ROB GY-6213 Robot Localization and Navigation Project III

Yash Khanolkar yak9424 N12169803

24 April 2024

1 Introduction

In Project 1, we applied an Extended Kalman Filter(EKF) in order to perform State Estimation if a Micro-aerial Vehicle.On the other hand in Project 2, we made use of Projective Geometry and Optical flow for Vision-based pose and velocity Estimation of our Robot. However, EKF is only accurate if the data provided to us is linear. In case of non-linear data, the EKF fails in it's approximation. Here, the Unscented Kalman Filter(UKF) can be applied. UKF is a numerical method for approximating the joint distribution of a gaussian random variable $x \epsilon R^n$ and a nonlinear transformation h of it by choosing a set of sigma points and considering the mean and variance at each iteration rather than a single value for the whole process. Though, this method is more expensive than EKF,it is fairly accurate in approximating non-linearities and jacobian matrices and derivative equations can be avoided.

2 Problem Statement

The aim of this project is to develop the Unscented Kalman Filter to combine the data which was used in Project 1 and the vision-based pose and velocity estimation performed in Project 2.This project is divided into 2 parts, where in the 1st part, we will use the pose estimates to determine the position and orientation while in the 2nd part, we will use the velocity from the Optical Flow. The pose is in the world frame in Part 1, so the same linear measurement model applies as used in Project 1. However, the velocity from the Optical Flow is derived in the camera frame due to which the model will be non-linear. This should be calculated with respect to the body frame as the UKF is implemented. Keeping this in mind, we need to achieve accurate results for non-linear model using UKF.

3 Procedure

In this project, we are dealing with a non-linear model where non-additive noise is prevalent. Hence, the function will have both state x and noise q as arguments

$$y = h(x, q), \quad x \sim N(\mu, \epsilon), \quad q \sim N(0, Q), \quad \begin{pmatrix} x \\ y \end{pmatrix} = (\begin{pmatrix} \mu \\ m_U \end{pmatrix}, \begin{pmatrix} \epsilon & C_U \\ {C_U}^T & S_U \end{pmatrix})$$

Taking the dimension of x and q to be n and n_q respectively, the overall dimension $n' = \text{n} + n_q$

Now as we are dealing with non-additive noise, the mean , covariance and the random variable will contain augmented values of both x and q where they are matrices of size 15 and 12 respectively.

$$\text{Augmented Random variable } x_{aug} = \begin{pmatrix} x \\ q \end{pmatrix}, \quad \text{mean } \mu_{aug} = \begin{pmatrix} \mu \\ 0 \end{pmatrix}$$

$$\text{covariance } \epsilon_{aug} = \begin{pmatrix} \epsilon & 0 \\ 0 & Q \end{pmatrix}$$

Firstly, we will implement the **Prediction Part** of the UKF which will be same for both the parts of the project, be it position and Orientation, or Velocity. We have,

$$x_t = f(x_{t-1}, u_t, q_t), \quad x \sim N(\mu, \epsilon), \quad q_t \sim N(0, Q_t)$$

In step 1, we need to compute the sigma points on which our UKF will be performed. We use the Chol function in MATLAB to determine the root of the covariance ϵ_{aug} which plays a major role in this step as we need to find the sigma points on both sides of the current mean.

Step 1: Compute Sigma Points

Augment the state $\boldsymbol{x}_t$ as $\boldsymbol{x}_{aug} = \begin{pmatrix} \boldsymbol{x}_{t-1} \\ \boldsymbol{q}_t \end{pmatrix}$ (size n / size n_q) and form the sigma points for the augmented random variable

$$\chi^{(i)}_{aug,t-1} = \boldsymbol{\mu}_{aug,t-1} \pm \sqrt{n' + \lambda'} \left[\sqrt{\boldsymbol{P}_{aug}} \right]_i \qquad \boldsymbol{\mu}_{aug,t-1} = \begin{pmatrix} \boldsymbol{\mu}_{t-1} \\ \boldsymbol{0} \end{pmatrix} \qquad \boldsymbol{P}_{aug} = \begin{pmatrix} \boldsymbol{\Sigma}_{t-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q}_t \end{pmatrix}$$

$\lambda' = \alpha^2(n' + k) - n'$ α and k determine sigma points spread

In Step 2, the $2n' + 1 = 55$ sigma points which we got are propagated through the non linear function f so that we can derive all the state parameters of our robot for that time instant, which is stored in our function.

Step 2: Propagate Sigma Points through the nonlinear function f

$$\chi^{(i)}_t = f(\chi^{(i),x}_{aug,t-1}, u_t, \chi^{(i),q}_{aug,t-1}) \qquad i = 0, \ldots 2n' \qquad \chi^{(i)}_{aug,t-1} = \begin{bmatrix} \chi^{(i),x}_{aug,t-1} \\ \chi^{(i),q}_{aug,t-1} \end{bmatrix}$$

size n (for $\chi_t^{(i)}$); size n (for $\chi^{(i),x}_{aug,t-1}$); size n_q (for $\chi^{(i),q}_{aug,t-1}$); size n (for $\chi^{(i),x}_{aug,t-1}$); size n_q (for $\chi^{(i),q}_{aug,t-1}$)

In Step 3, we predict the mean and covariance of the sigma point based on the parameters stored in the function. This mean and covariance is stored as the estimated mean and covariance based on which we update them into the current mean and covariance and this process continues for all the iterations.

Step 3: Compute the mean and covariance

$$\bar{\mu}_t = \sum_{i=0}^{2n'} W_i^{(m)\prime} \chi_t^{(i)} \qquad \bar{\Sigma}_t = \sum_{i=0}^{2n'} W_i^{(c)\prime} \left(\chi_t^{(i)} - \bar{\mu}_t\right)\left(\chi_t^{(i)} - \bar{\mu}_t\right)^T \leftarrow \text{Prediction}$$

$$W_0^{(c)\prime} = \frac{\lambda'}{n' + \lambda'} + (1 - \alpha^2 + \beta) \quad W_i^{(c)\prime} = \frac{1}{2(n' + \lambda')}$$

$$W_0^{(m)\prime} = \frac{\lambda'}{n' + \lambda'} \qquad W_i^{(m)\prime} = \frac{1}{2(n' + \lambda')} \qquad i = 1, \ldots 2n'$$

The next step which we perform is the **Update Part** of our UKF which will differ for part 1 and 2. Now while calculating the position and the orientation information of our robot the Linear Additive vision pose update is the same as that performed in Kalman Filter in Project 1. It follows a fairly easy list of steps.

$$z_t = CX + \eta, \qquad \eta \sim N(0, R)$$

$$\mu = \mu_t + K_t(z_t - C\bar{\mu})$$

$$\epsilon_t = \bar{\epsilon}_t - K_t C \bar{\epsilon}_t$$

$$K_t = \bar{\epsilon}_t C^T (C \bar{\epsilon}_t C^T + R)$$

The values calculated from this process is our current mean and covariance which is used in the next iteration. For updating the velocity information of the robot, the process becomes a bit lengthy because the we need to express the measurement model in the camera frame with respect to the world while our linear and angular velocity values are present in the body frame with respect to the world frame . Hence, we need to apply Robot Kinematic as our measurement model is not linear additive in this case.

For the update part, we have

$$z_t = g(x_t, v_t) \qquad v_t \sim N(0, V_t)$$

The number of sigma point in our case is $n^{''} = 15$ as we neglect the n_v terms.

Step 1: Compute Sigma Points

Augment the state x_t as $x_{aug} = \begin{pmatrix} x_t \\ v_t \end{pmatrix}$ (x_t: size n, v_t: size n_v) and form the sigma points for the augmented random variable

$$\chi_{aug,t}^{(i)} = \mu_{aug,t} \pm \sqrt{n'' + \lambda''} \left[\sqrt{\Sigma_{t,aug}}\right]_i \qquad \mu_{aug,t} = \begin{pmatrix} \bar{\mu}_t \\ 0 \end{pmatrix} \qquad \Sigma_{t,aug} = \begin{pmatrix} \bar{\Sigma}_t & 0 \\ 0 & V_t \end{pmatrix}$$

($\bar{\mu}_t$, $\bar{\Sigma}_t$: Obtained at prediction stage)

$$\lambda'' = \alpha^2(n'' + k) - n'' \qquad \alpha \text{ and } k \text{ determine sigma points spread}$$

In the 1st step the the update part, we use the mean and covariance values obtained in the prediction stage to get a total of 31 sigma points. Now we pass the sigma points through the non-linear function g but for that we need to bring our measurement model in the required frame.

$$z_t = g(x_t) + v_t = {}^C\dot{p}_C^W = g(x_2, x_3, {}^B\omega_B^W) + v_t \qquad v_t \sim N(0, V)$$

$$\mu_t = \mu_t + K_t(z_t - z_{\mu,t})$$

$$z_t = {}^C\dot{p}_C^W, z_{\mu,t} = g_\mu(x_2, x_3, {}^B\omega_B^W)$$

$${}^B\dot{p}_B^W = l(x_2, x_3)$$

Hence, the measurement model can be found out by,

$$g_\mu = R_B^C l(x_2, x_3) - R_B^C S(r_{BC}^B) R_C^B {}^C\omega_C^W$$

The change of frame for the velocity can be performed using the adjoint method given by,

$$\begin{bmatrix} {}^C\dot{p}_C^W \\ {}^C\omega_C^W \end{bmatrix} = \begin{bmatrix} R_B^C & -R_B^C S(r_{BC}^B) \\ 0 & R_B^C \end{bmatrix} \begin{bmatrix} {}^B\dot{p}_B^W \\ {}^B\omega_B^W \end{bmatrix}$$

After we determine z_t, we can perform step 2.

Step 2: Propagate Sigma Points through the nonlinear function g

$$Z_t^{(i)} = g(\chi_{aug,t}^{(i),x}, \chi_{aug,t}^{(i),v}) \qquad i = 0, \dots 2n'' \quad \chi_{aug,t-1}^{(i)} = \begin{bmatrix} \chi_{aug,t-1}^{(i),x} \\ \chi_{aug,t-1}^{(i),v} \end{bmatrix}$$

size n

size n_v

Finally, we perform the final step where we find the predicted mean , covariance and cross-covariance from the measurement model and the sigma points derived above.

Step 3: Compute the predicted mean, predicted covariance of the measurement, and predicted cross-covariance

$$z_{\mu,t} = \sum_{i=0}^{2n''} W_i^{(m)''} Z_t^{(i)} \qquad \text{Update}$$

$$C_t = \sum_{i=0}^{2n''} W_i^{(c)''} \left(\chi_{aug,t}^{(i),x} - \bar{\mu}_t\right)\left(Z_t^{(i)} - z_{\mu,t}\right)^T \qquad S_t = \sum_{i=0}^{2n''} W_i^{(c)''} \left(Z_t^{(i)} - z_{\mu,t}\right)\left(Z_t^{(i)} - z_{\mu,t}\right)^T$$

$$W_0^{(m)''} = \frac{\lambda''}{n'' + \lambda''} \qquad W_i^{(m)''} = \frac{1}{2(n'' + \lambda'')}$$

$$W_0^{(c)''} = \frac{\lambda''}{n'' + \lambda''} + (1 - \alpha^2 + \beta) \qquad W_i^{(c)''} = \frac{1}{2(n'' + \lambda'')}$$

We conclude by calculating the filter gain, filtered state mean and covariance, conditional to the measurement.

$$\mu_t = \bar{\mu_t} + K_t(z_t - z_{\mu,t})$$

$$\epsilon_t = \bar{\epsilon}_t - K_t S_t K_t^T$$

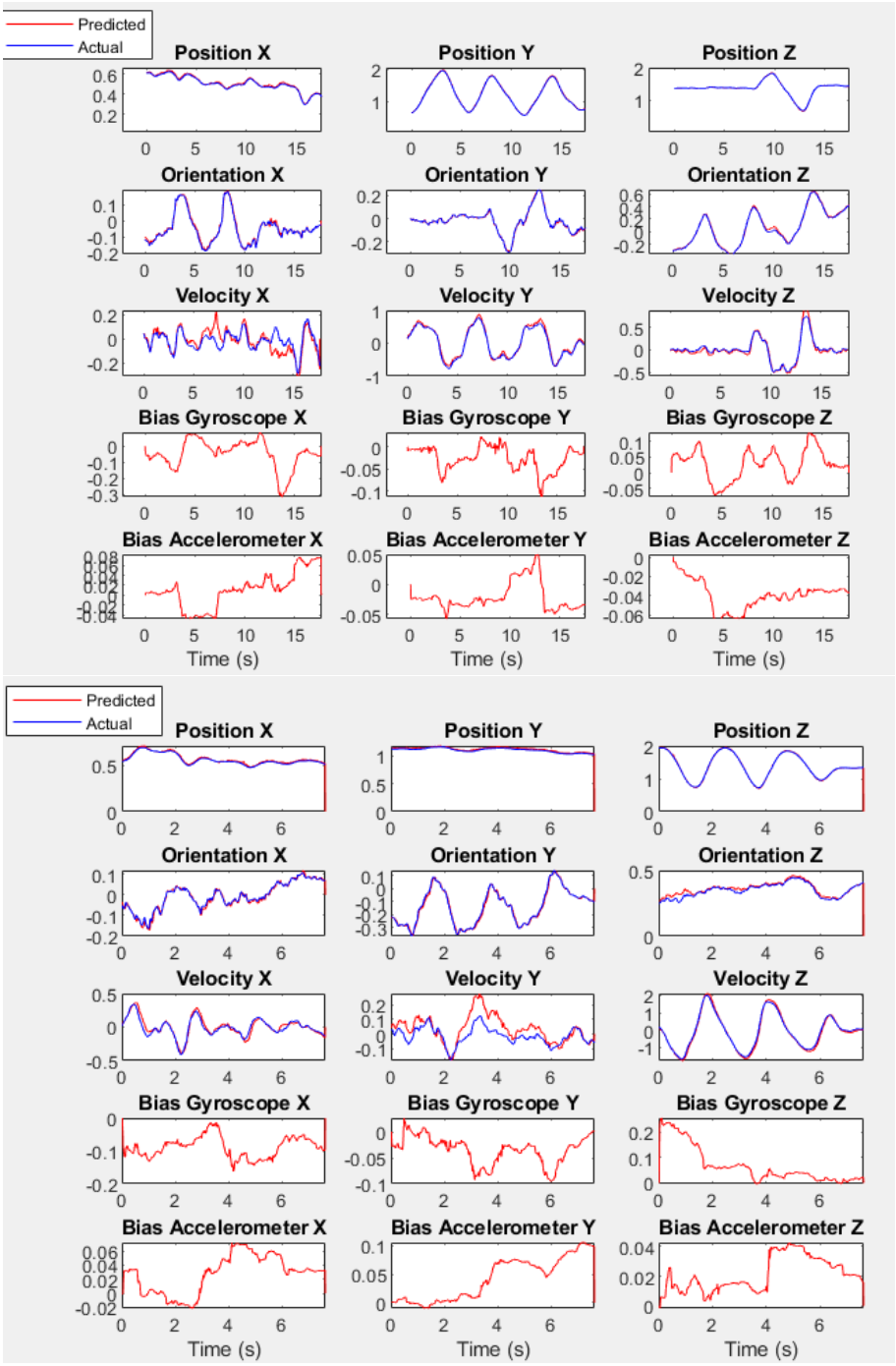
$$K_t = C_t S_t^{-1}$$

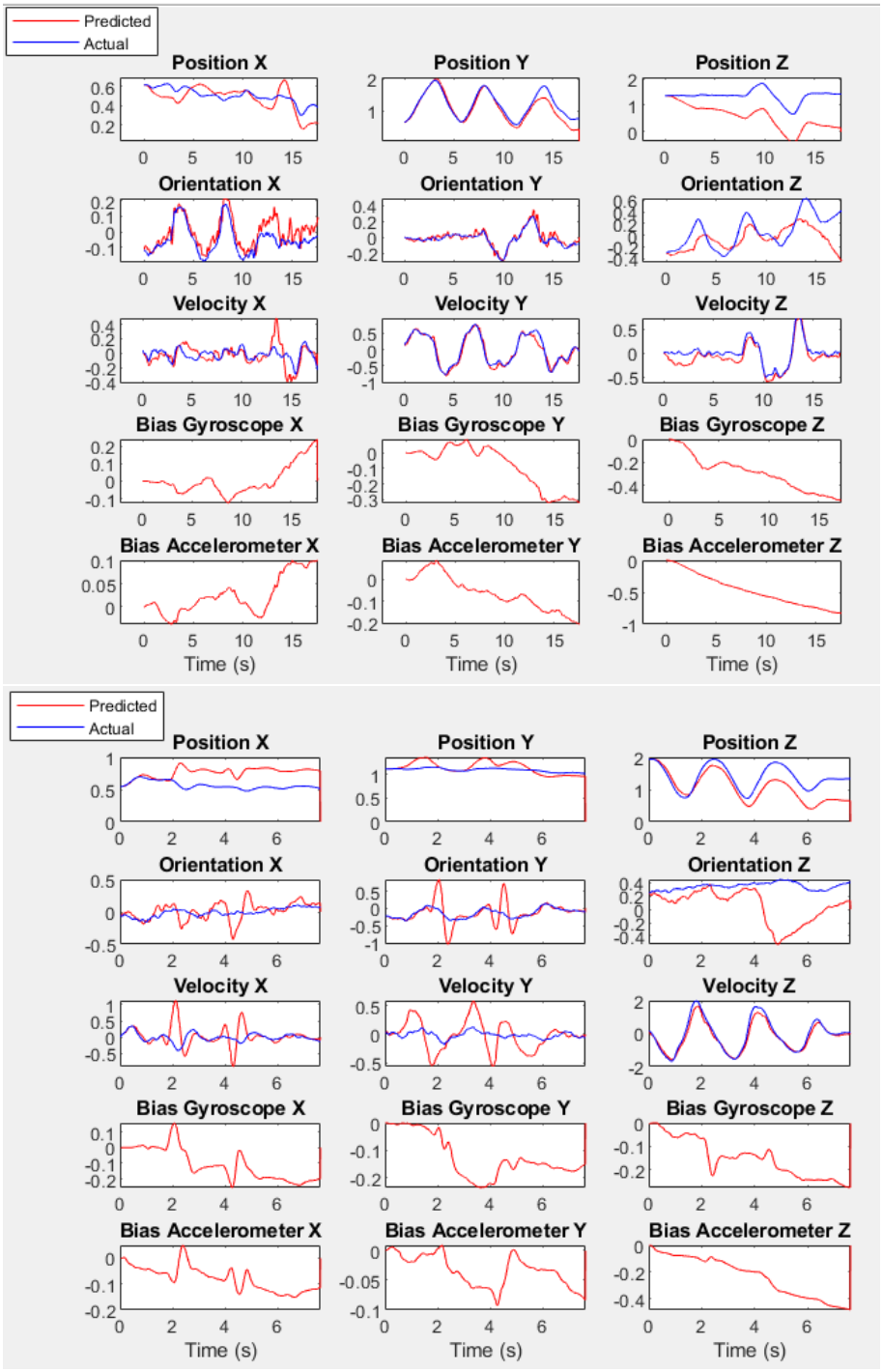
4 Code

In the Kalman Filter 1 main file, we have defined the time period and extracted the values for measurement model, calling both the prediction and update functions and saving the mean and covariance values in the memory. In the prediction stage, we have first set the noise variable and other parameter and called the Cholesky function to extract the root of the covariance. After this initial step, we have started our prediction stage where we have first computed the sigma points from the augmented mean and variance. These points are propagated through the non-linear function which needs a load of calculations involving G, R and g to give us the orientation, position, velocity and noise parameter out of only the first 3 are important to us. the position and orientation thus achieved is passed through the summation of weights of mean and covariance to get the predicted mean and covariance. This is useful in the update step of the UKF. Since the measurement model for rotation and orientation is considered as linear, hence the predicted mean and variance is easily found by just applying the formula setting the noise parameter R.

In Part 2, the velocity information needs to be predicted and the the main file and the prediction step is performed exactly the same way as in Part 1. However in the update part, We considered the estimated mean and covariance values calculated from the prediction step and used these values to compute the sigma points as the measurement model is not linear. Also while propagating sigma points through the non-linear function, we have to keep in mind that the information given is in the body frame with respect to the world frame which we have to align in the camera frame with respect to the world. We executed basic kinematics and adjoint matrix to achieve our result. Finally, in the last step we have found out the predicted mean, covariance and the cross-covariance which is used to get the final predicted mean and covariance, which is saved in the savedstates matrix.

5 Results


Predicted
Actual
Position X
Position Y
Position Z
Orientation X
Orientation Y
Orientation Z
Velocity X
Velocity Y
Velocity Z
Bias Gyroscope X
Bias Gyroscope Y
Bias Gyroscope Z
Bias Accelerometer X
Bias Accelerometer Y
Bias Accelerometer Z
Time (s)
Time (s)
Time (s)
Predicted
Actual
Position X
Position Y
Position Z
Orientation X
Orientation Y
Orientation Z
Velocity X
Velocity Y
Velocity Z
Bias Gyroscope X
Bias Gyroscope Y
Bias Gyroscope Z
Bias Accelerometer X
Bias Accelerometer Y
Bias Accelerometer Z
Time (s)
Time (s)
Time (s)


Predicted
Actual
Position X
Position Y
Position Z
Orientation X
Orientation Y
Orientation Z
Velocity X
Velocity Y
Velocity Z
Bias Gyroscope X
Bias Gyroscope Y
Bias Gyroscope Z
Bias Accelerometer X
Bias Accelerometer Y
Bias Accelerometer Z
Time (s)
Time (s)
Time (s)
Predicted
Actual
Position X
Position Y
Position Z
Orientation X
Orientation Y
Orientation Z
Velocity X
Velocity Y
Velocity Z
Bias Gyroscope X
Bias Gyroscope Y
Bias Gyroscope Z
Bias Accelerometer X
Bias Accelerometer Y
Bias Accelerometer Z
Time (s)
Time (s)
Time (s)

In the first two images, which are for position and Orientation of Dataset 1 and Dataset 4 respectively, our graph follows the expected graph accurately. This shows how the UKF successfully uses sigma points to measure states accurately.

In the next 2 images, which show the velocity of Dataset 1 and Dataset 4 respectively, the calculated graph does not exactly follow the expected graph due to noise uncertainities and due to recurring errors from the Optical Flow. This shows us that despite our best efforts to tune the noise the UKF is not foolproof for all cases.

6 Conclusion

We have demonstrated how Unscented Kalman Filter behaves way better than conventional Kalman filter when non-linear model is considered.Though it requires higher computation, the fact that we are not using jacobian and differentiation reduces the computation time in MATLAB significantly. From the graphs we can see that the precision in following the predicted output is unmatched in UKF. The need to switch frames did result in the noise uncertainties in MATLAB but practically it will turn out to be much more accurate and efficient.

Also, this project highlights the advantages of Matlab in effortless decoding, running and displaying accurately, all the aspects and details of the path and pattern of quadrotor motion. Lastly, implementing UKF taught us that learning the previous states and results through different sigma points helps the machine to become more intelligent than capturing only the previous state and moves us a step forward in precisely controlling and mapping the path undertaken by our robot

7 References

1) *Probabilistic Robotics - Sebastian Thrun, Wolfram Burgard, and Dieter Fox,Edition 1, MIT Press, 2005*
2)*Bayesian Filtering and Smoothing - Simo Särkkä,3rd Edition, Cambridge University Press, 2001*
3) *Y. Ma, S. Soatto, J. Kosecka, S. Shankar Sastry, "An Invitation to 3D Vision, Springer"*
4) *Professor Loianno's Lecture notes*
5) *https://www.mathworks.com/*