Zenuity Sensor Fusion Task : Object Tracking

Jay Patravali

August 2, 2018

Problem Summary

The problem is of tracking a submarine vessel using sonar sensor mounted on a boat. The sonar sensor records noisy measurements with standard deviation ≈ 0.1 m in range, standard deviation ≈ 3 degrees in angle. The vehicle initial belief state is taken as x and y equal to first groundtruth reading and zero x and y velocities. Submarine has a constant velocity motion model.

1 EKF Equations and Matrices

Problem can be addressed using an Extended Kalman Filter. Equations for EFK are.

Model,

$$x_k = f(x_{k-1}) + v_k \tag{1}$$
$$z_k = h_k(x_k) + w_k \tag{2}$$

Predict,

$$\hat{x}_k = F x_{k-1} \tag{3}$$
$$P_k = F_k P_{k-1} F_k^T + Q_k \tag{4}$$

Update,

$$K = P_k H_k^T (H_k P_k H_k^T + R)^{-1} \tag{5}$$
$$\hat{x}_k = \hat{x}_k + K(z_k - h_k(\hat{x}_k)) \tag{6}$$
$$P_k = (I - K H_k) P_k \tag{7}$$

where matrices expand as ,

$$x_{k-1} = \begin{bmatrix} x \\ y \\ vx \\ vy \end{bmatrix}$$

$$F_k = \begin{bmatrix} 1 & 0 & Ts & 0 \\ 0 & 1 & 0 & Ts \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_{k-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Q_k = q \begin{bmatrix} Ts^3/3 & 0 & Ts^2/2 & 0 \\ 0 & Ts^3/3 & 0 & Ts^2/2 \\ Ts^2/2 & 0 & Ts & 0 \\ 0 & Ts^2/2 & 0 & Ts \end{bmatrix}$$

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$h_k(\hat{x}_k) = \begin{bmatrix} \hat{x}_k + n_1 \\ \hat{y}_k + n_2 \end{bmatrix}$$

$$z_k(\hat{x}_k) = \begin{bmatrix} meas\_range * cos(meas\_angle) \\ meas\_range * sin(meas\_angle) \end{bmatrix}$$

$$H_k = \frac{\partial h}{\partial (x, y, vx, vy)} = \begin{bmatrix} \frac{\partial h}{\partial x}\hat{x}_k & \frac{\partial h}{\partial x}\hat{y}_k & 0 & 0 \\ \frac{\partial h}{\partial y}\hat{x}_k & \frac{\partial h}{\partial y}\hat{y}_k & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$R = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_a^2 \end{bmatrix}$$

and these variables are described as follows ,

- v_k is zero-mean gaussian process noise and w_k is measurement noise of zero-mean gaussian distribution.
- vx and vy is velocities in x and y coordinates
- P_k is covariance of state, and P_{k-1} is initial covariance
- F_k is state transition matrix
- Q_k is covariance matrix of process noise, q is a tunable scaling factor.
- R is measurement noise covariance matrix, with σ_r^2 and σ_a^2 as variance in range and angle for a zero mean gaussian distribution.
- z_k is the observed state derived from the measurements obtained from the sensor. In this case range and azimuth angle is transformed into X and Y states.
- $h_k(\hat{x}_k)$ is observation function that describes predicted state in X and Y with added gaussian noise – n_1, n_2 that are drawn from $N(0, \sigma_r^2)$ and $N(0, \sigma_a^2)$ respectively.

– H_k is jacobian of observation function $h_k(\hat{x}_k)$ evaluated at the current estimate of the state from the prediction step.

– $\hat{x}_k$ is mean of distribution of state estimated by the EKF.

– K is the Kalman gain.

2 Generating sensor measurements

Given the ground truth trajectory, we can assume the boat to be stationary at the origin, and calculate range and angle from this frame of reference. We generate 100 such unique sequences each containing 500 measurements for corresponding ground truth value. The code and visualization is given in $scripts/generate_measurement.py$.

$$\begin{aligned} range\_val &= vals\_x^2 + vals\_y^2 \\ noisy\_r &= range\_val + normal\_distribution(0, \sigma_r) \\ azimuth &= math.atan(vals\_y/vals\_x) \\ noisy\_a &= azimuth + normal\_distribution(0, \sigma_a) \end{aligned} \tag{8}$$

Where, normal_distribution() is added zero-mean gaussian noise. $vals\_x$ and $vals\_y$ is x and y coordinates from ground truth data. Here, $noisy_r$ is range with added zero-mean gaussian noise with std dev 0.1m $noisy_a$ is range with added zero-mean gaussian noise with std dev $3 * PI/180$ radians. A single sequence is illustrated in Fig. 1.

To run the code, just type `python generate_measurements.py` !

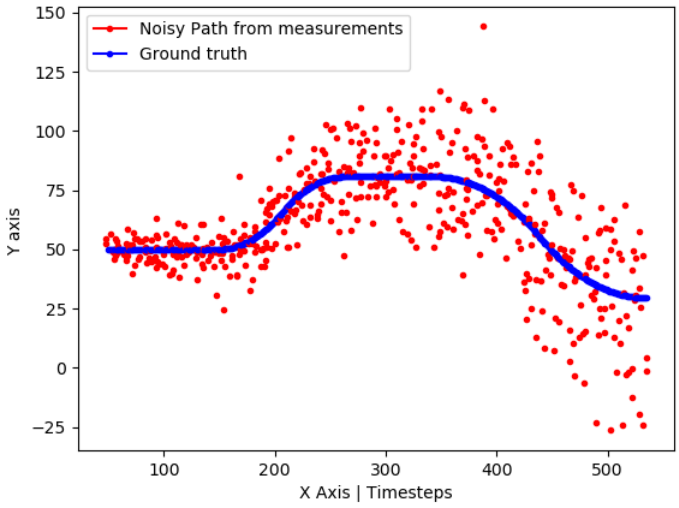

Figure 1: Path reconstructed from generated noisy measurements

3 Running the EFK and Results

Code is setup to run for 100 EFK samples and plot results stored in `results` dir.

- scripts are contained in `scripts` dir
- all data contained in `data` dir
- run `python main.py -q_list 0.01 0.1 1`

Assumptions:

1. The boat is assumed to be stationary at the origin (0,0). Hence the the noisy measurements generated assuming the boat to be at the origin.
2. Initial state belief covariance is taken as an identity matrix.

Running the EKF, we plot the trajectories of the submarine vessel as given in Fig. 2, 3,4. The x-axis represents the X point coordinates and similarly, Y-axis represents the Y point coordinates in meters. The prediction state trajectory is plotted from states after the prediction step!

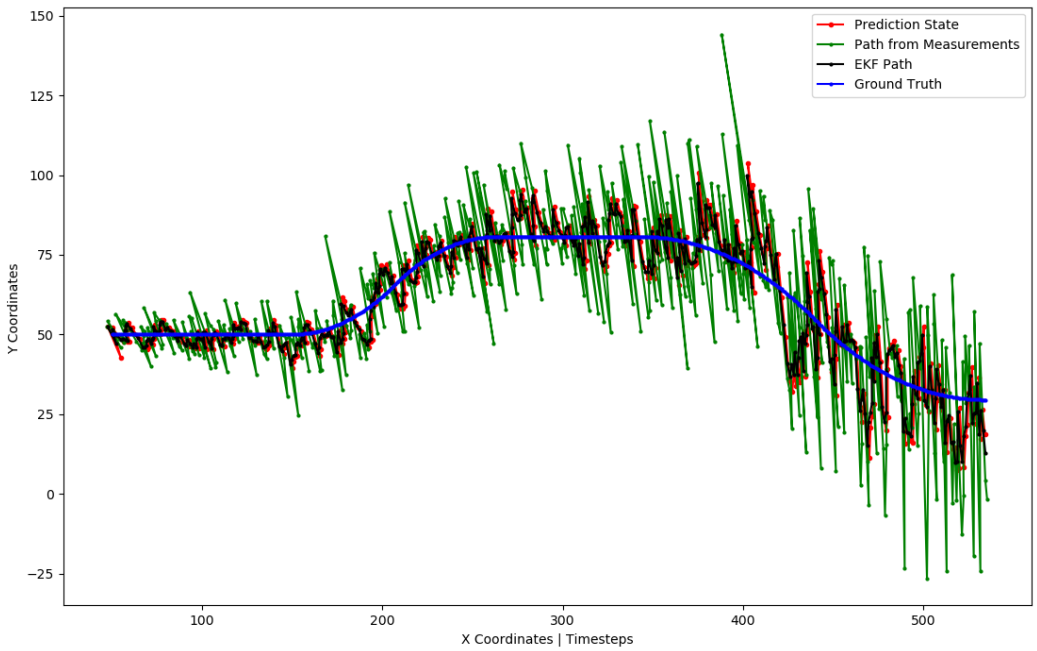

Figure 2: Results: Vessel Trajectory for q value = 0.01

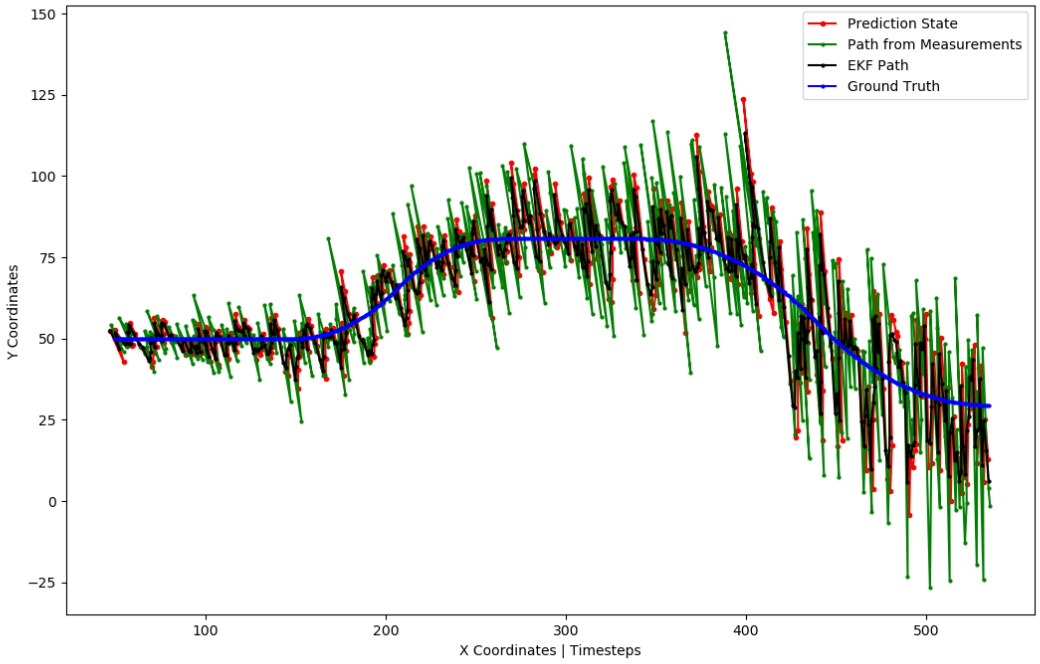

Figure 3: Results: Vessel Trajectory for q value = 0.1

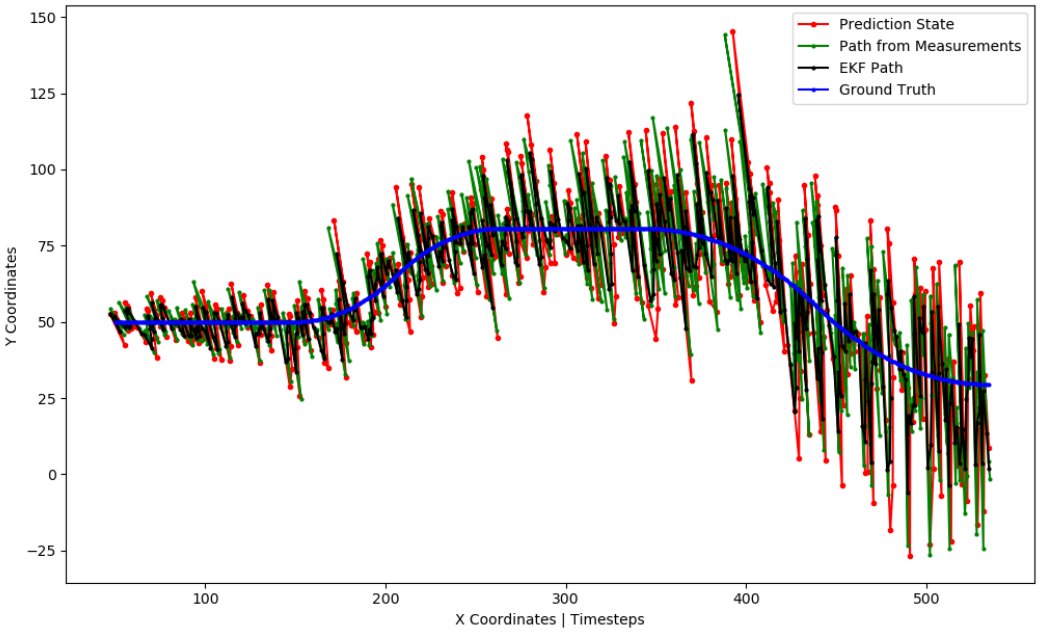

Figure 4: Results: Vessel Trajectory for q value = 1

4 100 Monte-Carlo Samples and RMSE Error

Run the code for three different q_factor values and compute error statistics. Root mean squared is a metric for accuracy here.

$$\text{RMSE} = \sqrt{\frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{n}}$$

Where, y_t is the ground truth data and $\hat{y}_t$ is the corresponding estimated data.

For each q scaling factor, we get RMSE values in x,y, vx and vy for a 500 timestep sequence computed over 100 EKF samples, which are then plotted. These plots are shown in Fig. 5 for three different q scaling values.

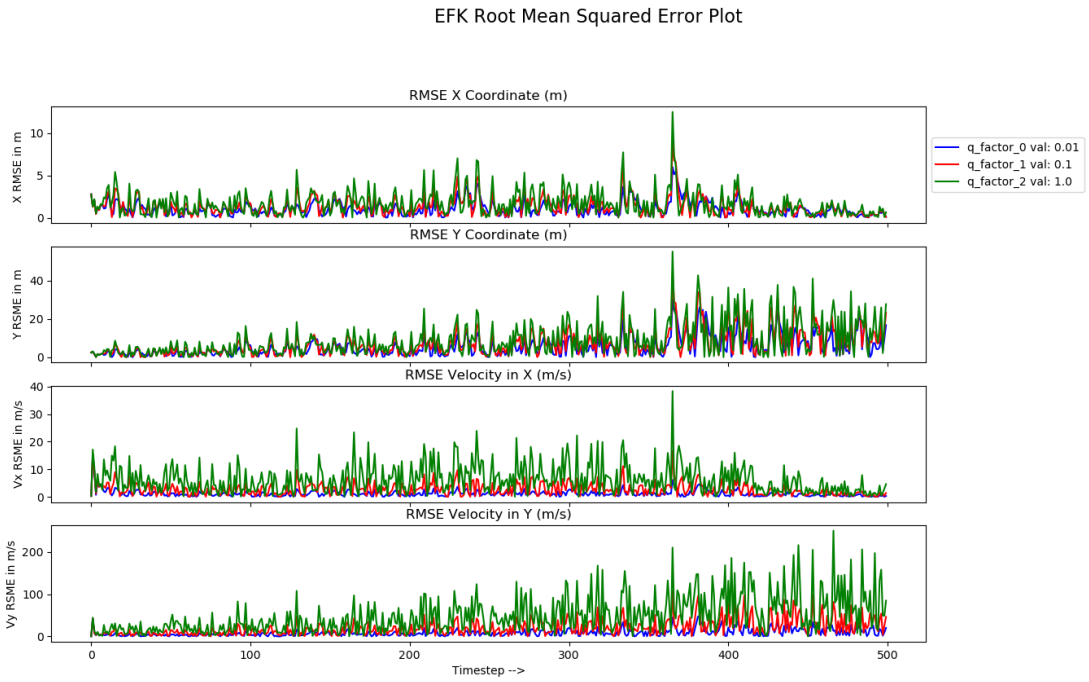

Figure 5: RMSE Errors in x,y and x,y velocities computed over 100 EKF samples.

5 Tuning the q factor

Three values of q are chosen, which are passed as a list arguement to the main.py file. We can see from the Fig. 5, that with increase in a scaling value, the RMSE error increases. We can thus infer that with increase in process noise, the error in performance of the EKF increases.

6 References

1. Probabilistic Robotics, Sebastian Thrun, Wolfram Burgard, Dieter Fox.

2. Lecture Notes on EKF, Adam Hoover, http://cecas.clemson.edu/ ahoover/ece854/lecture-notes/lecture-ekf.pdf