Special Problems Fall 2018

Part Three

Planar Robot Simultaneous Localization and Mapping (SLAM) with the Extended Kalman Filter (EKF)

Sanandeesh Kamat *ssk93*

November 29, 2018

Contents

This final paper replicates a subset of algorithms described in *Probabilistic Robotics* by Sebastian Thrun, Wolfram Burgard, and Dieter Fox (**Thrun et al. 2005**). This author is deeply indebted to Dr. Thrun and his colleagues for producing this invaluable exposition into applied Bayesian filtering.

1 Introduction

This paper introduces the Simultaneous Localization and Mapping (SLAM) problem and the Extended Kalman Filter (EKF) SLAM solution for planar robots using feature based maps. The effort was based entirely on Chapters 3, 5, 6, 7, and 10 of Thrun et al. 2005. As background, the reader must be familiar with the EKF, the velocity motion model, and the feature measurment model. A thorough description of these topics was provided in the previous Special Problems paper on EKF-Localization (Kamat, 2018), and will not be repeated here.

Recall that mapping requires knowledge of the robot pose, and localization requires knowledge of the environment map. In situations where the robot possesses neither its own pose nor the environment map, the robot must perform SLAM. Concurrently, the prior map estimate helps localize the robot and the prior pose estimate helps map the landmarks. The mechanics of the SLAM Bayes filter can be conceptualized just like the localization EKF, except the landmark coordinates are now included in the state vector instead of given as input.

There are two main types of SLAM, online SLAM and full SLAM. They are represented as hidden Markov models (HMM) in Figure 1. These HMMs are used to indirectly estimate latent state variables x_t, M based on directly available measurements z_t and motion commands u_t.

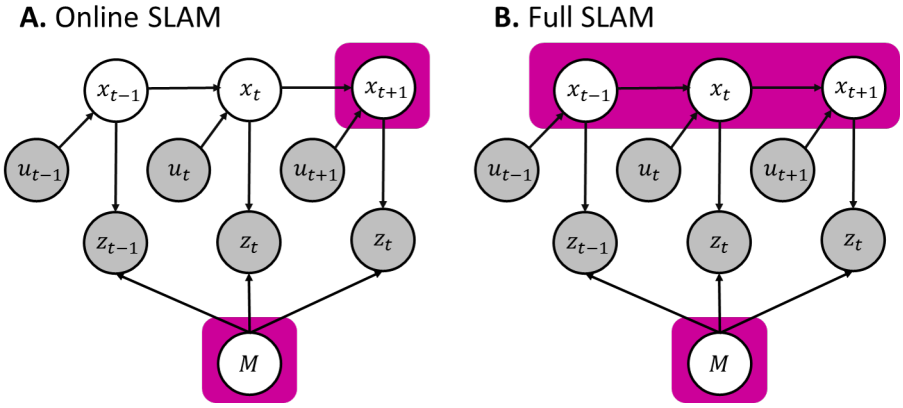

Figure 1: Online and Full SLAM as Hidden Markov Models

Whereas online SLAM includes only the momentary pose x_t as a state vari-

able, full SLAM includes all poses from start to present $x_{0\cdots t}$. In terms of posterior densities, online SLAM produces

$$p(x_t, m|z_{1:t}, u_{1:t}). \tag{1}$$

This is a lower dimensional problem that does not accumulate past measurements and controls. On the other hand, full SLAM produces,

$$p(x_{1:t}, m|z_{1:t}, u_{1:t}). \tag{2}$$

The online SLAM posterior is produced by marginalizing the present pose from the previous poses in the full SLAM posterior,

$$p(x_t, m|z_{1:t}, u_{1:t}) = \int \int \cdots \int p(x_{1:t}, m|z_{1:t}, u_{1:t})\, dx_1\, dx_2 \cdots dx_{t-1}. \tag{3}$$

The full posterior is desirable because it captures all that there is to know about the map and robot trajectory. For example, the residual uncertainty of the full posterior can support prediction and planning. However, when the state vector includes both the pose and map, the full posterior quickly climbs to an unsustainable dimensionality. Therefore, the EKF-SLAM algorithm explored here, only supports online SLAM and can maintain a limited number of landmarks in the state vector. Two important assumptions made are (1) that the robot possesses perfect correspondence between landmarks and measurements, and (2) that the robot observes all landmarks all of the time. The reader will see that these two assumptions significantly bolster EKF-SLAM performance.

2 SLAM with Extended Kalman Filters

EKF-SLAM is of historical value because it was employed in the earliest SLAM implementations. Due to the limited capacity of the state vector, EKF-SLAM is only online and uses feature based maps. EKF SLAM for known correspondences matches EKF-Localization, except the state vector under estimation includes the coordinates of the environmental landmarks as well. This combined state vector is

$$y_t = \begin{pmatrix} x_t \\ m \end{pmatrix} \tag{4}$$

$$= (x_t, y_t, \theta_t, m_{1,x}, m_{1,y}, \cdots, m_{N,x}, m_{N,y})^T \tag{5}$$

For N landmarks, the dimensionality of this expanded state vector is $y_t \in \mathbf{R}^{3+2N}$. The online posterior of y_t is

$$p(y_t | z_{1:t}, u_{1:t}). \tag{6}$$

Kalman filters assume a Gaussian prior distribution over state y_t, and Gaussian noise within linear motion models $g(y_{t-1}, u_t)$ and measurement models $h(y_t)$. However, motion and measurement models used here are both non-linear. Therefore, EKFs linearize these non-linear motion and measurement models to approximately fit them into the Kalman filter paradigm. The Gaussian estimate of the true state vector y_t is represented with the mean vector μ_t and covariance matrix Σ_t. These are initialized as

$$\mu_0 = (0, 0, 0, \cdots, 0)^T, \tag{7}$$

and,

$$\Sigma_0 = \begin{pmatrix} 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \infty & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \infty \end{pmatrix}. \tag{8}$$

The initial position is taken to be the origin of the global coordinate system.

The true state vector evolves according to the non-deterministic motion model,

$$y_t = y_{t-1} + \begin{pmatrix} -\frac{v_t}{w_t}\sin(\theta) + \frac{v_t}{w_t}\sin(\theta + \hat{w}_t \Delta t) \\ \frac{v_t}{w_t}\cos(\theta) - \frac{v_t}{w_t}\cos(\theta + \hat{w}_t \Delta t) \\ \hat{w}_t \Delta t + \hat{\gamma}_t \Delta t \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{9}$$

where $\hat{v}_t, \hat{w}_t, \hat{\gamma}_t$ are the noisy motion commands. Here, only the pose state variables change and the map state variables remain static. This transformation is expressed more compactly as

$$y_t = y_{t-1} + F_x^T \begin{pmatrix} -\frac{\hat{v}_t}{\hat{w}_t}\sin(\theta) + \frac{\hat{v}_t}{\hat{w}_t}\sin(\theta + \hat{w}_t\Delta t) \\ \frac{\hat{v}_t}{\hat{w}_t}\cos(\theta) - \frac{\hat{v}_t}{\hat{w}_t}\cos(\theta + \hat{w}_t\Delta t) \\ \hat{w}_t\Delta t + \hat{\gamma}_t\Delta t \end{pmatrix}, \tag{10}$$

where F_x is a sparse matrix used to map a vector from $\mathbf{R}^3$ to $\mathbf{R}^{3+2N}$.

$$F_x = \begin{pmatrix} 1 & 0 & 0 & 0\cdots 0 \\ 0 & 1 & 0 & 0\cdots 0 \\ 0 & 0 & 1 & 0\cdots 0 \end{pmatrix}. \tag{11}$$

Table 1 describes the EKF-SLAM algorithm. The inputs are the posterior state estimates from the previous time step, $\mu_{t-1}, \mathbf{\Sigma}_{t-1}$, the current measurements $\mathbf{z}_t$, and current motion commands, $\mathbf{u}_t$. Unlike for EKF-Localization, the map $\mathbf{m}$ is no longer available as input and now belongs to the state vector. The Prediction Step only alters the mean and covariance values involving robot pose, and leaves those of the map (including pose-map covariances) unchanged. The predicted mean vector $\overline{\mu}$ evolves according to the deterministic motion model (zero error),

$$\overline{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v}{w}\sin(\theta) + \frac{v}{w}\sin(\theta + w\Delta t) \\ \frac{v}{w}\cos(\theta) - \frac{v}{w}\cos(\theta + w\Delta t) \\ w\Delta t \end{pmatrix} \tag{12}$$

The predicted covariance matrix $\overline{\mathbf{\Sigma}}_t$ evolves according to the propogation of uncertainty of the pose $\mathbf{\Sigma}_{t-1}$ plus uncertainty of the motion commands $\mathbf{M}_t$.

$$\overline{\mathbf{\Sigma}}_t = \mathbf{G}_t\mathbf{\Sigma}_{t-1}\mathbf{G}_t^T + F_x^T\mathbf{V}_t\mathbf{M}_t\mathbf{V}_t^T F_x. \tag{13}$$

Overall, the computation of $\overline{\mathbf{\Sigma}}_t$ mirrors that of EKF-Localization, except that F_x is used to map the update of 3×3 matrix to $(3 + 2N \times 3 + 2N)$.

The full Jacobian of the motion model with respect to *robot pose* is

$$\mathbf{G}_t = I + F_x^T g_t F_x, \tag{14}$$

and is used to propogate $\mathbf{\Sigma}_{t-1}$ across time. The $(3+2N\times 3+2N)$ identity matrix I ensures that the map-related elements of $\mathbf{\Sigma}_{t-1}$ propogate through unchanged. The inner 3×3 matrix,

$$g_t = \begin{pmatrix} 0 & 0 & -\frac{v_t}{w_t}\cos(\theta) + \frac{v_t}{w_t}\cos(\theta + w_t\Delta t) \\ 0 & 0 & -\frac{v_t}{w_t}\sin(\theta) + \frac{v_t}{w_t}\sin(\theta + w_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix}, \tag{15}$$

contains the actual partial dereivities of the motion model. The full Jacobian

of the motion model with respect to *motion command* is

$$F_x^T \mathbf{V}_t = F_x^T \begin{pmatrix} -\frac{\sin(\theta)+\sin(\theta+w_t\Delta t)}{w_t} & \frac{v_t(\sin(\theta)-\sin(\theta+w_t\Delta t))}{w_t^2} + \frac{v_t\cos(\theta+w_t\Delta t)\Delta t}{w_t} \\ \frac{\cos(\theta)-\cos(\theta+w_t\Delta t)}{w_t} & -\frac{v_t(\cos(\theta)-\cos(\theta+w_t\Delta t))}{w_t^2} + \frac{v_t\sin(\theta+w_t\Delta t)\Delta t}{w_t} \\ 0 & \Delta t \end{pmatrix}, \tag{16}$$

and is used to propogate motion command covariance $\mathbf{M}_t$ from command space to pose space. The map-related elements of $F_x^T \mathbf{V}_t \mathbf{M}_t \mathbf{V}_t^T F_x$ are all zero.

Measurements are produced according to the non-deterministic measurement model,

$$z_t^i = \begin{pmatrix} \sqrt{(m_{j,x}-x)^2 + (m_{j,y}-y)^2)} \\ atan2(m_{j,y}-y, m_{j,x}-x) \end{pmatrix} + \mathbf{N}(0, \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{pmatrix}). \tag{17}$$

Here, the observed landmark range and bearing are perturbed by zero mean Gaussian noise.

The Measurement Update Step integrates measurements to correct pose estimates, as well as map estimates. Landmarks which have never been seen before are initialized according to the inverse of the measurement model,

$$\begin{pmatrix} m_{j,x} \\ m_{j,y} \end{pmatrix} = \begin{pmatrix} \overline{\mu}_{t,x} \\ \overline{\mu}_{t,y} \end{pmatrix} + \begin{pmatrix} r_t^i \cos(\phi_t^i + \overline{\mu}_{t,\theta}) \\ r_t^i \sin(\phi_t^i + \overline{\mu}_{t,\theta}) \end{pmatrix}. \tag{18}$$

Hence, the error in the initial landmark coordinate comes from both sensor error as well as pose error. Note that this direct (i.e. bijective) method of initializing landmark coordinates is unique only to active sensing systems (e.g. Radar,Lidar). Passive sensing systems (e.g. cameras) do not enjoy such direct inverse solutions for observed landmark coordinates. They require, instead, the integration of multiple observations with multi-view geometric models to produce landmark estimates.

Just as in EKF-Localization, the EKF-SLAM Measurement Update Step also iterates over each observed landmark and...

- Predicts what the measurement should look like given $\overline{\mu}_t$.

$$\hat{z}_t^i = \begin{pmatrix} \sqrt{(m_{j,x}-\overline{\mu}_{t,x})^2 + (m_{j,y}-\overline{\mu}_{t,y})^2)} \\ atan2(m_{j,y}-\overline{\mu}_{t,y}, m_{j,x}-\overline{\mu}_{t,x}) \end{pmatrix} \tag{19}$$

- Updates the posterior density according to the innovation factor $(z_t^i - \hat{z}_t^i)$, measurement model Jacobian $\mathbf{H}_t^i$, the Kalman Gain $\mathbf{K}_t^i$.

$$\overline{\mu}_t = \overline{\mu}_t + \mathbf{K}_t^i(z_t^i - \hat{z}_t^i) \tag{20}$$

$$\overline{\mathbf{\Sigma}}_t = (I - \mathbf{K}_t^i \mathbf{H}_t^i)\overline{\mathbf{\Sigma}}_t \tag{21}$$

The key differences for EKF-SLAM are in how the measurement model Jacobian $\mathbf{H}_t^i$, the Kalman Gain $\mathbf{K}_t^i$ are structured. The Jacobian $\mathbf{H}_t^i$ is the gradient

of the measurement model with respect the the full state vector y_t. However, in each iteration of the Measurement Update, the measurement model only uses the robot pose and the ith landmark $(x_t, y_t, \theta_t, m_{i,x}, m_{i,y})$. This low dimensional *inner* Jacobian is

$$h_t^i = \frac{1}{q} \begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & +\sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_x & +\delta_x \end{pmatrix}, \tag{22}$$

which augments the previous EKF-Localization Jacobian with two columns for the ith landmark. The inner (2×5) Jacobian h_t^i is mapped to the full $(2\times3+2N)$ dimensional matrix,

$$\mathbf{H}_t^i = h_t^i F_{x,i}, \tag{23}$$

with a sparse $(5 \times 3 + 2N)$ matrix

$$F_{x,i} = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 0 & 1 & 0 \cdots 0 \end{pmatrix}. \tag{24}$$

On each iteration, the diagonal 1s on the bottom two rows of $F_{x,i}$ slide across the matrix to place the ith landmark gradients of h_t^i in the appropriate location of the full $(2 \times 3 + 2N)$ Jacobian $\mathbf{H}_t^i$. Next, the Kalman Gain is computed as

$$\mathbf{K}_t^i = \overline{\mathbf{\Sigma}}_t [\mathbf{H}_t^i]^T [\mathbf{H}_t^i \overline{\mathbf{\Sigma}}_t [\mathbf{H}_t^i]^T + \mathbf{Q}_t]^{-1} \tag{25}$$

The Kalman Gain $\mathbf{K}_t^i$ is the ratio of (1) the cross-covariance between the observation z_t^i and full state vector y_t, $\overline{\mathbf{\Sigma}}_t [\mathbf{H}_t^i]^T$, and (2) the covariance matrix of the observation z_t^i, $[\mathbf{H}_t^i \overline{\mathbf{\Sigma}}_t [\mathbf{H}_t^i]^T + \mathbf{Q}_t]^{-1}$. The resulting $(3 + 2N \times 2)$ Kalman Gain $\mathbf{K}_t^i$ *folds* the innovation factor $(z_t^i - \hat{z}_t^i)$ into the updated state estimate μ_t. By inspection, one can see that $\mathbf{K}_t^i$ is not likely to be sparse, and is populated for state variables *beyond* only the robot pose and ith landmark coordinates. This fact is crucial for why EKF-SLAM does not need to explicitly maintain state variable of the past (i.e. online SLAM). Landmarks seen presently can impact the estimate of landmarks not currently visible because relationships *between* state variables are represented by the off-diagonal elements of posterior covariance of state Σ_t.

	EKF_SLAM$(\mu_{t-1}, \Sigma_{t-1}, \mathbf{u}_t, \mathbf{z}_t)$
	Prediction Step Section 4.1
1	$F_x = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 \end{pmatrix}$
	Jacobians of Linearized Motion Model $\mathbf{G}_t, \mathbf{V}_t$
2	$\mathbf{G}_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{v_t}{w_t}\cos(\theta) + \frac{v_t}{w_t}\cos(\theta + w_t\Delta t) \\ 0 & 0 & -\frac{v_t}{w_t}\sin(\theta) + \frac{v_t}{w_t}\sin(\theta + w_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$
3	$\mathbf{V}_t = \begin{pmatrix} -\frac{\sin(\theta)+\sin(\theta+w_t\Delta t)}{w_t} & \frac{v_t(\sin(\theta)-\sin(\theta+w_t\Delta t))}{w_t^2} + \frac{v_t\cos(\theta+w_t\Delta t)\Delta t}{w_t} \\ \frac{\cos(\theta)-\cos(\theta+w_t\Delta t)}{w_t} & -\frac{v_t(\cos(\theta)-\cos(\theta+w_t\Delta t))}{w_t^2} + \frac{v_t\sin(\theta+w_t\Delta t)\Delta t}{w_t} \\ 0 & \Delta t \end{pmatrix}$
	Motion Noise Covariance Matrix $\mathbf{M}_t$
4	$\mathbf{M}_t = \begin{pmatrix} \alpha_1 v_t^2 + \alpha_2 w_t^2 & 0 \\ 0 & \alpha_1 v_t^2 + \alpha_2 w_t^2 \end{pmatrix}$
	Prediction Update from Motion Model $\overline{\mu}_t, \overline{\mathbf{\Sigma}}_t$
5	$\overline{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v}{w}\sin(\theta) + \frac{v}{w}\sin(\theta + w\Delta t) \\ \frac{v}{w}\cos(\theta) - \frac{v}{w}\cos(\theta + w\Delta t) \\ w\Delta t \end{pmatrix}$
6	$\overline{\mathbf{\Sigma}}_t = \mathbf{G}_t\mathbf{\Sigma}_{t-1}\mathbf{G}_t^T + F_x^T\mathbf{V}_t\mathbf{M}_t\mathbf{V}_t^T F_x$
	Measurement Step Section 4.2
	Measurement Noise Covariance Matrix $\mathbf{Q}_t$
7	$\mathbf{Q}_t = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{pmatrix}$
	for $i = (1 \cdots N)$
	if (m_j Never Seen Before)
8	$\begin{pmatrix} m_{j,x} \\ m_{j,y} \end{pmatrix} = \begin{pmatrix} \overline{\mu}_{t,x} \\ \overline{\mu}_{t,y} \end{pmatrix} + \begin{pmatrix} r_t^i\cos(\phi_t^i + \overline{\mu}_{t,\theta}) \\ r_t^i\sin(\phi_t^i + \overline{\mu}_{t,\theta}) \end{pmatrix}$
	endif
	Predicted Measurement of ith Landmark $\hat{z}_t^i$
9	$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} m_{j,x} - \overline{\mu}_{t,x} \\ m_{j,y} - \overline{\mu}_{t,y} \end{pmatrix}$
10	$q = \delta^T\delta$
11	$\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ atan2(\delta_y, \delta_x) - \overline{\mu}_{t,\theta} \end{pmatrix}$
	Jacobians of Linearized Measurement Model $\mathbf{H}_t^i$
12	$F_{x,j} = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 0 & 1 & 0 \cdots 0 \end{pmatrix}$
13	$\mathbf{H}_t^i = \frac{1}{q}\begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & +\sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_x & +\delta_x \end{pmatrix} F_{x,j}$
	Kalman Gain $\mathbf{K}_t^i$
14	$\mathbf{K}_t^i = \overline{\mathbf{\Sigma}}_t[\mathbf{H}_t^i]^T[\mathbf{H}_t^i\overline{\mathbf{\Sigma}}_t[\mathbf{H}_t^i]^T + \mathbf{Q}_t]^{-1}$
	Measurement Update from Measurement Model (LMMSE) $\mu_t, \mathbf{\Sigma}_t$
15	$\overline{\mu}_t = \overline{\mu}_t + \mathbf{K}_t^i(z_t^i - \hat{z}_t^i)$
16	$\overline{\mathbf{\Sigma}}_t = (I - \mathbf{K}_t^i\mathbf{H}_t^i)\overline{\mathbf{\Sigma}}_t$
	endfor
17	$\mu_t = \overline{\mu_t}$
18	$\mathbf{\Sigma}_t = \overline{\mathbf{\Sigma}_t}$
19	return $\mu_t, \mathbf{\Sigma}_t$

Table 1: The EKF SLAM Algorithm with Known Landmark Correspondences

The evolution of the covariance matrix elements is an essential aspect of how the EKF handles the SLAM problem. In particular, the variance (diagonal) elements of the landmarks are intended to *decrease* and the covariance between landmarks and robot pose are expected to *increase*. Figure 2 illustrats a sequence of covariance matrices from a sample EKF-SLAM (described in Section 3.2). The checkerboard effect on the images relates to how alternating elements down the diagonal represent either x or y coordinates of each landmark.

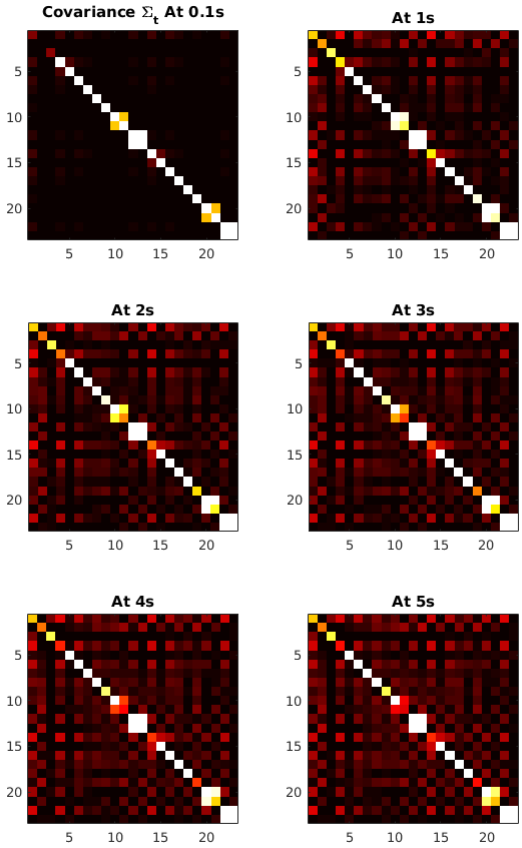

Figure 2: Sequence of Posterior Covariance Matrices during EKF SLAM

3 Simulation Results

3.1 Scenario Design

The goal of these simulations is to explore how various parameters influence the localization & mapping (SLAM) performance. These parameters include the motion error $\alpha_1 \cdots \alpha 4$, measurement error $\sigma_r^2, \sigma_\phi^2$, and map configuration parameters. Therefore, the simulations were broken into three categories.

- Simulations which modulate the *motion error parameters* $\alpha_1 \cdots \alpha 4$, and leave the remaining parameters at default.
- Simulations which modulate the *measurement error parameters* $\sigma_r^2, \sigma_\phi^2$, and leave the remaining parameters at default.
- Simulations which modulate the *map configuration* M, and leave the remaining parameters at default.

Table 2 describes all of the parameters represented in the simulations, including those which were not probed for analysis (no range of values).

	Default	**Range**
Time Step	$Ts = 0.1s$	
Motion Commands	$v = 2\text{m/s}$ $w = 0.2\text{rad/s}$	
Map Configuration	$N = 10$ $R = 50\text{m}$	$N = 3 \cdots 28$ $R = 5 \cdots 100$
Motion Error	$\alpha_{1,2} = 0.5$ $\alpha_{3,4} = 0.5$ $\alpha_{5,6} = 0.5$	$\alpha_{1,2} = 0.1 \cdots 5.0$ $\alpha_{3,4} = 0.1 \cdots 5.0$
Measurement Error	$\sigma_R^2 = 0.5$ $\sigma_\phi^2 = 0.05$	$\sigma_R^2 = 0.1 \cdots 5.0$ $\sigma_\phi^2 = 0.01 \cdots 1.0$

Table 2: Selected Simulation Parameters

As before, the environment consisted of N landmarks uniformly spaced along the perimeter of a circle of radius R centered on the origin. Figure 3 illustrates three example maps of different N, R configurations.

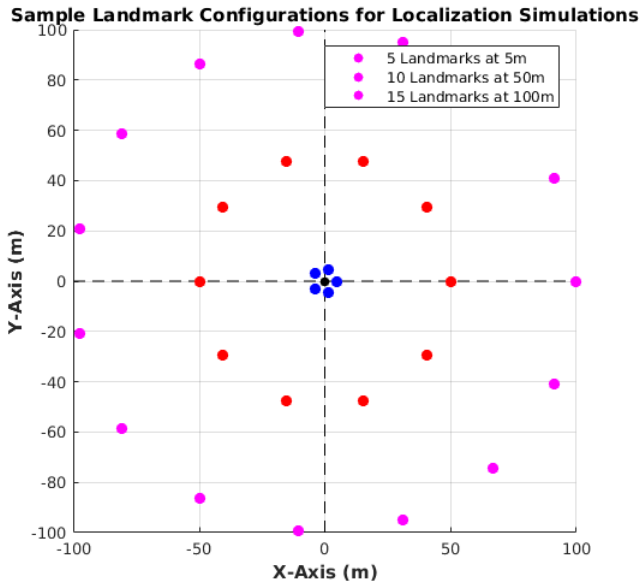

Figure 3: Sample Landmark Configurations for Localization Simulations

The performance of the localization and mapping was measured according to the position error

$$\epsilon_{x,y} = \sqrt{(\hat{x} - x)^2 + (\hat{y} - y)^2} \tag{26}$$

and angular error

$$\epsilon_\theta = \left| \hat{\theta} - \theta \right|. \tag{27}$$

3.2 Performance at Default Parameters

Figure 4 provides the robot localization results for 10 simulations run at the default parameters (Table 2). Figure 4A shows the true paths (black dashed) and the estimated paths (green solid) taken by the robot. Figure 4B and 3C plot the position and angular localization error, respectively, versus time. These outputs indicate a modest capability of localization, with $\mu_{PosErr} = 0.9m$ and $\mu_{\theta Err} = 7.7deg$. Most trials produced errors flattening over time, and a few exhibited growing errors.

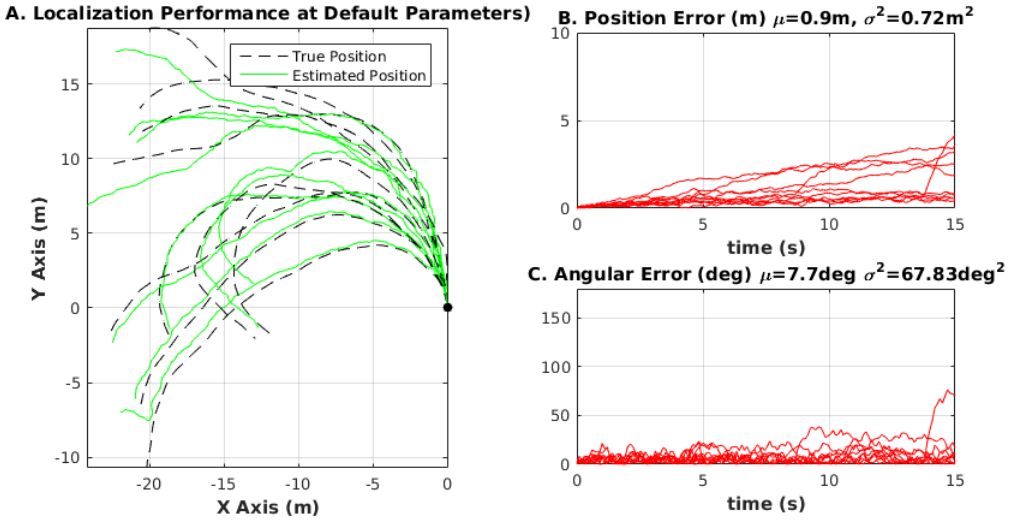

Figure 4: EKF SLAM Localization Results at Default Parameters (Table 2)

Figure 5 provides the corresponding mapping estimates for the same 10 simulations. Figure 5A shows the true landmark positions (red circles) and the estimated landmark positions (blue lines) for each simulation. Although the landmarks are stationary throughout the simulations, landmark estimates are expected to shift over time. Most of the landmark estimates remain close to their true values with $\mu_{PosErr} = 0.2m$.

However, the two leftmost landmark estimates are wildly inaccurate! These errors consistently appeared across repeated simulations. Although the reasons are still inconclusive, it is suspected that these mapping errors are due to the landmarks being close to π radians (in polar coordinates). Measurements of landmarks in the π-region perturbed by Gaussian noise will randomly shift over and under the π line, which may be causing undesirable estimation errors (e.g. 4-quadrant $\tan^{-1}()$). In seperate results (not shown here) it was shown that changing the initial pose of the robot had no effect on the problematic π-region landmarks. Moreover, simulations for which any landmarks within the $\pi \pm 0.5$ radian neighbourhood were removed did not exhibit any mapping errors (and

hence enjoyed superior SLAM performance). However, the results in this paper maintain all landmarks evenly spaced from $0 \cdots 2\pi$ radians and so do contain the mapping errors illicited by landmarks in the π-region. Figure 5B plots the *mean* landmark position error for each trial across time. Hence, the outlier errors of the π-region landmarks in Figure 5A are averaged away by the other succesfully mapped landmarks Figure 5B.

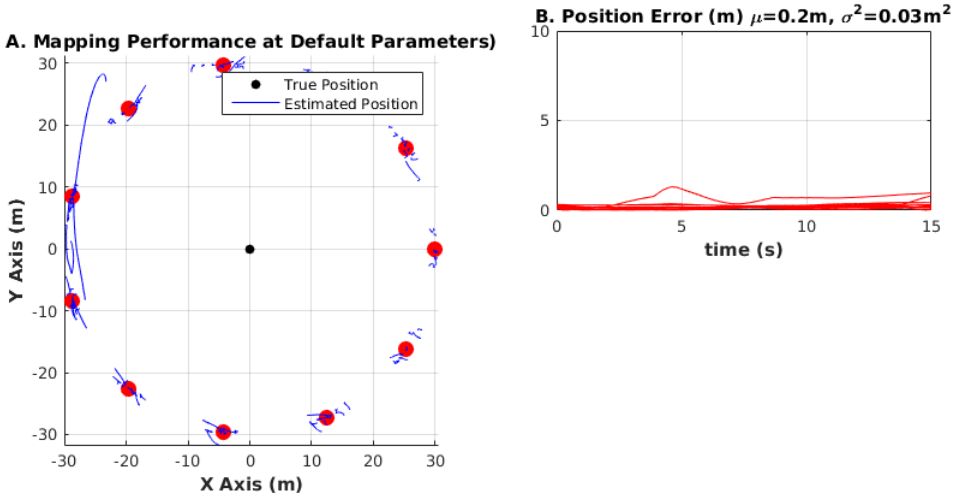

Figure 5: Mapping Results at Default Parameters (Table 5)

The full $(3+2N \times 3+2N)$ covariance matrices Σ_t were averaged across trials and analyzed across time. Three types of sub-covariance matrices were of interest. Their determinants were calculated signify overall covariance magnitude and are shown in Figure 6.

- Covariance between robot position and landmark positions.

$$\Sigma_{RobLM} = \begin{pmatrix} \sigma_{Rob_X}\sigma_{LM_X} & \sigma_{Rob_X}\sigma_{LM_Y} \\ \sigma_{Rob_Y}\sigma_{LM_X} & \sigma_{Rob_Y}\sigma_{LM_Y} \end{pmatrix} \tag{28}$$

  Figure 6A shows the determinants of these matrices for each landmark. As expected, they began low and increased over time.

- Covariance within each landmark position.

$$\Sigma_{LM} = \begin{pmatrix} \sigma_{LM_X^2} & \sigma_{LM_X}\sigma_{LM_Y} \\ \sigma_{LM_Y}\sigma_{LM_X} & \sigma^2_{LM_Y} \end{pmatrix} \tag{29}$$

  Figure 6B shows the determinants of these matrices for each landmark. As expected they began high and decreased over time.

- Covariance between landmark positions.

$$\Sigma_{LM_i LM_j} = \begin{pmatrix} \sigma_{LM_{x,i}}\sigma_{LM_{x,j}} & \sigma_{LM_{x,i}}\sigma_{LM_{y,j}} \\ \sigma_{LM_{y,i}}\sigma_{LM_{x,j}} & \sigma_{LM_{y,i}}\sigma_{LM_{y,j}} \end{pmatrix} \tag{30}$$

  Figure 6C shows the determinants of these matrices for each pair of landmarks. It was expected that they would all begin low and increase uniformly (such as in Figure 6B). However, possibly due to the problematic π-region landmarks, many of the covariance determinants went negative.

Other statistics, such as the Frobenius Norm, may also be informative to compute on these three types of covariance matrices.

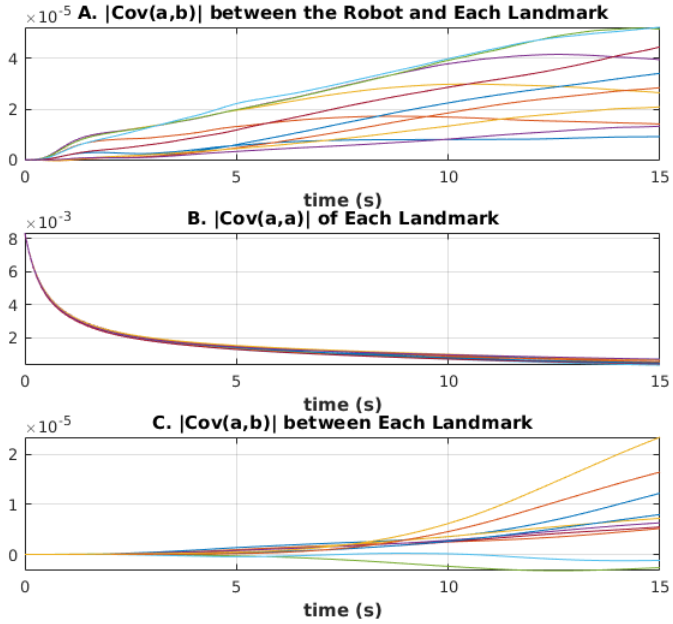

Figure 6: Determinants of Covariance Matrices at Default Parameters (Table 5)

The following three sections provide mean localization and mapping errors over a much larger numbers of trials. The trials focus on different simulation parameters induvidually and yield insight into how sensitive the overall EKF-SLAM performance is to their modification.

3.3 Effects of Motion Error on Performance

This section describes how the overall EKF-SLAM performance was affected by wide modifications to the motion error parameters $\alpha_{1,2,3,4}$. Recall from Thrun et al. 2005 that $\alpha_{1,2}$ and $\alpha_{3,4}$ scale the variances of zero-mean Gaussian errors in tranlational and rotational motion commands, respectively. That is,

$$\begin{pmatrix} \hat{v} \\ \hat{w} \end{pmatrix} = \begin{pmatrix} v \\ w \end{pmatrix} + \begin{pmatrix} \epsilon_{\alpha_1 v^2 + \alpha_2 w^2} \\ \epsilon_{\alpha_3 v^2 + \alpha_4 w^2} \end{pmatrix}, \tag{31}$$

where ϵ_{b^2} is Norm$(0, b^2)$. Figure 7A and 7B show the mean robot position and angular (i.e. localization) error, respectively. Figure 7A shows that translational motion error $\alpha_{1,2}$ and angular motion error $\alpha_{3,4}$ have weak and strong influences on position error, respectively. Surprisingly, Figure 6B shows almost no significant influence of either $\alpha_{1,2}$ or $\alpha_{3,4}$ on angular error.

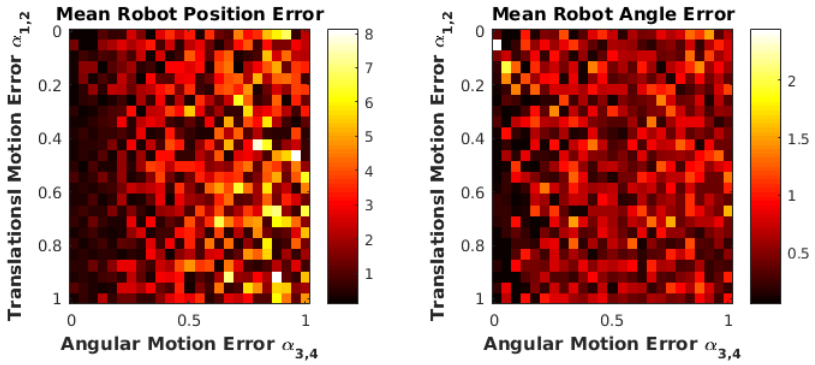

Figure 7: Localization Results at Default Parameters (Table 5)

The Figure 8 shows the mean landmark position (i.e. mapping) error. There is a close correspondece between 7A, and 8. It seems that $\alpha_{1,2}$ and $\alpha_{3,4}$ influence robot position and landmark position error with the same pattern (although with different magnitudes).

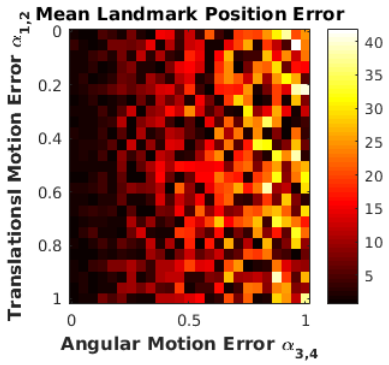

Figure 8: Localization Results at Default Parameters (Table 5)

3.4 Effects of Measurement Error on Performance

This section describes how the overall EKF-SLAM performance was affected by wide modifications to the measurement error parameters σ_r^2 and σ_Φ^2. Recall from Thrun et al. 2005 that σ_r^2 and σ_Φ^2 are the variances of zero-mean Gaussian errors in range and bearing measurements, respectively. That is,

$$\begin{pmatrix} z_{t,r}^i \\ z_{t,\Phi}^i \end{pmatrix} = \begin{pmatrix} z_{t,r}^i \\ z_{t,\Phi}^i \end{pmatrix} + \begin{pmatrix} \epsilon_{\sigma_r^2} \\ \epsilon_{\sigma_\Phi^2} \end{pmatrix}, \tag{32}$$

where ϵ_{b^2} is Norm$(0, b^2)$. Figure 9A and 9B show the mean robot position and angular error, respectively. Figure 9A shows that σ_r^2 and σ_Φ^2 have strong and weak influences on position error, respectively. Similar to Figure 6B, Figure 8B shows almost no significant influence of either σ_r^2 or σ_Φ^2 on angular error.

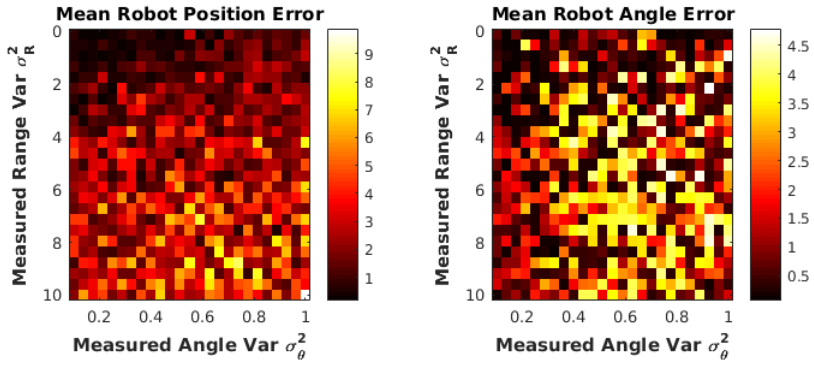

Figure 9: Localization Results at Default Parameters (Table 5)

The Figure 10 shows the mean landmark position error. It shows that σ_r^2 and σ_Φ^2 influence landmark position error weakly and strongly, respectively.

Unlike for the Section 3.3, comparing Figure 9A and 10, patterns of influence for landmark and robot position errors do not match.

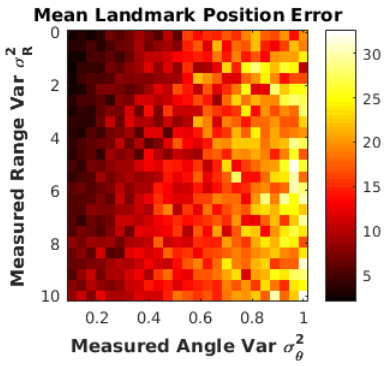

Figure 10: Localization Results at Default Parameters (Table 5)

3.5 Effects of Map Configuration on Performance

This section describes how the overall EKF-SLAM performance was affected by wide modifications to the number of landmarks N and their radial range to the origin R. Figure 3 illustrates examples of (N, R) map configurations. Figures 11A and 11B show the mean robot position and angular error, respectively. Surprisingly, both Figures 11A and 11B show weak influence of landmark N and R on localization error!

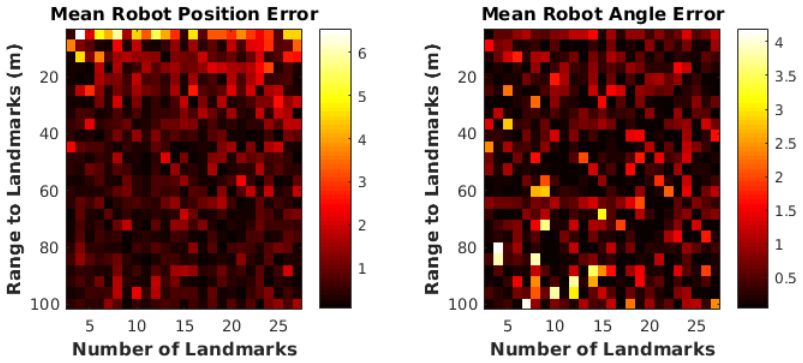

Figure 11: Localization Results at Default Parameters (Table 5)

The Figure 12 shows the mean landmark position error. Unlike Figures 11A and 11B, Figure 12 does show a modest influence of landmark R to landmark position error. Unexpectedly, landmark N shows no influence here.

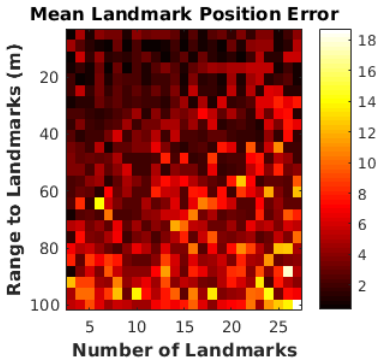

Figure 12: Localization Results at Default Parameters (Table 5)

It appears that wide modifications to map configuration primarily affect mapping performance; not localization performance.

4 Conclusion

This paper introduced the Simultaneous Localization and Mapping (SLAM) problem and the Extended Kalman Filter (EKF) SLAM solution for planar robots using feature based maps. The effort was based entirely on Chapters 3, 5, 6, 7, and 10 of Thrun et al. 2005. The educational purpose was to implement an EKF-SLAM simulation capability with Matlab software, and to comprehensively probe the results. These results included...

- Section 3.2 *Close Up*

  A few raw localization and mapping estimates for indivudual trials at default simulation parameters (Table 2). Evolution of sub-covariance matrices between and within landmarks and robot pose.

- Sections 3.3 to 3.5 *High Level*

  The influence of motion error $\alpha_{1\ldots4}$, measurement error $\sigma_{r,\Phi}$, and map configuration (N, R) on the mean localization and mapping errors across a large number of trials.

The most significant problem that arose was the reproducibly high error for landmarks in the π-region of the map (in polar coordinates). More time is required to investigate the causes of this (likely mathematical) software bug.

In spite of this π-region landmark issue, the overall results shown in this paper exhibit modestly accurate localization and mapping estimates. This is because of two *wildly unrealistic* assumptions made in this paper. Firstly, perfect landmark correspondence was provided to the robot. Landmark ambiguity represents a significant challenge for EKF-SLAM because signature distinctiveness is rarely naturally available. Secondly, all landmarks were observable at

all times. Due to physical obstructions and sensor variability, this is rarely the case in practice. These two assumptions enabled all of the landmarks to have relatively accurate initial coordinates (because μ_0 begins at the origin) on the very first iteration which helped them converge to their true coordinates quickly. In reality, landmarks discovered at any $t > 0$ will be initialized with respect to the latest, noisier estimate of pose.

Therefore, the three most important points to improve about this paper are to

- Investigate and resolve the π-region landmark issue.
- Introduce landmark ambiguity. Implement ML correspondence estimator as well as map management techniques.
- Reduce the robot field of view of to restrict the number of landmarks visible at any time.

Further topics to explore in *Probabilistic Robotics* include higher level tasks such as path planning and prediction.

5 References

1. Kamat. (2018). *Planar Robot Localization with the Extended Kalman Filter (EKF).* An unpublished school report. Available from

   `https://drive.google.com/file/d/1_xD-lSe36zxA_PwHaFMd2DyPvIGQStg5/view`

2. Mathworks. (2011). *Statistics and Machine Learning Toolbox (r2018a).* Retrieved April 15, 2018 from https://www.mathworks.com/help/stats/

3. Sarkka S (2013). *Bayesian Filtering and Smoothing.* Cambridge University Press.

4. Thrun S, Burgard W, Fox D. (2005). *Probabilistic Robotics.* Intelligent Robotics and Autonomous Agents. MIT Press.

6 Appendix: EKF-SLAM Matlab Code

```
function [muCurr, covCurr] = EKFSLAM(muPrev, covPrev, v, w, obs, alpha, varObs)
% Initialize Variables
Ts = 0.1; % Time Step (s)
theta = muPrev(3); % Robot Azimuth
% ========================= Prediction Step ========================
dimState = length(covPrev);
Fx = zeros(3, dimState);
Fx(1:3,1:3) = eye(3);
% Jacobian for Linearized Motion Model, Evaluated at u_t and mu_t-1
Gt = eye(dimState) + Fx'*[0, 0, -(v/w)*cos(theta)+(v/w)*cos(theta+w*Ts);
                         0, 0, -(v/w)*sin(theta)+(v/w)*sin(theta+w*Ts);
                         0, 0,                   0                  ]*Fx;
% Jacobian for Linearized Motion Model wrt Motion Parameters, Evaluated at u_t & mu_t-1
Vt = [(-sin(theta)+sin(theta+w*Ts))/w,
      (v*(sin(theta)-sin(theta+w*Ts)))/w^2+(v*cos(theta+w*Ts)*Ts)/w;
      (cos(theta)-cos(theta+w*Ts))/w,
      -(v*(cos(theta)-cos(theta+w*Ts)))/w^2+(v*sin(theta+w*Ts)*Ts)/w;
                   0,
                   Ts];
% Motion Noise Covariance Matrix from the Control
Mt = [alpha(1)*v^2+alpha(2)*w^2,              0;
                  0,              alpha(3)*v^2+alpha(4)*w^2];
% Motion Update
% Expectation of Predicted State
muCurr_Pred = muPrev + Fx'*[-(v/w)*sin(theta)+(v/w)*sin(theta+w*Ts); % Pred x coord
                            (v/w)*cos(theta)-(v/w)*cos(theta+w*Ts); % Pred y coord
                                                    w*Ts];                % Pred Az ang
muCurr_Pred(3) = mod(muCurr_Pred(3)+2*pi, 2*pi);
% Covariance of Predicted State
covCurr_Pred = Gt*covPrev*Gt' + Fx'*Vt*Mt*Vt'*Fx;
% ========================= Measurement Step ========================
% Covariance of Measurement
varRng = varObs(1);
varPhi = varObs(2);
Qt = [varRng, 0,    0;
      0,    varPhi, 0;
      0,      0,    0.01];
numObs = size(obs, 2);
for iObs = 1:numObs
    % Index of this Landmark in the State Vector
    iObsState.x = 3+(2*iObs-1);
    iObsState.y = 3+(2*iObs);
    % If This Landmark has Not Been Observed Yet ({x,y}=0)
    if (muCurr_Pred(iObsState.x)==0 && muCurr_Pred(iObsState.y)==0)
```

```
        % Initialize Location by the projected location obtained from
        % the corresponding range and bearing measurement (i.e. local to global).
        muCurr_Pred(iObsState.x) = muCurr_Pred(1) +
                                   obs(1,iObs)*cos(obs(2,iObs)+muCurr_Pred(3));
        muCurr_Pred(iObsState.y) = muCurr_Pred(2) +
                                   obs(1,iObs)*sin(obs(2,iObs)+muCurr_Pred(3));
    end
    deltaX = muCurr_Pred(iObsState.x)-muCurr_Pred(1);
    deltaY = muCurr_Pred(iObsState.y)-muCurr_Pred(2);
    % Predicted Local Observation of Landmark
    rng = sqrt((deltaX)^2+(deltaY)^2);
    predObs = [rng;                                     % Range
               atan2(deltaY, deltaX)-muCurr_Pred(3); % Bearing
               1];                                      % Signature
    % The Jacobian of the Meas-Model wrt. robot pose and landmark coord
    ht = (1/(rng^2)).*[-rng*deltaX, -rng*deltaY,    0, rng*deltaX, rng*deltaY,     0;
                            deltaY,     -deltaX, -rng,    -deltaY,     deltaX,     0;
                                 0,           0,    0,          0,          0, rng^2];
    % Transform Low Dim Jacobian to Full State Dim
    Fxj = GenerateFxj(iObs, dimState);
    Ht = ht*Fxj; % [3 x dimState]
    St = Ht*covCurr_Pred*Ht'+Qt;     % Covariance of the Observation [3x3]
    Kt = covCurr_Pred*Ht'*(St^(-1)); % Kalman Gain (Cross Covariance over Observation Covar)
%         Kt = covCurr_Pred*Ht'/(St);
    % Linear Min Mean Square Estimator (LMMSE) of Gaussian State
    muCurr_Pred = muCurr_Pred + Kt*(obs(:,iObs)-predObs); % Expected State
    muCurr_Pred(3) = mod(muCurr_Pred(3)+2*pi, 2*pi);
    covCurr_Pred = (eye(dimState)-Kt*Ht)*covCurr_Pred;         % Covariance of State
end
% Set the Final 1st/2nd Moments of the Posterior Estimate
muCurr  = muCurr_Pred;
muCurr(3) = mod(muCurr(3)+2*pi, 2*pi);
covCurr = covCurr_Pred;
return;
end


%% Generate Matrix to Map Low Dim Jacobian to Full State Dim
function [Fxj] = GenerateFxj(iObs, dimState)
    iCol = 3+(2*iObs-1);
    Fxj = zeros(6, dimState); % Maps low-dim Jacobian to full state-dim
    Fxj(1:3,1:3) = eye(3);
    Fxj(4:5,iCol:(iCol+1)) = eye(2);
    return;
end
```