar{P}_k = \left[ \begin{array}{cc} \tilde{P}_k\left(1:3, 1:3\right) & \tilde{P}_k\left(\left(2i+2\right):\left(2i+3\right), 1:3\right) \\ \tilde{P}_k\left(1:3, \left(2i+2\right):\left(2i+3\right)\right) & \tilde{P}_k\left(\left(2i+2\right):\left(2i+3\right), \left(2i+2\right):\left(2i+3\right)\right) \end{array} \right]$
2 :  $\tilde{P}_k^i = \frac{1}{2}\left(I - \frac{1}{\gamma^2} S_k \bar{P}_k -\right)^{-1}\left(2I - \frac{1}{\gamma^2} S_k \bar{P}_k\right)\bar{P}_k\left(I - \frac{1}{\gamma^2} S_k \bar{P}_k\right)^{-1}$
3 :  $\tilde{P}_k\left(1:3, 1:3\right) = \tilde{P}_k^i\left(1:3, 1:3\right)$
4 :  $\tilde{P}_k\left(1:3, \left(2i+2\right):\left(2i+3\right)\right) = \tilde{P}_k^i\left(1:3, \left(2i+2\right):\left(2i+3\right)\right)$
5 :  $\tilde{P}_k\left(\left(2i+2\right):\left(2i+3\right), 1:3\right) = \tilde{P}_k^i\left(\left(2i+2\right):\left(2i+3\right), 1:3\right)$
6 :  $\tilde{P}_k\left(\left(2i+2\right):\left(2i+3\right), \left(2i+2\right):\left(2i+3\right)\right) = \tilde{P}_k^i\left(\left(2i+2\right):\left(2i+3\right), \left(2i+2\right):\left(2i+3\right)\right)$
7 :  return $\tilde{P}_k$
```

Chapter 8

Results

8.1 EKF-SLAM

8.1.1 Cramér-Rao Lower Bound (CRLB)

Prior to the presentation of the simulation results, the Cramér-Rao Lower Bound (CRLB) should be introduced. As detailed in [5], the Jacobians calculated in the EKF algorithm for the linearization step can be improved by using their true values instead of the prior estimates (for simplicity, henceforth denoted as EKFJ). The covariance matrix that comes as a result, known as the CRLB, is a theoretical lower bound for the covariance magnitude. The performance of the EKF algorithm can be assessed against this theoretical limit.

As the EKF theory and practice is well established, it is much more interesting to show the cases where it fails rather than where it works. To inflame the differences, high noise scenarios will be presented. In Figure 8.1, the values $\sigma_\omega = 5^\circ, \sigma_v = 10\sigma_\omega, \sigma_\phi = 1^\circ$, and $\sigma_r = 10\sigma_\phi$ were used, displaying the estimate errors in x, y, and θ. As can be seen, the EKF estimates (in the blue) deviate wildly from the 3σ bound, while the EKFJ (in the red) is much more accurate. While these plots do not tell the entire story as they do not illustrate covariances between elements of the pose, they are a good first approximation.

Since in this case the EKF was inconsistent whereas the EKFJ was not, we can study the plots to identify where the failure occurred. At the end of the first peak in the 3σ plots, the covariance estimate dips below that of the CRLB. Once that occurred, all bets were off and the estimate quality in x, θ, and later on y severely degraded. It was only the loop closure at the end that brought all the estimates back to the correct boundaries.

As a different and possibly more illuminating way to illustrate the problem with the inconsistency of the algorithm, we can look at the estimation of the map at the end of the rover traverse, along with the estimates of the rover position at select intervals. As can be seen in Figure 8.2, the overconfidence results in estimates that do not even come close to containing the true locations (illustrated by the 3σ ellipses). This presents a fundamental problem for landmark recognition (which we ignored in this set of simulations), as statistical belief should provide a good indicator of whether the landmark was seen before or not. However, as these landmarks lie far beyond the 3σ range, it is unlikely in this case that the rover will recognize that the loop has been closed.

Though in the case presented the EKFJ estimates were well behaved, that result cannot be guaranteed in the general case. The following section details some additional tests to demonstrate this point, where visual results are not necessary to illustrate the other effects.

8.1.2 Consistency Tests

To assess the statistical performance of the EKF-SLAM algorithm, consistency trials were designed. In these tests, a map was generated and hundreds of simulation scenarios were generated and run, given preset model parameters, such as noise and trajectory plans. Four situations were considered: when the EKF estimate for the robot pose exceeded the 3-σ estimate of the covariance matrix, when the EKF exceeded the CRLB, as well as the same two bounds for the EKF with True Jacobians estimate. It was hoped that by looking at these counts over a large number of trials that the general performance could be evaluated.

Table 8.1: Consistency tests for EKF-SLAM

Number of Trials	**Noise Characteristics** $\sigma_{\omega,v,\phi,r}$	**Number of Loops**	**Number of Steps**	**EKF** 3σ	**EKF CRLB**	**EKFJ** 3σ	**EKFJ CRLB**
500	$3, 0.52, 1, 0.17$	2	2000	431	429	422	421
100	$3, 0.52, 1, 0.17$	1	200	12	11	12	11
100	$3, 0.3, 1, 0.1$	1	200	17	16	17	15

In a Gaussian distribution, 99.7% of the probability lies in the 3σ range. As a result, it would be expected that if the covariance values were accurate, the number of times that the bounds are exceeded should be fairly low. As can be seen, this is not the case, even with the larger CRLB, and the more accurate EKFJ estimate, though the performance is still slightly better. The performance in the first set of trials is the most telling, with 86.2% of the EKF estimates becoming inconsistent. We can note however, that consistency was maintained in some situations nonetheless, so there are situations where the algorithm performs fine. In general, the EKF-SLAM algorithm fails due to the nonlinear effects, and in real practice, with non-Gaussian noise, the performance would be even worse.

The other two sets of trials were smaller, but were generated to show the effects of the noise magnitudes, especially in those that affect the heading estimate. Only a single loop was traversed, and only over 200 steps, so to decrease the processing time required to run the trials. With less steps and the rover traveling a greater distance, the velocity was higher, so the effects of the noise lower - thus resulting in a much lower inconsistency rate (but still much higher than desired). It can be seen that with lower magnitudes of noise used in the dataset generation, a higher inconsistency rate occurred. This can be attributed to the fact that larger noise parameters in the algorithm allows for greater robustness to noise other than just the Gaussian additive parameters.

A common approach to increase robustness of the EKF algorithm is to artificially inflate the noise parameters Q_k and R_k, effectively preparing the algorithm to handle larger magnitudes of deviation. While it would be expected that this negatively affects the accuracy of the estimates, as the more serious inconsistency issues are addressed, the estimates perform better. Figure 8.3 shows the improvements in performance of a 3x increase in the noise parameters for the EKF-SLAM algorithm. If additional knowledge is known, such as the relative ratios of unmodeled dynamics in the system model compared to the measurement model, relative adjustments of Q_k and R_k could be made to have the algorithm favour the sensor measurements over the odometry, or vice versa.

8.2 REKF-SLAM

The REKF algorithm can be viewed as modifying the noise parameters as well, but through the tuning of the γ^2 bound parameter. This provides a bit more mathematical rigor to the goal of *robustifying* the estimation filter.

Unfortunately, in implementation and simulation of the algorithm, poor results were obtained. The only limitation on the value of γ^2 was that P_k needed to stay positive definite throughout the REKF-SLAM algorithm. For simplicity, the value was varied by hand, so to see the effects of different values. Figure 8.4 illustrates what happens to the filter performance as the bound is lowered, prior to reaching the values where positive definiteness is lost with a value of $\gamma^2 = 550$. This is the same dataset as the one used on the EKF-SLAM algorithm in the previous section.

Other trials involving the use of the other filter forms as established by other researchers gave similar results. Since seeking the optimal γ^2 in this case was done by hand, the lowest possible value was definitely not found in these trials. However, it is expected that when nearing the optimal value, lowering the bound should cause the maximum errors to decrease (once the bound reaches relevant magnitudes). Since the opposite occurred and the estimate got worse as the bound went lower, it is suspected that numerical instabilities may be at fault.

Though the lower limit of γ^2 was not able to be reached, interesting results were obtained in the iterations conducted on the way there. It is expected that lowering the bound would lower the maximum error, but increase the average overall error (since an additional constraint is applied). This occurred as expected, with a hybrid filter (one that optimizes both the KF and the RKF goals) arising as a result. As such, prior to the region of numerical instabilities, improvements on the EKF algorithm were observed. Figure 8.5 shows the improved performance for $\gamma^2 = 1000$, with the REKF-SLAM estimate in green.

Additionally, as the REKF also defines a S_k matrix (which was taken as identity before), relative weighting can be applied to the portions of the state that the filter designer wishes to be more accurate. Motivated by the results in [1] that suggest that reduction in errors in the heading angle, θ, greatly affect the overall performance of the other estimates, $S_{3,3}$ was increased by a factor of 10. The right hand side plots of Figure 8.5 show the effects. Though it is not easily seen in the plots, the overall average error and maximum error in the robot pose estimates were improved with the additional weighting towards θ.

Regardless of the implementation errors, some other pieces of insight can still be obtained. Seeing as how the $\tilde{P}_k$ formula involved the identity matrix subtracting an expression at each step, it was found that the initial landmark uncertainties placed a limitation on the magnitude of γ^2 allowed (so to keep that subexpression positive definite). As such, the entire P_k^- could not be used in the calculation, and a local $\tilde{P}_k^i$ calculation was developed. This limitation is purely a construct of this simulation scenario, as all the landmarks are encoded *a priori*. In real implementation, the state is grown as the rover travels and sees new landmarks, which would definitely not be instantiated with such large covariances.

To address the missing $\frac{1}{2}$ term, without it, only very large values of γ^2 (near infinity) completed a run, while all the other values caused the assertion of positive definiteness for P_k to fail. When the guarantee on P_k is lost, the algorithm switches over into maximizing the error - which could lead to a catastrophic failure of the system. As such, the code aborted whenever this was the case. It was found that including the $\frac{1}{2}$ coefficient allowed for more reasonable (lower) values to be used for γ^2, but seeing as how the algorithm did not perform as desired, definite claims cannot be made.

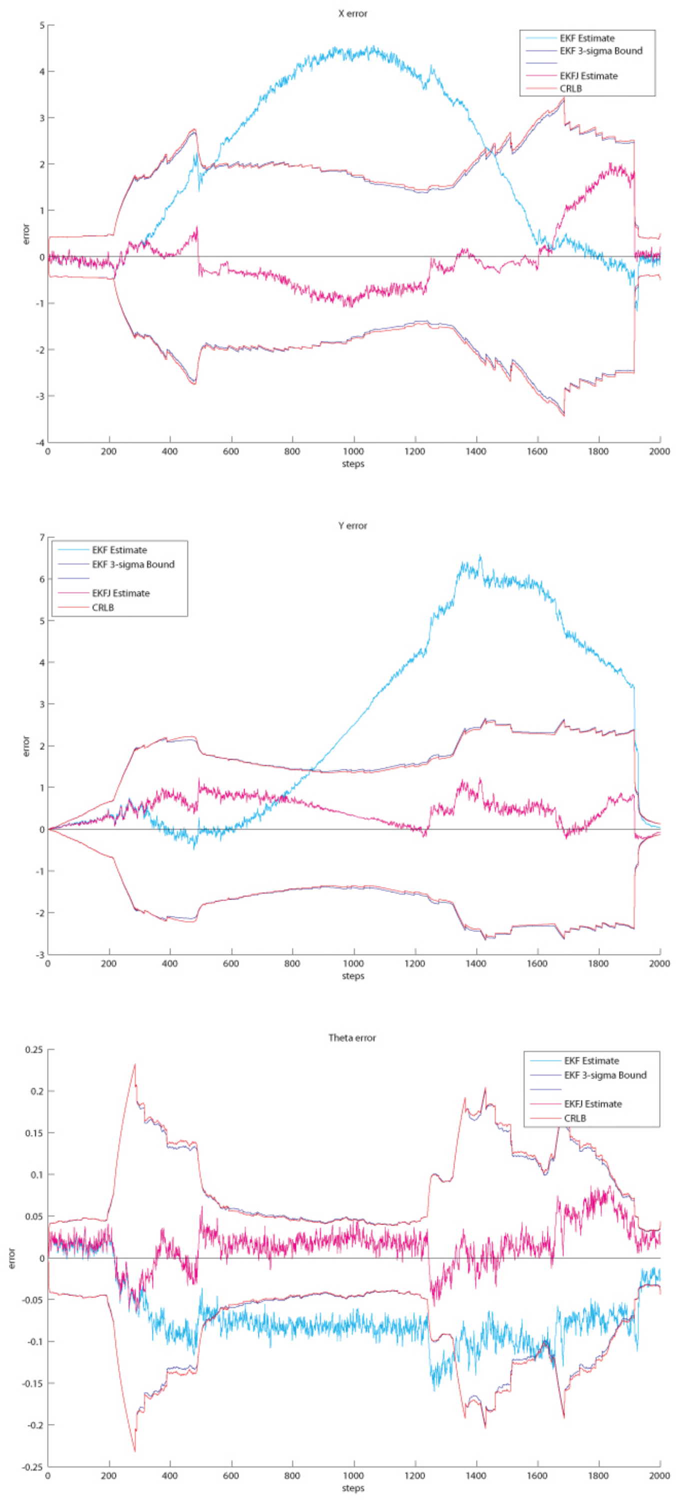

Figure 8.1: Example of EKF estimate error inconsistencies in robot pose: (x, y, θ).

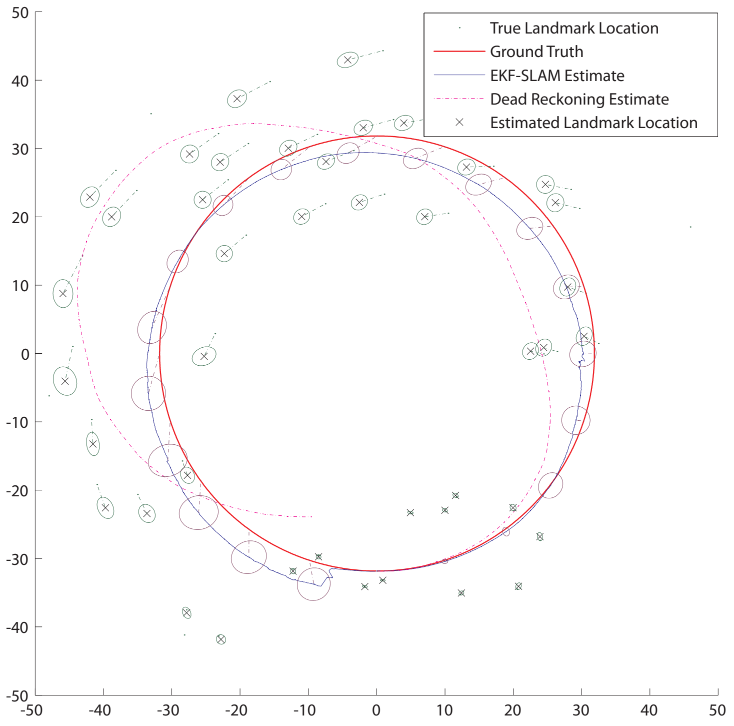

Figure 8.2: Example of landmark inconsistencies.

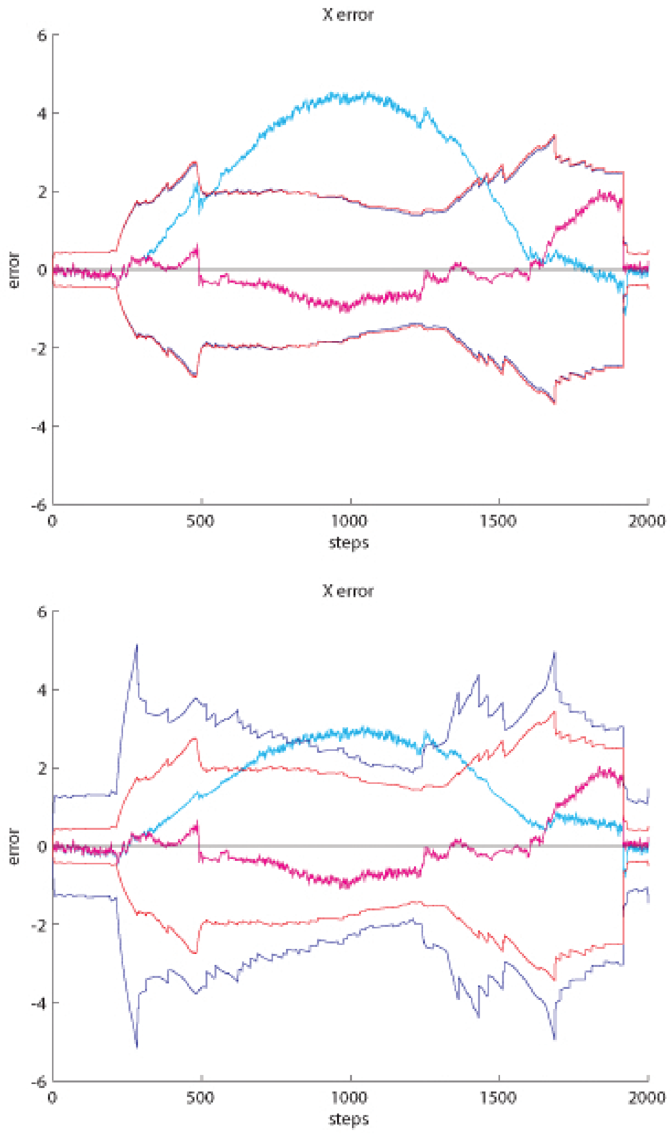

Figure 8.3: Example of estimate performance increase due to 3x artificial noise inflation.

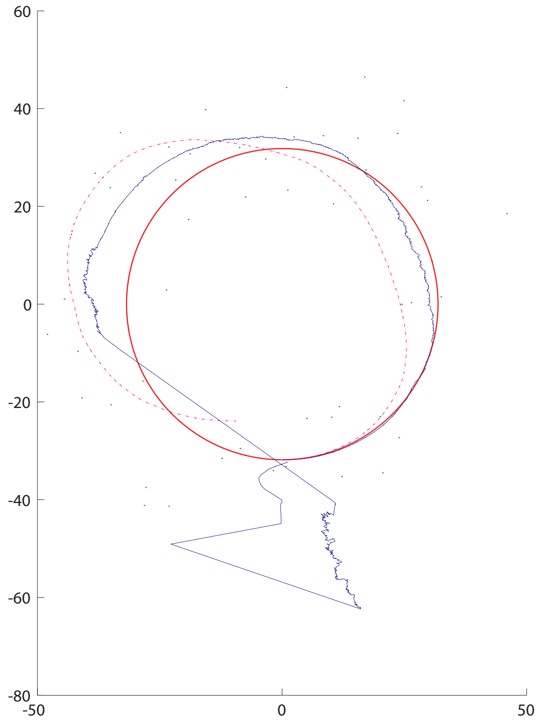

Figure 8.4: Example of REKF filter estimate (blue) failure for $\gamma^2 = 550$.

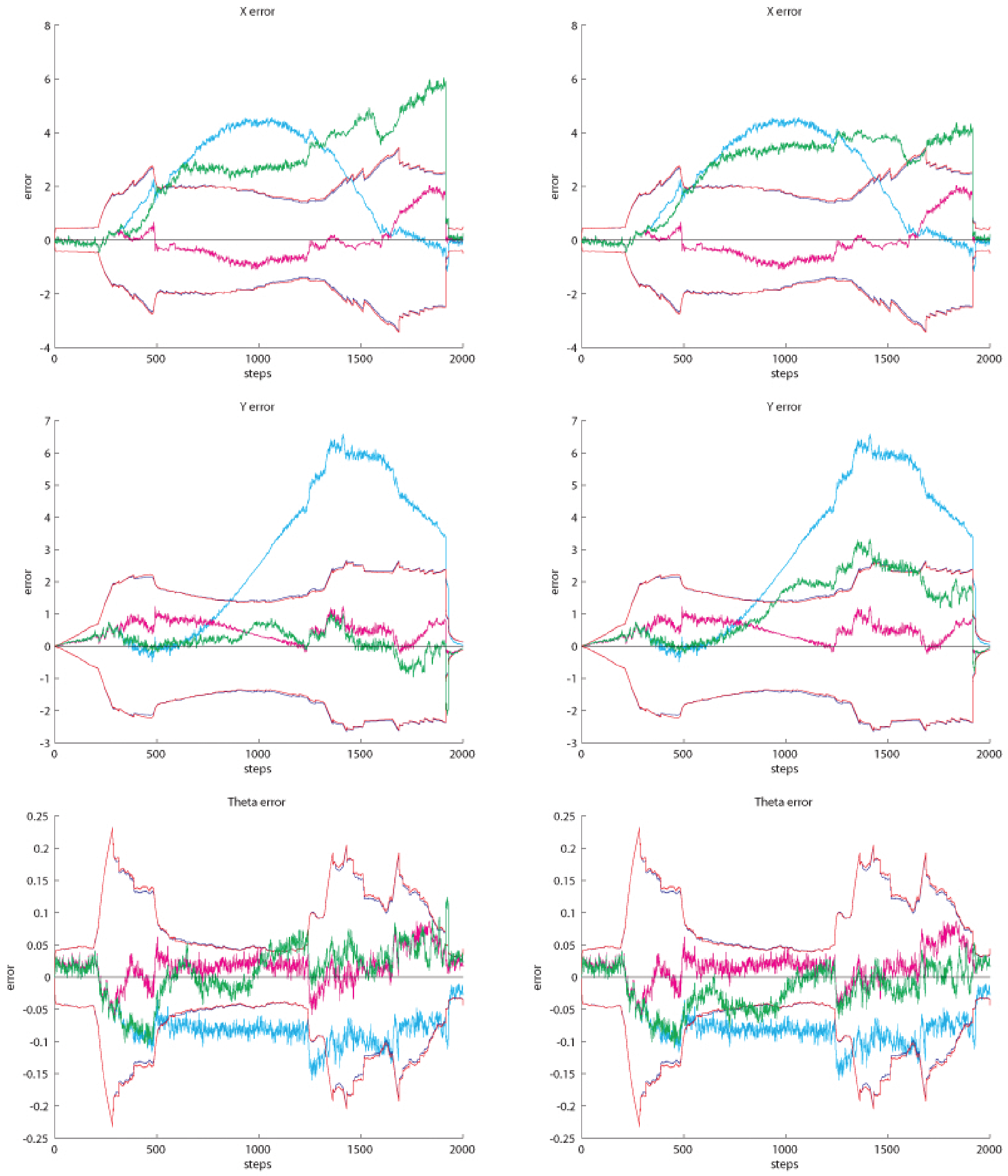

Figure 8.5: Example of hybrid filter performance for $\gamma^2 = 1000$.
(left) $S_{k3,3} = 1$, (right) $S_{k3,3} = 10$

Chapter 9

Conclusions and Recommendations

The EKF, in construction, violates the assumptions as set forth by the derivation of the Kalman Filter. As such, it is not surprising that even just in simulation with Gaussian noise, the non-linear effects lead to inconsistent results. The CRLB provides a theoretical bound on the confidence of the algorithm, and while the EKF can be seen in the results to be overconfident in some situations, even the EKF with True Jacobians does not always perform well due the nonlinear effects.

However, the EKF is still commonly used in practice, because there are methods of coping with these violations of the basic assumptions. By increasing the noise parameters Q_k and R_k, the system becomes more robust - as it is prepared to handle stronger deviations than just the noise. Unfortunately, these parameter adjustments are usually made on the basis of heuristics by experienced engineers.

The REKF approach does not assume any particular form for the noise, and lumps the unmodeled system dynamics into perturbing error terms. The derivation formulation presents a game between Nature and the filter designer, with opposing goals. The resulting algorithm resembles the Kalman Filter closely, and the effect of varying γ^2 can be viewed as modifications of the noise parameters Q_k and R_k. In effect, the REKF presents a mathematical method of conducting the parameter adjustments. While this presents more rigor in the filter design, the cost function unfortunately cannot be optimized directly, and the lower bound of γ^2 must be obtained experimentally. In addition to choosing the value through experimentation, consideration must be given to future applications as well, as when the filter fails, it fails catastrophically. A possibility to consider to address this is to define a dynamic bound, γ_k^2. The decision on the dynamic bound could be influenced by knowledge of the filter performance under regular circumstances. Though computationally inefficient, an example would be to run a standard EKF-SLAM algorithm in parallel to utilize the covariance bounds. Despite these shortcomings, γ^2 is a much more intuitive parameter to adjust, and the introduction of S_k allows for additional weighting towards the heading error, which was shown to have a positive effect on the efficacy of the entire algorithm.

In conducting the derivation for the RKF, a concise method was sought, to provide a clean document for future investigations into this approach. While that was achieved, a number of issues arose. The first is that the resulting form is considerably more complex than what results in other literature, the second was the missing $\frac{1}{2}$ coefficient, and the third is in the implementation. While it is believed that no errors were made in the derivation, the absence of the coefficient and subsequent requirement in the implementation brings the derivation to suspect. However, since the algorithm did not really perform as expected with improved results, there is no certainty to that statement. Additionally, to address the mismatch in form with other results, it is suspected that the cause may be in the slight difference in the definition of the error dynamics. As defined in 5.27, the error dynamics used are actually of the *a priori* estimates. Other literature seems to use the *a posteriori* estimates, and as such, the two definitions differ by a Kalman update, which could possibly explain the discrepancy. Time did not permit further investigation into this suspicion, but it must be noted some favourable results did arise with this form in implementation, and the derivation presented is very clear. With regards to the implementation, it is suspected that numerical issues are at fault, as both the filter form derived in this paper and filter forms shown to work by other researchers had similar results on this simulation framework. As well, time did not permit further investigation to solve this problem and obtain working results. However, some hybrid filter improvements were able to be demonstrated.

These consistency concerns represent fundamental issues in the SLAM problem, and should be addressed to achieve the goal of field applications of these technologies. The REKF provides a desired mathematical rigor to dealing with non-linearity and unknown noise characteristics, but it is only one aspect of a much larger framework. These fundamental problems cannot be ignored, and should be addressed, but there is also a lot more work to be done.

References

[1] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the EKF-SLAM Algorithm. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[2] S. J. Julier and J. K. Uhlmann. A Counter Example to the Theory of Simultaneous Localization and Map Building. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4238–4243, Seoul, Korea, 2001.

[3] D. Simon. *Optimal State Estimation*. John Wiley and Sons Ltd, 2006.

[4] R. Smith, M. Self, and P. Cheeseman. Estimation Uncertain Spatial Relationships in Robotics. *Autonomous Robot Vehicles*, pages 167–193, 1990.

[5] J.H. Taylor. The Cramér-Rao Estimation Error Lower Bound Computation for Deterministic Nonlinear Systems. *IEEE Transactions on Automatic Control*, AC–24(2):343–344, 1979.

[6] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[7] M. E. West and V. L. Syrmos. Navigation of an Autonomous Underwater Vehicle (AUV) Using Robust SLAM. In *Proceedings of the 2006 IEEE International Conference on Control Applications*, pages 1801–1806, Munich, Germany, 2006.