16.811 Project Report: Kalman Filter SLAM

Rockey Hester

December 13, 2019

1 Introduction

The Kalman Filter and its derivatives are ubiquitous techniques for state estimation, but undergraduate courses in controls and robotics commonly gloss over the details of their derivation and implementation due to the availability of many open source libraries and their difficulty of implementation. This project is an attempt to fill in some of the gaps in my knowledge by exploring the derivation and implementation of the Kalman Filter and how they can be applied to one of my favorite problems in robotics - Simultaneous Localization and Mapping (SLAM). First the Kalman Filter equations are presented and explained along with some other considerations for implementation and application to SLAM. Then, the dynamics for a particular system, the commonly used Jackal wheeled robot, are derived. Lastly, the Kalman Filter state estimate for a simulated Jackal robot is evaluated by comparing it to the state estimate resulting from simply integrating the equations of motion for a driving task.

2 Related Works

Choset et. al derive the linear Kalman Filter, Extended Kalman Filter (EKF), and Kalman Filter SLAM in [1]. The system dynamics are formulated as

$$\begin{aligned} x(k+1) &= F(k)x(k) + G(k)u(k) + v(k) \\ y(k) &= H(k)x(k) + w(k) \quad [1] \end{aligned}$$

for a linear system, where $x \in \mathbb{R}^n$ is the system state, $F \in \mathbb{R}^{n\times n}$ is the system dynamics, $u \in \mathbb{R}^m$ is the input, $G \in \mathbb{R}^{n\times m}$ is the input dynamics, $v \in \mathbb{R}^n$ with covariance $V(k) \in \mathbb{R}^{n\times n}$ is the process noise, $y \in \mathbb{R}^p$ is the output, $H \in \mathbb{R}^{p\times n}$ is the sensor dynamics, $w \in \mathbb{R}^p$ with covariance $W(k) \in \mathbb{R}^{p\times p}$ is the measurement noise, and $k \in \mathbb{Z}^+$ is the time step.

The Kalman Filter is a two step process. First, the current state, the system dynamics, and the system inputs are used to *predict* the next state (or rather, a distribution over the next state with mean $\hat{x}$ and covariance P). Second, the prediction is *updated* using the sensor data by weighting the innovation ν (the difference between the actual measurement and the measurement we would expect to receive from the predicted state) with the reliability of the sensors.

The prediction equation is as follows:

$$\begin{aligned} \hat{x}(k+1|k) &= F(k)\hat{x}(k|k) + G(k)u(k) \\ P(k+1|k) &= F(k)P(k|k)F(k)^T + V(k) \quad [1] \end{aligned}$$

The update equation is as follows:

$$\begin{aligned} \hat{x}(k+1|k+1) &= \hat{x}(k+1|k) + R\nu \\ P(k+1|k+1) &= P(k+1|k) - RH(k+1)P(k+1|k) \quad [1] \end{aligned}$$

where

$$\nu = y(k+1) - H(k+1)x(k+1|k)$$
$$S = H(k+1)P(k+1|k)H(k+1)^T + W(k+1)$$
$$R = P(k+1|k)H(k+1)^T S^{-1} \quad [1]$$

The EKF is a modified version of the linear Kalman Filter that accounts for systems with nonlinear dynamics (which means that they're unable to be formulated into the matrices F, G, and H).

The new system dynamics are defined as

$$x(k+1) = f(x(k), u(k), k) + v(k)$$
$$y(k) = h(x(k), k) + w(k) \quad [1]$$

where all is the same as before except f is a function mapping the state and input to $\mathbb{R}^n$ and h is a function mapping the state to $\mathbb{R}^p$. All that must be done to get these equations into a usable form is to linearize f and h.

The prediction equations are then

$$\hat{x}(k+1|k) = f(\hat{x}(k|k), u(k), k)$$
$$P(k+1|k) = F(k)P(k|k)F(k)^T + V(k) \quad [1]$$

where

$$F(k) = \frac{\partial f}{\partial x}\bigg|_{x=\hat{x}(k|k)} \quad [1]$$

The update is exactly the same as before except

$$H(k+1) = \frac{\partial h}{\partial x}\bigg|_{x=\hat{x}(k+1|k)} \quad [1]$$

The Kalman Filter can be extended to solve the problem of SLAM by estimating the locations of both the robot and any landmarks detected in the environment at the same time. This can be done by extending the state x to include the positions of the landmarks (x_{li}, y_{li}) on top of the regular state of the robot and by extending the sensor model to include some measure of the relative positions of the robot and each landmark (such as range and bearing via laser scan) as well as the other sensors on the robot (e.g. IMU, wheel odometry, etc). After linearizing any nonlinear dynamics, the prediction and update equations are exactly the same as in the Kalman Filter and EKF.

Since the landmarks are assumed to be fixed, $V(k)$ is just the covariance from the robot-only state in the upper left and 0's on the rest of the diagonal. However $W(k)$ must include a covariance value corresponding to each of the landmarks since there is still noise in the measurement of the landmarks' locations.

One relative position measurement is the following range and bearing model, which measures the distance and angle from the robot to each landmark l_i.

$$y_i(k) = \begin{pmatrix} \sqrt{(x_{li}(k) - x_r(k))^2 + (y_{li}(k) - y_r(k))^2} \\ \texttt{atan2}(y_{li}(k) - y_r(k)), (x_{li}(k) - x_r(k)) \end{pmatrix} + w_i(k) \quad [1]$$

The final consideration for Kalman Filter SLAM is the problem of *data association* - it is not immediately clear which landmark each measurement is associated with. This problem can be solved with the Mahalanobis distance χ.

$$\chi^2_{ij} = \nu_{ij} S^{-1}_{ij} \nu^T_{ij} = (y(k)_i - h(k)_j)^T S^{-1}_{ij} (y(k)_i - h(k)_j) \quad [1]$$

The Mahalanobis distance is computed for every pair of measurements and possible landmarks. Generally, a measurement is associated with the landmark with the lowest corresponding Mahalanobis distance. However, if there is no landmark with a sufficiently low Mahalanobis distance, this indicates that a new landmark should be initialized and added to the state.

In his lecture on the Kalman Filter [2], Simon explains the process of computing $V(k)$ and $W(k)$ for a given system. This can be done by fitting a distribution to the output of the system while it is stationary.

The Jackal robot from Clearpath Robotics [3] is a popular wheeled robot for research. It is accompanied by a simulator in ROS Gazebo and RViz, which can accommodate multiple sensors, including IMU, wheel odometry, and LiDAR.

3 Approach

The Jackal simulator in ROS Gazebo is used with data visualization in RViz. While the simulator has a built-in state estimator, I implemented an EKF in Python using ROS to better understand how it works. The system state consists of the position and orientation of the robot in the 2D plane, the inputs to the system are the linear and angular velocity u_1 and u_2, and the outputs are only the linear and angular velocity from the wheel encoders. Including linear and angular velocity in the state and linearizing, we have

$$f(x) = \begin{pmatrix} \cos\theta(k) u_1(k) + x(k) \\ \sin\theta(k) u_1(k) + y(k) \\ u_1(k) \\ u_2(k) + \theta(k) \\ u_2(k) \end{pmatrix} \rightarrow F(x) = \begin{pmatrix} 1 & 0 & \cos\theta(k) & -\sin\theta(k) u_1(k) & 0 \\ 0 & 1 & \sin\theta(k) & \cos\theta(k) u_1(k) & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$h(x) = \begin{pmatrix} u_1(k) \\ u_2(k) \end{pmatrix} \rightarrow H(x) = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

To determine the measurement and process noise distributions, we first monitor the outputs linear and angular velocity as the system is stationary.

The variation in measured linear velocity is on the order of 1e$-$5 and the variation in measured angular velocity is on the order of 1e$-$2, so we have

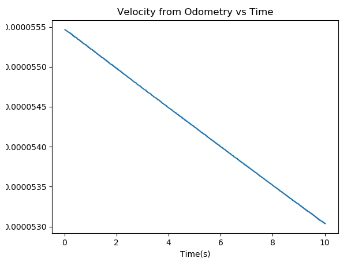

Figure 1: Variation in measured linear velocity while stationary

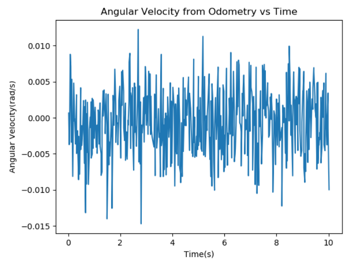

Figure 2: Varitation in measured angular velocity while stationary

$$W(k) = \begin{pmatrix} 0.00001 & 0 \\ 0 & 0.01 \end{pmatrix}$$

Next using this measurement model we monitor the state variables x, y, and θ while the system is stationary. This is done by subscribing to the Jackal simulator's built-in state estimator for the ground truth values and subtracting the ground truth values from the estimates (because we assume that the mean of the noise is the true value). The noise data is sorted, the mean and variance of the data are calculated, and the data (in red) is plotted alongside a normal distribution (in blue) to confirm the Gaussian nature of the noise.

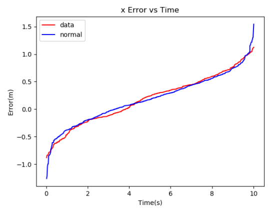

Figure 3: Variation in x while stationary

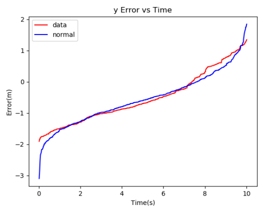

Figure 4: Variation in y while stationary

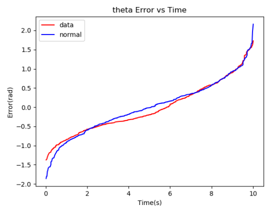

Figure 5: Variation in θ while stationary

While the variances for each variable were on the order of 1e$-$1, it was determined experimentally that much smaller noise values were required for stable performance, resulting in $V(k)$ having 1e$-$5 on the diagonal.

To evaluate the performance of the EKF, we compute the state estimate separately using the EKF and simply integrating the equations of motion and record the truth while driving the Jackal around in a simulated environment.

4 Results

The ground truth data for each variable is in blue. The EKF results are in red. The integration results are in green. There is an anomaly in the orientation results that I believe is caused by not wrapping the angle θ to $[-\pi, \pi]$.

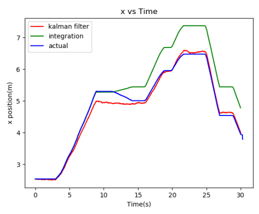

Figure 6: Comparison of estimated x with actual

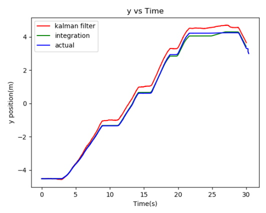

Figure 7: Comparison of estimated y with actual

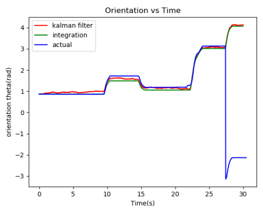

Figure 8: Comparison of estimated θ with actual

5 Discussion

The EKF performance is clearly better than integration for the x position, but is generally worse for y position and approximately equivalent for θ. This was disappointing as I had hoped the EKF performance would be far superior than just integrating the equations of motion. Some possible explanations for the discrepancy are that I only took data for 30 seconds and perhaps the drift would be worse for the integration method with more time, testing in simulation rather than on a real robot may have artificially improved the results for one or both methods, I was only using odometry and not the other available sensors like IMU and laser scan, and there may be errors in my implementation. I believe the latter is possible because the performance was highly dependent on the values in the process and measurement noise covariances. With values significantly greater than 1e$-$5, the EKF estimate jumps all over the place (literally).

Going forward, I will follow the conventional wisdom of "don't implement a Kalman Filter yourself," but at some point I would like to ignore the conventional wisdom of "don't implement SLAM yourself."

References

1. H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, "Principles of Robot Motion", MIT Press, ch. 8, pp. 289-319, 2016, isbn: 9780262303958

2. D. Simon, "A Crash Course on Kalman Filtering", Cleveland State University, 2014

3. Clearpath Robotics, "Jackal Simulator", 2015