INTRODUCTION TO THE EXTENDED KALMAN FILTER

INTRODUCTION TO STATE ESTIMATION

State estimation like the name implies refers to the problem of determining the state of a robot. But what is the state and why do we care about it? Early sailors were faced with the problem of determining where their ship was at sea and so had to develop maps of the oceans by using their compass, charting the sun and stars and observing coastlines. In a similar manner, for a robot to navigate in its environment, it needs to know where it is and where objects are in its environment. Hence the two central questions in State Estimation are

- Where are am I in the world? – Localization
- Where are the objects/landmarks in the environment? -Mapping

The answers to the two questions above form components of the state of the robot. As defined in the book "Probabilistic Robotics" by Sebastian Thrun et al, the state is referred to as all components of the environment that impact the future. In more concrete terms, the state of a robot is a set of quantities such as its position and velocity. The state can also be expanded to include the positions of objects in the environment.

SOLUTION TO THE STATE ESTIMATION PROBLEM

One can choose to estimate the state of a robot by simply integrating the velocity commands to obtain the position. However, this solution does not work in the real environment because of noise. This noise may be due to environmental effects like a rough terrain, winds or may be as a result of inaccuracies/inefficiencies in the robot's movement. As such, we conclude that we cannot know the exact state of the robot with perfect certainty. This inherent uncertainty in the dynamics of a robot and its environment is what inspired the probabilistic approach/solution to the state estimation problem.

This probabilistic approach led to the two key aspects of the robot's interaction with its environment, namely motion and measurement being modeled as probability density functions. The issue then is how to propagate the probability density functions(pdf) as the robot moves in the world. A key insight that researchers found was that Bayes rule and the Bayes Filter provided consistent and rigorous way to model the propagation of pdfs.

Briefly, the Bayes filter is a generic algorithm that combines our motion and measurement uncertainties, attempts to reduce the noise and recover the best estimate of our state. The Extended Kalman Filter is but one of a family of techniques that fall under the Bayes Filter.

EXTENDED KALMAN FILTER

The Extended Kalman Filter (EKF) is a modification of the popular Kalman Filter. The EKF is designed to handle nonlinear motion and measurement models which is what we mostly encounter in the real world. In essence, the EKF provides a Gaussian approximation to the state of the robot. I provide below a brief summary of the main terms and equations used in the EKF.

EKF EQUATIONS

1. Prediction

$$\bar{\mu}_t = g(u_t, \mu_{t-1})$$
$$\bar{\Sigma}_t = G\Sigma_{t-1}G^T + R$$

2. Measurement/Correction

$$K_t = \bar{\Sigma}_t H^T (H\,\bar{\Sigma}_t H^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$$
$$\Sigma_t = (I - K_t H)\bar{\Sigma}_t$$

Recalling that we are working with random variables and probability distributions, here is a short description of the meaning of the terms in the EKF equations:

$\bar{\mu}_t$: This is the mean of the Gaussian estimate of the state at time t.

$g(u_t, \mu_{t-1})$: The nonlinear motion function that estimates our current state from the previous state and control input

Σ_t: This refers to the covariance matrix of the state distribution

G: The Jacobian of the motion function g. This Jacobian function is used to update the covariance matrix Σ_t

R: The covariance matrix of a zero mean Gaussian error that represents motion noise

Q_t : The covariance matrix pf a zero mean Gaussian error that represents sensor disturbances

$h(x)$: The is the measurement function that maps our state to a measurement

H: The Jacobian of h. It is used to compute the Kalman gain and for updating the state covariance matrix.

K_t: The Kalman gain that is used to update the state and covariance matrix from the previous time step. If we examine the last two equations of the measurement/correction step, we see

that the mean and covariance of the state are formed using a linear combination of terms in which the Kalman gain functions as a weight. If we have perfect measurement, The Kalman gain discards the prediction update and gives full weight to the measurement-essentially mapping our measurement vector back to the state vector.

For EKF localization, we execute the prediction and measurement update for each time step using the equations outlined above. My implementation of EKF localization made use of the same equations. Due to the fact that my implementation of the EKF was in ROS using Python, I have provided a brief summary of my EKF localization node.

EKF LOCALIZATION NODE

- Package name: waffle_slam
- Node name: ekf_localization
- File name: ekf_localization.py
- Subscribed topics: cmd_vel, imu and odom topics. I specifically use the data form the odom topic for my measurement
- Published topics: odom_combined

The variant of the localization problem I implement is position tracking. The robot's initial position was set at x =0, y =0 and bearing/yaw = 0. Control input on the cmd_vel topic was provided via the turtlebot3_teleop_keyboard node in the turtlebot3_teleop package.

EKF SLAM

SLAM is a more involved and complex problem that localization. This is because the dimensionality of our problem has increased as we are tracking more state variables. In addition, we have to solve the data association problem in which we must resolve which measurements correspond to particular landmarks.

I implement the EKF SLAM algorithm with unknown correspondences as described in Chapter 10, pages 321-322 of the book Probabilistic Robotics. It will be tedious to reproduce the algorithm as described in the pages of the book, but I provide a very short pseudocode version which can be seen on the next page.

Time(t) = 0

Move robot using motion model(g)

For each observed feature(z)

Initialize the landmark's location

Get sensor data

For each landmark in the map

Compute Maximum Likelihood estimate based on Mahalanobis distance

If distance to all known landmarks exceeds a threshold, add new landmark

and update state and covariance quantities to reflect increased dimensions

Compute Kalman gain and update the state and covariance quantities

Return the updated state and covariance

My implementation of EKF SLAM essentially follows the description of the algorithm in the Probabilistic Robotics book. The key difference is that I do not have signature variable(s_t) which the book utilizes. As a result of the omission of the signature variable, all my other matrix dimensions change accordingly. For example, in the book, $h(x)$ is a matrix of dimension 3 x 6 while in my case $h(x)$ is of dimension 2 x 5.

CONCLUSION

Implementing the EKF serves as a very good introduction into the topic of SLAM. My implementation is still not very robust and work will continue on it to make the robot pose estimation better and to reduce the occurrence of spurious features which corrupts the whole state estimation process. Further work will also be done on Particle Filters and the package will be updated to reflect that.

REFERENCES

Sebastian Thrun, Wolfram Burgard, Dieter Fox (2006) *Probabilistic Robotics* MIT Press