Lecture 10: *Simultaneous Localization and Mapping*

Scribe: *Edward Hu, Shijie Zhou, Christopher Imantaka, R. Michael Swan*

Contents

1 Mapping

1.1 Motivation

In the localization lecture, we've discussed localizing, or figuring out the robot's pose in the environment given a map. Now, we want to solve the complementary problem: how can we acquire a map of the world from scratch? We can start with the simplest of maps, an occupancy grid (Figure 2).

1.2 Formulation

In Figure 1 below we see the causal diagram for mapping in our system. The map is built up based on observations and states z_t and x_t like so:

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

where m is the map. For the time being, we will assume that we know where we are and that inputs do not provide further information. Therefore, inputs u_t are not included in

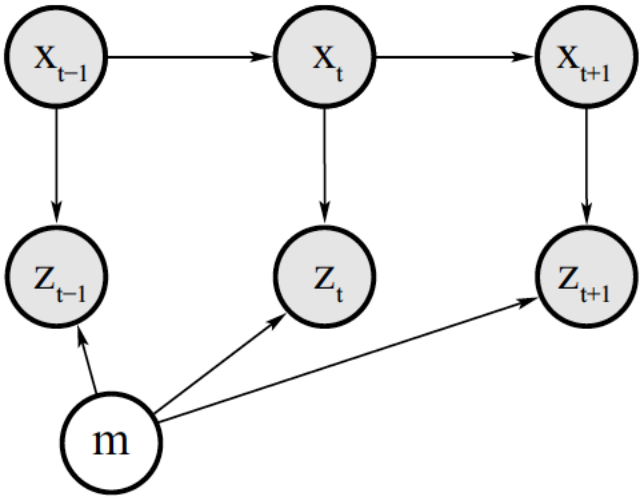

Figure 1: Figure 9.2 from TBF[1]. Graphical model of mapping with known poses. The shaded variables (poses x and measurements z) are known. The goal of mapping is to recover the map m.

the basic formulation. That said, what is m? There are infinitely many possible maps, so how do we break it down? One approach is to use an **occupancy grid** which essentially breaks down the map into an array of cells of some size such that:

$$m = \sum_i m_i \tag{2}$$

Where each cell m_i has a binary occupancy value, "1" for free and "0" for occupied. Occupancy grids are usually used to map 2-D floorplans, though it can generalize to 3-D at significant computational expense. Applying equation (2) to equation (1) to look at some cell m_i we get:

$$p(m_i|z_{1:t}, x_{1:t}) \tag{3}$$

This gives us our initial formulation. We still need to apply the log-odds representation of occupancy to avoid numerical instability. That is:

$$l_{t,i} = log\frac{p(m_i|z_{1:t}, x_{1:t})}{1 - p(m_i|z_{1:t}, x_{1:t})} \tag{4}$$

1.3 Algorithm

The occupancy grid algorithm is fairly simple to understand. Essentially we update the map based on what we can currently see.

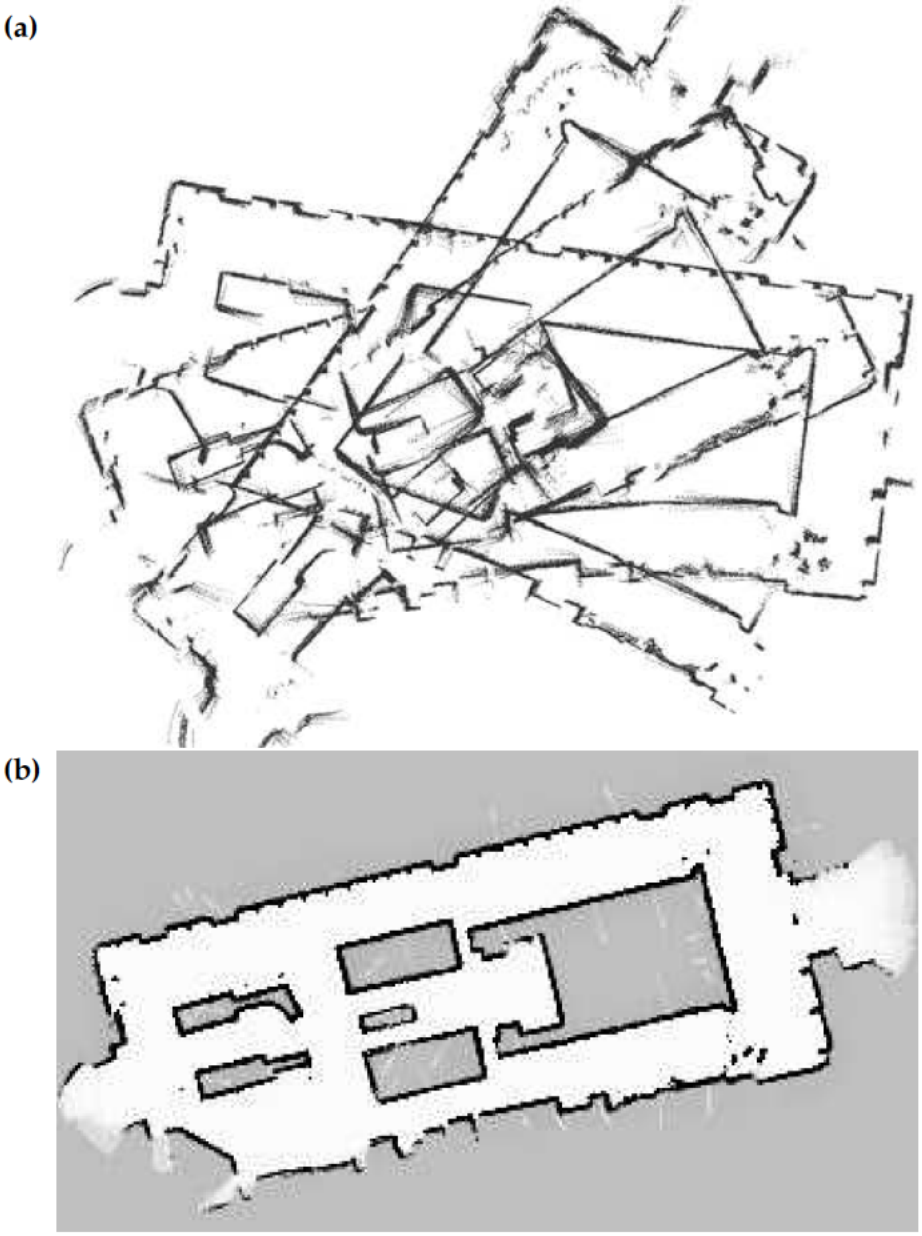

Figure 2: Figure 9.1 from TBF[1]. (a) Raw range data, position indexed by odometry. (b) Occupancy grid map.

1: **Algorithm occupancy_grid_mapping($\{l_{t-1,i}\}, x_t, z_t$):**
2: *for all cells* $\mathbf{m}_i$ *do*
3: *if* $\mathbf{m}_i$ *in perceptual field of* z_t *then*
4: $l_{t,i} = l_{t-1,i} + \textbf{inverse\_sensor\_model}(\mathbf{m}_i, x_t, z_t) - l_0$
5: *else*
6: $l_{t,i} = l_{t-1,i}$
7: *endif*
8: *endfor*
9: *return* $\{l_{t,i}\}$

Figure 3: Table 9.2 from TBF [1]. A simple inverse measurement model for robots equipped with range finders. Here α is the thickness of obstacles, and β the width of a sensor beam. The values locc and lfree in lines 9 and 11 denote the amount of evidence a reading carries for the two different cases

The constant l_0 is the prior of occupancy as represented by a log-odds ratio:

$$l_0 = log\frac{p(m_i = 1)}{p(m_i = 0)} = log\frac{p(m_i}{1 - p(m_i)} \tag{5}$$

Note from the algorithm that we have something called the **inverse sensor model**. That is:

$$inverse\_sensor\_model(m_i, x_t, z_t) = p(m_i|z_t, x_t) \tag{6}$$

which is a marginalized inverse measurement model that reasons from effects to causes. It provides information about this world conditioned on a measurement caused by this world.

1.4 Derivation

Applying Bayes rule and then the Markov assumption to equation (6) (the inverse sensor model) then provides the derivation:

$$\begin{aligned}
p(m_i|z_{1:t}, x_{1:t}) &= p(m_i|z_{1:t-1}, x_{1:t-1}, z_t, x_t) \\
&= \eta p(z_t|m_i, z_{1:t-1}, x_{1:t-1}, x_t) p(m_i|z_{1:t-1}, x_{1:t-1}) \\
&= \eta p(z_t|x_t, m_i) p(m_i|z_{1:t-1}, x_{1:t-1}) \\
p(z_t|x_t, m_i) &= \frac{p(m_i|z_t, x_t) p(z_t|x_t)}{p(m_i|x_t)} \\
p(m_i|z_{1:t}, x_{1:t}) &= \frac{\eta p(m_i|z_t, x_t) p(z_t|x_t)}{p(m_i)} \cdot p(m_i|x_{1:t-1}, z_{1:t-1})
\end{aligned}$$

Note that we assume above that having just a state does not give us any info about the map, while having an observation and a state does. Therefore $p(m_i|x_t)$ simplifies to $p(m_i)$ in the derivation above. We are able to simplify further using the ratio $\frac{p(m_i=0)}{p(m_i=1)}$ as seen below:

$$\frac{p(m_i = 1|z_{1:t}, x_{1:t})}{p(m_i = 0|z_t, x_{1:t})} = \frac{p(m_i = 1|z_t, x_t)}{p(m_i = 0|z_t, x_t)} \cdot \frac{p(m_i = 0)}{p(m_i = 1)} \cdot \frac{p(m_i = 1|z_{1:t-1}, x_{1:t-1})}{p(m_i = 0|z_t, x_{1:t-1})}$$

Then taking the log odds of that which gives us $l_t = inverse\_sensor\_model + l_0 + l_{t-1}$ (in order) as seen here:

$$log\frac{p(m_i = 1|z_{1:t}, x_{1:t})}{p(m_i = 0|z_t, x_{1:t})} = log\frac{p(m_i = 1|z_t, x_t)}{p(m_i = 0|z_t, x_t)} + log\frac{p(m_i = 0)}{p(m_i = 1)} + log\frac{p(m_i = 1|z_{1:t-1}, x_{1:t-1})}{p(m_i = 0|z_t, x_{1:t-1})}$$

2 Simultaneous Localization and Mapping

2.1 Motivation

How can a robot, when dropped into an unknown environment, both localize itself and build a map at the same time? SLAM combines the concepts of localization and mapping to jointly estimate both at the same time.

2.2 Formulation

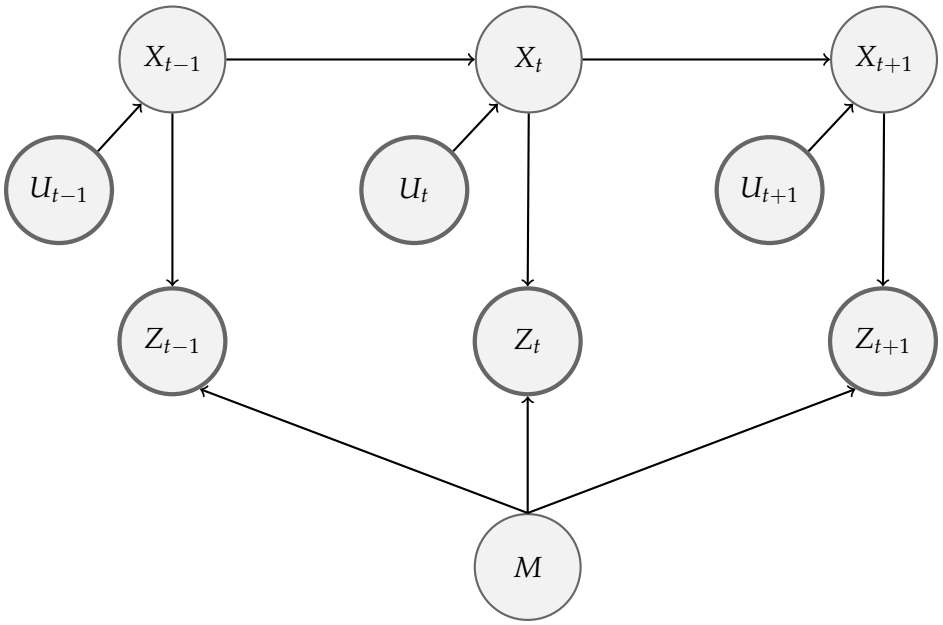

Figure 4: Graphical model of simultaneous Localization and Mapping. The thick variables (control u and measurements z) are known. The thin variables (poses x and map m) are unknown. The goal of mapping is to recover the map m and the poses x.

In the figure below we see the case when both x and m are unknown however this algorithm we are assuming that the correspondences are known. Here we can denote the state vector which comprise robot pose and the map as μ_t. It is given by

$$\begin{aligned}\mu &= \begin{bmatrix} X_t & m \end{bmatrix}^T \\ &= \begin{bmatrix} x & y & \theta & m_{1x} & m_{1y} & S_1 & \cdots & m_{Nx} & m_{Ny} & S_N \end{bmatrix}^T\end{aligned}$$

x, y, θ represent the robot's coordinates at time t. m_{ix}, m_{iy} is the coordinates of the i-th landmark with s_i to be its signature, for $i = 1, 2, \cdots, N$. Since there are N landmarks, the dimension of the state vector is $3 + 3N$.

EKF SLAM calculates the online posterior

$$p(\mu_t | z_{1:t}, u_{1:t}) \tag{7}$$

using the algorithm introduced in the next part, which is Figure 6.

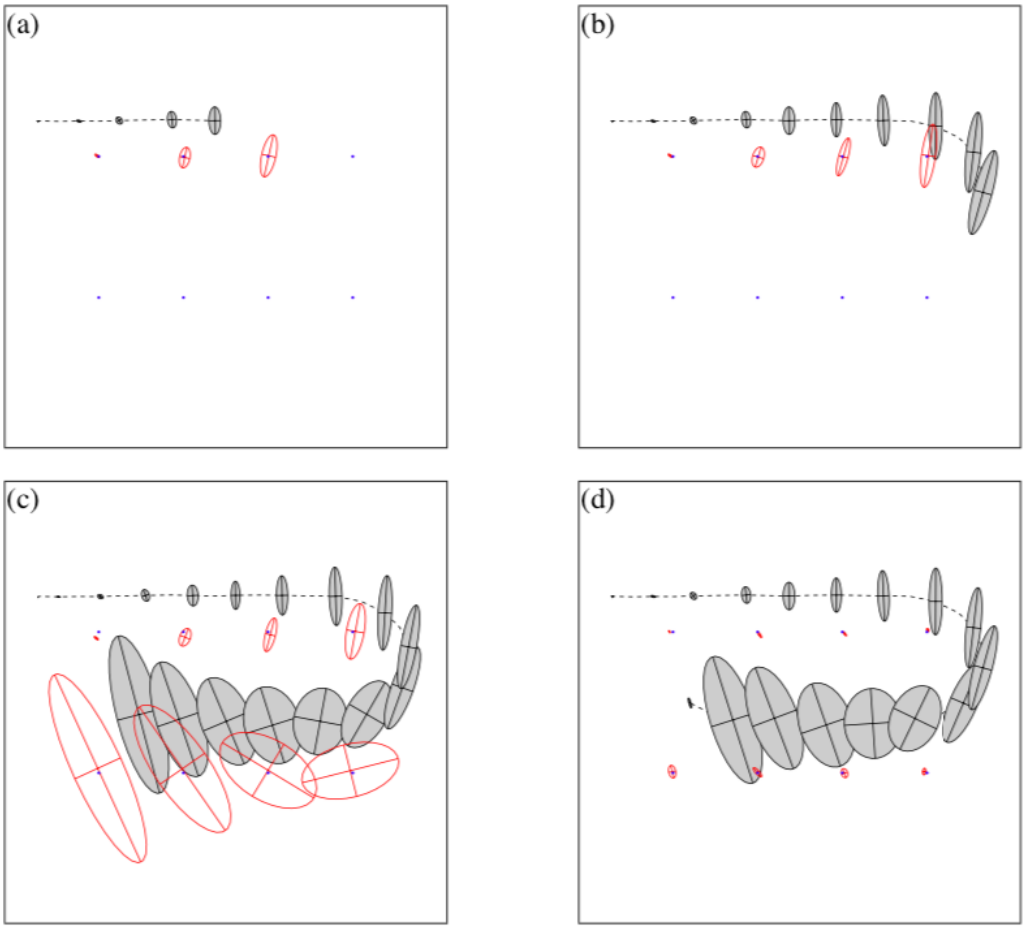

Figure 5: Figure 10.1 from TBF [1]. EKF applied to the online SLAM problem. The robot's path is a dotted line, and its estimations of its own position are shaded ellipses. Eight distinguishable landmarks of unknown location are shown as small dots, and their location estimations are shown as white ellipses. In (a)–(c) the robot's positional uncertainty is increasing, as is its uncertainty about the landmarks it encounters. In (d) the robot senses the first landmark again, and the uncertainty of all landmarks decreases, as does the uncertainty of its current pose.

Figure 5 is the figure we seen in class. It illustrates how EKF SLAM algorithm works for an artificial example. The robot moves from the origin of its coordinate system. The uncertainty of its pose increases as time goes, which is represented as the ellipses. And the uncertainty of the landmarks also grows with time.

However, in the last figure (d), we can see that the uncertainty decrease a lot. This is because the robot observes the landmark it saw in the very beginning of mapping, and

whose location is relatively well known. This observation reduces the pose error of the robot as well as other landmarks in the map.

This phenomenon arises from a correlation that is expressed in the co-variance matrix of the Gaussian posterior. Since most of the uncertainty in earlier landmarks is causes by the robot pose, and this very uncertainty persists over time, the location estimates of those landmarks are correlated. When gaining information on the robot's pose, this information spreads to previously observed landmarks. This effect is probably the most important characteristic of the SLAM posterior: Information that helps localize the robot is propagated through map, and as a result improves the localization of other landmarks in the map.

2.3 Algorithm

The EKF SLAM algorithm estimate the robot pose x_t and the coordinates of all the landmarks meets on the way robot moves. So it is necessary to have all the landmark coordinates to be included in the estimate process. This is the main difference compare with the EKF localization algorithm.

Algorithm EKF_SLAM_known_correspondences($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t$):

$$F_x = \begin{pmatrix} 1 & 0 & 0 & \underbrace{0 \cdots 0}_{3N} \\ 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 \end{pmatrix}$$

$$\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t} \sin \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \frac{v_t}{\omega_t} \cos \mu_{t-1,\theta} - \frac{v_t}{\omega_t} \cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \omega_t \Delta t \end{pmatrix}$$

$$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t} \cos \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t} \sin \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$$

$$\bar{\Sigma}_t = G_t \; \Sigma_{t-1} \; G_t^T + F_x^T \; R_t \; F_x$$

$$Q_t = \begin{pmatrix} \sigma_r^2 & 0 & 0 \\ 0 & \sigma_\phi^2 & 0 \\ 0 & 0 & \sigma_s^2 \end{pmatrix}$$

for all observed features $z_t^i = (r_t^i \;\; \phi_t^i \;\; s_t^i)^T$ *do*

$j = c_t^i$

if landmark j *never seen before*

$$\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \\ \bar{\mu}_{j,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{pmatrix} + \begin{pmatrix} r_t^i \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ r_t^i \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$$

endif

$$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{j,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{j,y} - \bar{\mu}_{t,y} \end{pmatrix}$$

$$q = \delta^T \delta$$

$$\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \text{atan2}(\delta_y, \delta_x) - \bar{\mu}_{t,\theta} \\ \bar{\mu}_{j,s} \end{pmatrix}$$

$$F_{x,j} = \begin{pmatrix} 1 & 0 & 0 & \underbrace{0 \cdots 0}_{3j-3} & 0 & 0 & 0 & \underbrace{0 \cdots 0}_{3N-3j} \\ 0 & 1 & 0 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 & 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 0 & 0 & 1 & 0 \cdots 0 \end{pmatrix}$$

$$H_t^i = \frac{1}{q} \begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & \sqrt{q}\delta_y & 0 \\ \delta_y & -\delta_x & -q & -\delta_y & +\delta_x & 0 \\ 0 & 0 & 0 & 0 & 0 & q \end{pmatrix} F_{x,j}$$

$$K_t^i = \bar{\Sigma}_t \; H_t^{iT} (H_t^i \; \bar{\Sigma}_t \; H_t^{iT} + Q_t)^{-1}$$

$$\bar{\mu}_t = \bar{\mu}_t + K_t^i (z_t^i - \hat{z}_t^i)$$

$$\bar{\Sigma}_t = (I - K_t^i \; H_t^i) \; \bar{\Sigma}_t$$

endfor

$$\mu_t = \bar{\mu}_t$$

$$\Sigma_t = \bar{\Sigma}_t$$

return μ_t, Σ_t

Figure 6: Table 10.1 from [1]. The EKF algorithm for the SLAM problem (with known correspondences).

2.4 Derivation

The derivation of the EKF SLAM algorithm for the case of known correspondences is similar to the EKF. The key difference is the augmented state vector, which now includes the locations of all landmarks in addition to the robot pose.

The following initial mean and covariance express this belief:

$$\mu_t = \begin{pmatrix} 0 & 0 & 0 & ... & 0 \end{pmatrix}^T$$

$$\Sigma_0 = \begin{pmatrix} 0 & 0 & 0 & \infty & \dots & \infty \\ 0 & 0 & 0 & \infty & \dots & \infty \\ 0 & 0 & 0 & \infty & \dots & \infty \\ \infty & \infty & \infty & \infty & \dots & \infty \\ \vdots & \vdots & \vdots & \vdots & \ddots & \dots \\ \infty & \infty & \infty & \infty & \dots & \infty \end{pmatrix}$$

The covariance matrix is the size of $(3N+3)x(3N+3)$. It is composed of a small 3x3 matrix of zeros for the robot pose variables. All other covariances values are infinite.

In SLAM, this motion model is extended to the augmented state vector but because the motion only affects the robot's pose and all other landmarks remain where they are, only the first three elements in the update are non-zero. This makes the equation more compact:

$$Y_t = Y_{t-1} + F_x \begin{pmatrix} -\frac{v_t}{w_t} sin\theta + \frac{v_t}{w_t} sin\theta sin(\theta + w_t\Delta t) \\ \frac{v_t}{w_t} cos\theta - \frac{v_t}{w_t} cos(\theta + w_t\Delta t) \\ w_t\Delta t + \gamma_t\Delta t \end{pmatrix}$$

Where

$$F_x = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \end{pmatrix}$$

The full motion model with noise is then as follows:

$$Y_t = Y_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{w_t} sin\theta + \frac{v_t}{w_t} sin(\theta + w_t\Delta t) \\ \frac{v_t}{w_t} cos\theta - \frac{v_t}{w_t} cos(\theta + w_t\Delta t) \\ w_t\Delta t + \gamma_t\Delta t \end{pmatrix} + N(0, F_x^T R_t F_x)$$

$$g(u_t, y_{t-1}) = Y_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{w_t} sin\theta + \frac{v_t}{w_t} sin\theta sin(\theta + w_t\Delta t) \\ \frac{v_t}{w_t} cos\theta - \frac{v_t}{w_t} cos(\theta + w_t\Delta t) \\ w_t\Delta t + \gamma_t\Delta t \end{pmatrix}$$

Where $F_x^T R_t F_x$ extends the covariance matrix to the dimension of the full state vector squared.

As usual in EKF's, the motion function g is approximated using a first degree Taylor expansion. With this we have the Jacobian $g_t$that characterizes the change of the robot pose:

$$G_t = I + F_x^T g_t F_x$$

Where

$$g_t = \begin{pmatrix} 0 & 0 & \frac{v_t}{w_t} cos\mu_{t-1,\theta} - \frac{v_t}{w_t} cos(\mu_{t-1,\theta} + w_t \Delta t) \\ 0 & 0 & \frac{v_t}{w_t} sin\mu_{t-1,\theta} - \frac{v_t}{w_t} sin(\mu_{t-1,\theta} + w_t \Delta t) \\ 0 & 0 & 0 \end{pmatrix}$$

Plugging these approximations into the standard EKF algorithm gives us Line 2 through 5 from figure 6.

The result of this update are the mean $\bar{\mu}_t$ and the covariance $\bar{\Sigma}_t$ of the estimate at time t after updating the filter with the control μ_t, but before integrating the measurement z_t.

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

$$h_{(y_t,j)} = \begin{pmatrix} \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2} \\ atan2(m_{j,y} - y, m_{j,x} - x) - \theta \\ m_{j,s} \end{pmatrix}$$

$$Q_t = N(0, \begin{pmatrix} \sigma_r & 0 & 0 \\ 0 & \sigma_\phi & 0 \\ 0 & 0 & \sigma_s \end{pmatrix})$$

Where i is the index of an individual landmark observation in z_t, and j = c_t^i is the index of the observed landmark at time t.

$$H_t^i = h_t^i F_{x,j}$$

Here $H_t^i$is the derivative of h with respect to the full state vector y_t. h_t^i is the Jacobian of the function $h(y_t, j)$ at $\bar{\mu}_t$, calculated with respect to the state variables x_t and m_j.

The scalar $q_t = (m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2$, and as before, j = c_j^i is the landmark that corresponds to the measurement z_t^i. The matrix $F_{x,j}$ is the dimension (3N +3)x5. These expressions make up for the gist of the kalman gain calculation in Line 8 through 17 in

the EKF SLAM algorithm.

When a landmark is observed for the first time, its initial pose estimate in μ_0 leads to a poor linearization. This is because with the default initialization in μ_0, the point about which h is being linearized is $(\hat{\mu}_{j,x}\hat{\mu}_{j,y}\hat{\mu}_{j,s})^T = (000)^T$, which is a poor estimator of the equal landmark location. A better landmark estimate $(\hat{\mu}_{j,x}\hat{\mu}_{j,y}\hat{\mu}_{j,s})^T$ with expected position:

$$\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \\ \bar{\mu}_{j,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{pmatrix} + r_t^i \begin{pmatrix} cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ sin(\phi_t^i + \bar{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$$

This initialization is only possible because the measurement function h is bijective.

References

[1] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic robotics. MIT Press, Cambridge, Mass., 2005.