Kalman Filter: my lab summary in one page

Problem Statement:
The objective of this lab was to implement a mathematical model of the robot sensor and actuator behaviour and use it to evaluate a state estimation algorithm. We estimated the state of simple two-wheeled non-holonomic robot using extended kalman filter (EKF).

Tasks: First we defined our Robot State as a 4 dimensional unit (x-coordinate, y-coordinate, theta and omega) and set our space dimension as 500mm x 750mm. To estimate the spatial state of our robot we used the time evolution system model and the observation model.

Time system model: (notation bar)

$$\begin{bmatrix} \overline{\theta}_{t+1} \\ \overline{x}_{t+1} \\ \overline{y}_{t+1} \\ \overline{\omega}_{t+1} \end{bmatrix} = \begin{bmatrix} \hat{\theta}_t + \omega_t \Delta t \\ \hat{x}_t + v_t cos\hat{\theta}_t \Delta t \\ \hat{y}_t + v_t sin\hat{\theta}_t \Delta t \\ \hat{\omega}_t \end{bmatrix}$$

Observation Model: (notation hat)

$$\begin{bmatrix} \phi_t \\ r_{1t} \\ r_{2t} \end{bmatrix} \approx \begin{bmatrix} tan^{-1}(\frac{y_t - \lambda_{y1}}{x_t - \lambda_{x1}}) - \theta_t \\ \sqrt{(x_t - \lambda_{x_1})^2 + (y_t - \lambda_{y_1})^2} \\ \sqrt{(x_t - \lambda_{x_2})^2 + (y_t - \lambda_{y_2})^2} \end{bmatrix}$$

Next we took the first derivative Jacobian of these models. From the time system model noise propagation by linearization, we obtained the F_t and W_t matrices while from the observation noise approximation model we obtained the H_t matrix. Next, we updated the covariance matrix sigma bar using the equation below:

$$\overline{\Sigma}_{t+1} = F_t \hat{\Sigma}_t F_t^T + W_t Q W_t^T$$

We also calculated the Kalman gain:

$$K_t = \overline{\Sigma}_{t^-} H^\top \left(H \overline{\Sigma}_{t^-} H^\top + R \right)^{-1}$$

Calculated the sigma hat covariance matrix using kalman gain, H_t and sigma bar:

$$\Sigma_{t^+} = \left(I - K_t H \right) \Sigma_{t^-}$$

Calculated the error:

$$\left(y_t - H \bar{x}_{t^-} \right)$$

and then updated the state hat using the state bar, kalman gain and error:

$$\hat{x}_{t+} = \bar{x}_{t^-} + K_t \left(y_t - H \bar{x}_{t^-} \right)$$

We plotted the trajectories of the ground truth state, sensor output and of the EKF trajectories. All these functions were plotted using the simulate_path_noise_with_EKF function which looped until the simulation time and at each time step updated the location and trajectory of ground truth state, sensor output and EKF class (called by simulate_EKF function).

Results:
We plotted trajectories comparing the ground truth and EKF trajectory at different initial states. From the results, we found that the EKF trajectory was very close to the ground truth results. There was some differences due to the error from the slip in the wheels (omega_std) not being adjusted by bias term. However, when the omega_std was set to zero, both our EKF and ground truth trajectories were almost identical. We also plotted many curved trajectories by changing the omega of the either the right wheel or the left wheel. Since the omegas were not the same for the two wheels, it created a curved path. Again our EKF trajectory was very close to our ground truth trajectory for the curved paths. Lastly, we also plotted trajectories with no knowledge of the initial state. To do this, we initialized our state bar (from time propagation model) and state hat (from observation model) to random states while tracking our ground truth. We found that our EKF graph was able to get close to our ground truth and showed signs of convergence. To characterize the performance of our EKF state estimator we compared the error values and also kept track of our covariance matrices sigma bar and sigma hat. We noticed that our errors kept decreasing showing signs of convergence while our sigma bar was always higher than sigma hat and there was a decrease in the uncertainty or values of the diagonal elements. To improve our EKF model, we could add bias term in our state to offset the gyro noise (since gyro sensor does not have zero additive noise). Our EKF model was computationally efficient to estimate the state of our robot but another state-estimator algorithm that we could have used is the UKF (unscented kalman filter) model which does not require the computation of a Jacobian and requires just a non linear state model.