Raspberry Pi GoPiGo Robot EKF SLAM Implementation

Gopal Menon
Computer Science Department
Utah State University
Logan, Utah 84322
Email: gopal.menon@aggiemail.usu.edu

***Abstract*—Robot motion is stochastic and not deterministic. A robot does not faithfully execute movement commands due to inaccuracies in its motion mechanism, approximations made in modeling its environment and incomplete modeling of the physics of its operating environment. Due to these reasons, the robot may not end up exactly at the location where it should have been if the movement command had been faithfully executed. The robot position may be thought of as a two dimensional probability distribution. This probability distribution will have a mean that will be equal to the expected robot location based on the movement command. The covariance will be the measure of how faithfully it executes commands. Due to this uncertainty in movement, with each motion command, the location uncertainty increases and after a while the robot gets lost. In some applications, robots need to be able to go into an unknown environment and explore it. This is a difficult problem due to the issue described above. If a map of the environment is available, the robot can use sensors to find its location in the environment. Conversely, if the robot knows where it is, it can generate an environment map. This is a sort of chicken and end problem. Simultaneous Localization and Mapping (SLAM) solves this problem. An Extended Kalman Filter (EKF) can be used to reduce robot location uncertainty. This document describes the implementation of EKF SLAM using a robot.**

I. Introduction

A. The robot platform

The robot used for this implementation is based on a Raspberry Pi and is manufactured and sold by Dexter Industries [1]. The robot is equipped with an ultrasonic sensor that is capable of measuring distances accurately to 250 cm. The sensor is mounted on a servo mechanism that can point the sensor in any direction ahead of the robot. The robot moves through use of two wheels that can be independently controlled by motors. Each wheel has an encoder using which the robot can issue commands for a wheel to move a fixed number of steps. The robot also has a third castor wheel that can move in any direction, but is not powered. The robot will be given commands to make it move and it can sense its surroundings using the ultrasonic sensor.

B. EKF SLAM

A Kalman filter is a gaussian filter algorithm that can use two gaussian distributions for a computed value and merge them together to form a distribution that has lesser variance than either of the two distributions. A Kalman filter however only works in the case of systems where the mechanics are linear. This is where the Extended Kalman Filter (EKF) comes in. An EKF uses an approximation to find the resultant distribution after a prior dostribution is transformed by a non-linear function.

The robot will compute its expected position based on its position and the movement command. It will also make independent measurements of its position using sensors. These two measurements will be combined through use of an EKF in such a way that the position uncertainty is reduced. The Kalman Gain or the reduction in the robot position uncertainty is inversely proportional to the sensor uncertainty. The better the sensor, the higher will be the Kalman Gain and the less uncertainty there will be in the robot position[5].

II. Problem description

A. The Bayes Filter

The Bayes Filter is an algorithm used for calculating beliefs[4]. In our case it is the belief of the robot position. This algorithm is used to compute the belief probability distribution bel from measurement and control data. The measurement data is obtained from sensors and the control data is obtained from the robot command or telemetry.

Bayes Rule says that

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \tag{1}$$

Since the denominator in Bayes Rule does not depend on x, the factor $\frac{1}{p(y)}$ can be written as η. So Bayes Rule in 1 can be written as

$$p(x|y) = \eta p(y|x)p(x) \tag{2}$$

Bayes Rule 1 can also be written as

$$p(x|y,z) = \frac{p(y|x,z)p(x|z)}{p(y|z)} \tag{3}$$

as long as $p(y|z) > 0$.

Applying 3 on the robot position x_t gives us

$$p(x_t|z_{1:t}, u_{1:t}) = \frac{p(z_t|x_t, z_{1:t-1}, u_{1:t})p(x_t|z_{1:t-1}, u_{1:t})}{p(z_t|z_{1:t-1}, u_{1:t})} = \eta p(z_t|x_t, z_{1:t-1}, u_{1:t})p(x_t|z_{1:t-1}, u_{1:t}) \quad (4)$$

Since the measurement z_t depends on position x_t and any past measurement $z_{1:t}$ or command $u_{1:t}$ does not provide any additional information on the measurement, we can say that

$$p(z_t|x_t, z_{1:t-1}, u_{1:t}) = p(z_t|x_t) \quad (5)$$

This allows us to simplify 4 as

$$p(x_t|z_{1:t}, u_{1:t}) = \eta p(z_t|x_t)p(x_t|z_{1:t-1}, u_{1:t}) = \eta p(z_t|x_t)\overline{bel}(x_t) \quad (6)$$

where

$$\overline{bel}(x_t) = p(x_t|z_{1:t-1}, u_{1:t}) \quad (7)$$

Using the theorem of total probability

$$p(x) = \int p(x|y)p(y)dy \quad (8)$$

equation 7 can be written as

$$\overline{bel}(x_t) = \int p(x_t|x_{t-1}, z_{1:t-1}, u_{1:t})p(x_{t-1}|z_{1:t-1}, u_{1:t})dx_{t-1} \quad (9)$$

If we know the state x_{t-1}, past measurements and controls have no bearing on the state x_t. So

$$p(x_t|x_{t-1}, z_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, u_t) \quad (10)$$

We can see that command u_t can be omitted from the set of conditioning variables in $p(x_{t-1}|z_{1:t-1}, u_{1:t})$. This gives us the recursive update equation 7 as

$$\overline{bel}(x_t) = \int p(x_t|x_{t-1}, u_t)p(x_{t-1}|z_{1:t-1}, u_{1:t-1})dx_{t-1} \quad (11)$$

This reduces to

$$\overline{bel}(x_t) = \int p(x_t|x_{t-1}, u_t)bel(x_{t-1})dx_{t-1} \quad (12)$$

B. The Bayes Filter Algorithm

Based on the above mathematical derivation for a Bayes Filter, we can write down the Bayes Filter Algorithm as follows

Algorithm 1 Bayes Filter Algorithm

1: **procedure** BAYESFILTERALGORITHM($bel(x_{t-1}, u_t, z_t)$)
2: **for** all x_t **do**
3: $\overline{bel}(x_t) = \int p(x_t|x_{t-1}, u_t)bel(x_{t-1})dx_{t-1}$
4: $bel(x_t) = \eta p(z_t|x_t)\overline{bel}(x_t)$
5: **end for**
6: **end procedure**

The belief $bel(x_t)$ at time t is computed from the belief $bel(x_{t-1})$ at time $t-1$. The input to the algorithm is the belief bel at time $t-1$, along with the most recent control value u_t and the most recent measurement z_t. Its output is the belief $bel(x_t)$ at time t.

The Bayes Filter Algorithm consist of two essentials steps. In line 3, which is the prediction step, it calculates its belief over state x_t based on the prior belief over state x_{t-1} and the control u_t. The belief $\overline{bel}(x_t)$ that the robot assigns to state x_t is obtained by the integral (or in other words the sum) of the product of two distributions: the prior assigned to x_{t-1}, and the probability that control u_t induces a transition from x_{t-1} to x_t.

The second step of the Bayes Filter is called the measurement update. In line 4, the Bayes Filter algorithm multiplies the belief $\overline{bel}(x_t)$ by the probability that the measurement z_t may have been observed. It does this for every hypothetical posterior state x_t.

C. The Kalman Filter

The Kalman Filter is a technique for implementing a Bayes Filter[4]. This is used for filtering and prediction in linear Gaussian systems.

A probability distribution of a one-dimensional normal distribution with mean μ and variance σ^2 is given by the following Gaussian function

$$p(x) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right\} \quad (13)$$

The Normal distribution 13 is for a scalar x. In the case of a robot, the robot location on a flat surface is given by $\begin{bmatrix} x & y & \theta \end{bmatrix}^T$ where x is the x coordinate, y is the y coordinate and θ is the robot orientation. This vector is called the robot pose. A Normal distribution over such a vector is called *multivariate*. These distributions have the following form

$$p(x) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\} \quad (14)$$

where μ is the mean vector that has the same dimensionality as the state x. Σ is the covariance matrix and its dimension is the dimensionality of the state x squared. Equation 14 is a generalization of equation 13. Both definitions are equivalent if x is a scalar and value and $\Sigma = \sigma^2$.

For a Kalman filter, in posteriors are Gaussian if the following three properties hold

1) The state transition probability $p(x_t|x_{t-1}, u_t)$ must be a linear function of its arguments with added Gaussian noise. This is expressed by the following equation

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \tag{15}$$

where A_t and B_t are matrices. A_t has the dimension equal to the square of the state vector, B_t has the dimension of $n \times m$, where n is the dimension of the state vector and m is the dimension of the control variable u_t. The random variable ε_t models the uncertainty introduced by the state transition and depends on the robot mechanics. Its mean is zero and covariance will be denoted by R_t. The state transition probability distribution can be obtained by plugging in 15 into 14. The mean of the posterior state is given by $A_t x_{t-1} + B_t u_t$ and the covariance by R_t.

2) The measurement probability $p(z_t|x_t)$ must also be linear in its arguments with added Gaussian noise

$$z_t = C_t x_t + \delta_t \tag{16}$$

Here C_t is a matrix of size $k \times n$, where k is the dimension of the measurement vector z_t. The vector δ_t describes the measurement noise. The distribution of δ_t is a multivariate Gaussian with zero mean and covariance Q_t.

3) Finally the initial belief $bel(x_0)$ must be normally distributed. We can denote the mean of this belief by mean μ_0 and covariance Σ_0.

The Kalman filter algorithm is given below

Algorithm 2 Kalman Filter Algorithm

1: **procedure** KALMAN FILTER ALGORITHM($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)
2: $\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t$
3: $\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$
4: $K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1}$
5: $\mu_t = \overline{\mu}_t + K_t(z_t - C_t \overline{\mu}_t)$
6: $\Sigma_t = (I - K_t C_t)\overline{\Sigma}_t$
7: return μ_t, Σ_t
8: **end procedure**

The algorithm represents the belief $bel(x_t)$ at time t by the mean μ_t and covariance Σ_t. The input is the belief at time $t-1$, the control u_t and measurement z_t. The output is the belief at time t.

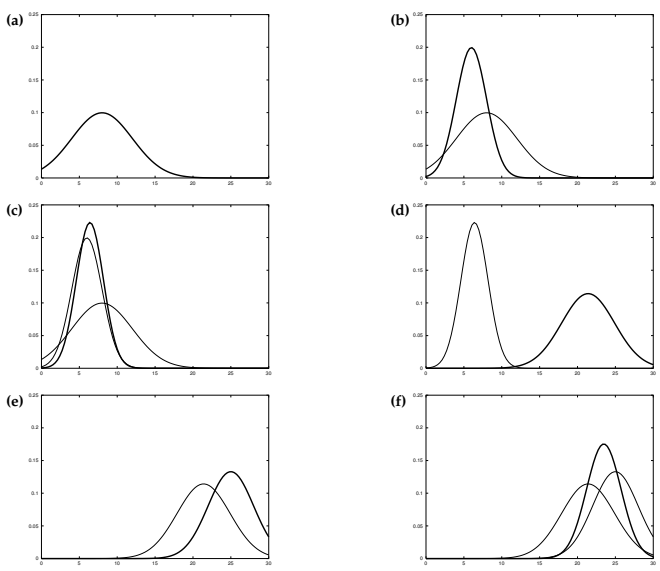

Fig. 1. Illustration of Kalman filters: (a) initial belief, (b) a measurement (in bold) with the associated uncertainty, (c) belief after integrating the measurement into the belief using the Kalman filter algorithm, (d) belief after motion to the right (which introduces uncertainty), (e) a new measurement with associated uncertainty, and (f) the resulting belief.

In lines 2 and 3, the predicted belief $\overline{\mu}$ and $\overline{\Sigma}$ is calculated representing the belief $\overline{bel}(x_t)$ one time step later, but before incorporating the measurement z_t. This belief is obtained by incorporating the control u_t. The mean is updated using the deterministic state transition function 15 with the mean μ_{t-1} substituted for the state x_{t-1}. The update of the covariance considers the fact that states depend on the previous state through the linear matrix A_t. The variable K_t computed in line 4 is called *Kalman gain*. This is the reduction is uncertainty that is obtained by combining the prediction and correction steps. The belief $\overline{bel}(x_t)$ is transformed into the desired belief $bel(x_t)$ in lines 5 and 6 by incorporating the measurement z_t. Line 5 manipulates the mean by adjusting it in proportion to the Kalman gain K_t and the deviation of the actual measurement z_t and the measurement predicted as per the measurement probability 16. Finally a new covariance of the posterior belief is calculated in line 6, adjusting for the information gain resulting from the measurement.

Figure 1 illustrates the Kalman filter for a simplistic one-dimensional localization scenario. Suppose the robot moves in the horizontal axis in each diagram in figure 1 with the prior over the robot localization given by the normal distribution shown in figure 1a. The robot queries its sensors on its location and these return a measurement that is centered at the peak of the bold Gaussian distribution in figure 1b. The peak corresponds to the value returned by the sensors and the its width or variance corresponds to

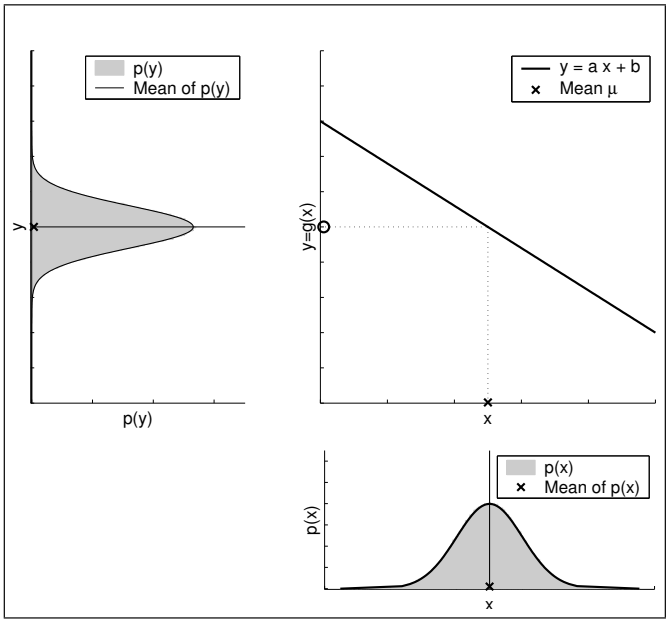

Fig. 2. Linear transformation of a Gaussian random variable. The lower right plot show the density of the original random variable, X. This random variable is passed through the function displayed in the upper right graph (the transformation of the mean is indicated by the dotted line). The density of the resulting random variable Y is plotted in the upper left graph.

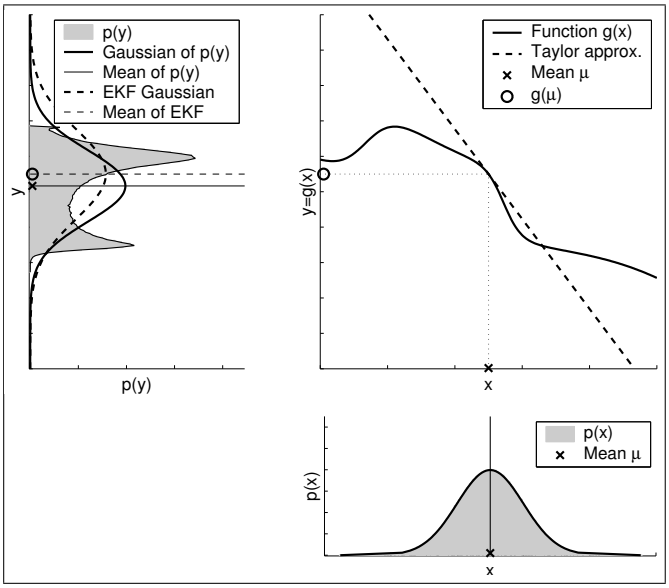

Fig. 3. Illustration of linearization applied by the EKF. Instead of passing the Gaussian through the nonlinear function g, it is passed through a linear approximation of g. The linear function is tangent to g at the mean of the original Gaussian. The resulting Gaussian is shown as the dashed line in the upper left graph. The linearization incurs an approximation error, as indicated by the mismatch between the linearized Gaussian (dashed) and the Gaussian computed from the highly accurate Monte-Carlo estimate (solid).

the uncertainty in measurement. Combining the prior with the measurement in lines 4 through 6 of the Kalman filter algorithm, yields the bold Gaussian in figure 1c. The belief's mean lies between the two original means, and its uncertainty width is smaller than both contributing Gaussians. The is the result of the information gain from using the Kalman filter.

Now lets assume that the robot moves towards the right. Its uncertainty grows due to the fact that the state transition is stochastic. Lines 2 and 3 of the Kalman filter algorithm provide us with the Gaussian shown in bold in figure 1d. This Gaussian is shifted by the amount the robot moved and is also wider because of the uncertainty introduced by the movement. The robot does a second sensor measurement shown by the bold Gaussian in figure 1e, which leads to the posterior shown in bold in figure 1f.

As this example illustrates, the Kalman filter alternates a measurement update step, in which sensor data is integrated into the present belief, with a prediction step, which modifies the belief. The prediction step increases and the update step decreases the uncertainty in the robot's belief.

D. The Extended Kalman Filter

For a Kalman filter to work, all the transformations need to be linear. Figure 2 illustrates the linear transform of a one-dimensional Gaussian random variable. The graph on the lower right shows the density of this random variable having mean μ and variance σ^2. The random variable is passed through a linear function $y = ax + b$ shown in the upper right graph. The resulting random variable will have Gaussian distribution with mean $a\mu + b$ and variance $a^2\sigma^2$. This resulting Gaussian is sown in the upper left graph of figure 2.

Unfortunately state transitions and measurements are rarely linear in practice[4]. This renders Kalman filters inapplicable to most robotics problems. The Extended Kalman filter, or EFK, relaxes one of the assumptions. Here the assumption is that state transition probabilities and landmark measurement probabilities are governed by nonlinear functions g and h respectively:

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t \tag{17}$$

$$z_t = h(x_t) + \delta_t \tag{18}$$

Here x_t is the robot location at time t, u_t is the robot command that causes it to reach position x_t, z_t is the landmark measurement at time t, ϵ_t is the robot motion uncertainty that depends of the mechanical properties of the robot and δ_t is the sensor measurement uncertainty.

Figure 3 illustrates the impact of a nonlinear transformation on a Gaussian random variable. The graph on the upper right plots the nonlinear function g. As you can see, the transformed function is not a Gaussian. The idea behind the EKF is linearization. The nonlinear function is approximated at mean of the Gaussian. Projecting this Gaussian through this linear approximation results in a Gaussian density as represented by the dotted line in the upper left of figure 3.

The Extended Kalman filter algorithm is given below

Algorithm 3 Extended Kalman Filter Algorithm

1: **procedure** EXTENDED KALMAN FILTER ALGORITHM($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)
2: $\overline{\mu}_t = g(u_t, \mu_{t-1})$
3: $\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
4: $K_t = \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1}$
5: $\mu_t = \overline{\mu}_t + K_t(z_t - h(\overline{\mu}_t))$
6: $\Sigma_t = (I - K_t H_t)\overline{\Sigma}_t$
7: return μ_t, Σ_t
8: **end procedure**

As can be seen, the EKF algorithm is similar to the Kalman filter algorithm in may ways. EKFs use Jacobians G_t and H_t instead of the corresponding linear system matrices A_t, B_t and C_t in Kalman filters. The Jacobian G_t corresponds to the matrices A_t and B_t, and the Jacobian H_t corresponds to C_t.

III. Approach/Methods

IV. Results

V. Discussion

VI. Future work

References

[1] GoPiGo - Dexter Industries. *Dexter Industries*. N.p., n.d. Web. 06 Nov. 2015. http://www.dexterindustries.com/gopigo/.

[2] Durrant-Whyte, H., and T. Bailey, *Simultaneous Localization and Mapping: Part I*, IEEE Robotics and Automation Magazine. 13.2 (2006): 99-110.

[3] Bailey, Tim, and Hugh Durrant-Whyte, *Simultaneous localization and mapping (SLAM): Part II*, IEEE Robotics and Automation Magazine. 13.3 (2006): 108-117.

[4] Thrun, Sebastian, Wolfram Burgard, and Dieter Fox, *Probabilistic Robotics*, MIT press, 2005.

[5] Stachniss, Cyrill, *SLAM Course - WS13/14*, YouTube. YouTube, n.d. Web. 18 Sept. 2015. https://www.youtube.com/playlist?list=PLgnQpQtFTOGQrZ4O5QzbIHgl3b1JHimN_.