ROBOT LOCALIZATION AND NAVIGATION (ROB-GY 6213)

PROJECT 3 - REPORT

*NAME: **Andrew Devadhason*** *Net ID: **ad7068***

PROJECT -3 TOPIC: UKF-based sensor fusion using IMU and vision-based pose and velocity.

TASK:

The goal of this project is to implement sensor fusion using the Unscented Kalman filter by making use of the IMU data and the vision-based pose and velocities estimated. The vision-based position and orientation is obtained through projective transformation and the linear and angular velocity are calculated through optical flow. Since this system is highly non-linear, the Unscented Kalman filter captures the non-linearity of the model much better to estimate the state of the quadrotor moving in an experimental space.

PROCESS:

The project is split into two parts with the measurement update given by the vision-based pose estimate in the first part and the optical flow-based velocity in the second part. The approach provides insights into two methods of metrics for state estimation and leverages the functionality of the UKF to obtain good state estimates despite noise models.

PART-1:

The first part of the project involves the use of UKF with the non-linear function having non-additive noise. For this case, the state mean and covariance are augmented such that the entity named sigma points are calculated. Sigma points are basically a deterministic approach to capture the spread and characteristics of the mean and covariance of the distribution of the state in a robust manner. This is the primary essence of UKF.

PREDICTION:

- **Parameter Definition:** As a first step the ***uPrev*** and ***covarPrev*** are augmented so that they can be further used for computing the sigma points followed by which the parameters *alpha*, *beta* and *kappa* are defined.

$$\lambda' = \alpha^2(n' + k) - n'$$

    - ***Alpha****: Alpha determines the spread of sigma points around the mean. It is typically set to a small positive value to ensure that sigma points are spread out sufficiently to capture the distribution of the state variables. For our approach, we have set it as $\alpha = 0.001$.*

$$\boldsymbol{\mu}_{aug,t-1} = \begin{pmatrix} \boldsymbol{\mu}_{t-1} \\ \mathbf{0} \end{pmatrix}$$

    - ***Beta****: Beta incorporates prior knowledge about the distribution of state variables. It is typically set to 2 for Gaussian distributions, indicating that the state variables are assumed to be normally distributed.*

$$\boldsymbol{P}_{aug} = \begin{pmatrix} \boldsymbol{\Sigma}_{t-1} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{Q}_t \end{pmatrix}$$

    - ***Kappa****: Kappa is a secondary scaling parameter that depends on the dimensionality of the state space. It controls the weighting of the covariance matrix. We have set kappa to 1.*

- **Computing Sigma Points:** The spread of sigma points *Lambda* is calculated initially. Sigma points are calculated by augmenting the state vector and covariance matrix with process noise. The square root of the augmented covariance matrix (root_PAug) is computed using Cholesky decomposition.

$$\chi^{(0)}_{aug,t-1} = \boldsymbol{\mu}_{aug,t-1}$$

$$\chi^{(i)}_{aug,t-1} = \boldsymbol{\mu}_{aug,t-1} \pm \sqrt{n' + \lambda'}\left[\sqrt{\boldsymbol{\Sigma}_{aug}}\right]_i$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{x}_3 \\ G(\mathbf{x}_2)^{-1}(\omega_m - \mathbf{x}_4 - \mathbf{n}_g) \\ \mathbf{g} + R(\mathbf{x}_2)(\mathbf{a}_m - \mathbf{x}_5 - \mathbf{n}_a) \\ \mathbf{n}_{bg} \\ \mathbf{n}_{ba} \end{bmatrix} = f(x,u,n)$$

$$\chi_t^{(i)} = f(\chi_{aug,t-1}^{(i),x}, u_t, \chi_{aug,t-1}^{(i),n})$$

- **Propagating Sigma Points through Non-linear Function:** For each sigma point, the function computes the predicted state using the non-linear motion model and the process noise. Here, the non-linear function is the same as project-1 through which the calculated sigma points are propagated ***2n+1*** times since the number of sigma points to be calculated is ***2n+1.***
- **Calculating weights for mean and covariance:** The weights for the mean and covariance are calculated as an initial step to further calculate the weighted mean and weighted covariance for the final estimate. The weights for the mean and covariance are calculated based on the UKF parameters alpha, beta, and kappa as shown in the equations.

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

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

$$\bar{\mu}_t = \sum_{i=0}^{2n'} W_i^{(m)} \chi_t^{(i)}$$

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

- **Calculating mean state estimate and covariance estimate:** The mean state estimate and covariance estimate are computed by multiplying the propagated sigma points and the weights respectively in the given fashion. Thus, the final estimates are calculated as part of the prediction step.

UPDATE: The update step is linear and is the same as the implementation followed in the first project.

- **Initializing Matrices**: The measurement matrix C is initialized for linearization. Since the measurement model zt is directly available, there's no need to compute C. The measurement noise covariance matrix R is also initialized**.**
- **Kalman Gain Calculation**: The Kalman gain K is computed using the estimated covariance matrix *covarEst*, the measurement matrix C, and the measurement noise covariance matrix R.
- **Current Covariance Matrix Update**: The updated covariance matrix *covarCurr* is calculated using the Kalman gain.

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

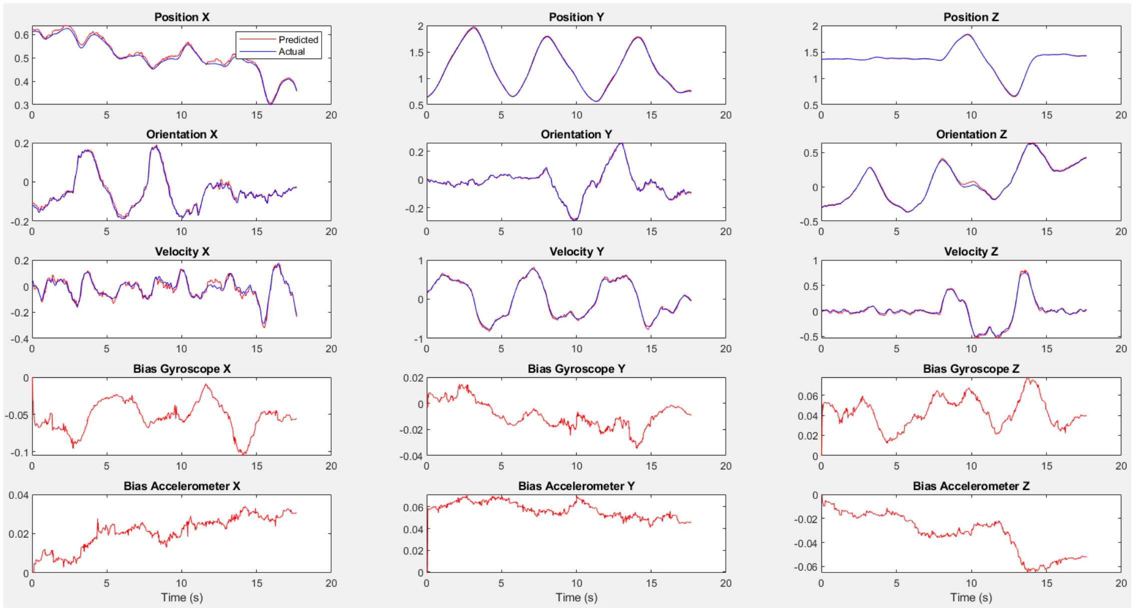
$$\Sigma_t = \bar{\Sigma}_t - K_t\, C\, \bar{\Sigma}_t$$

$$K_t = \bar{\Sigma}_t\, C^T\, (C\, \bar{\Sigma}_t\, C^T + R)^{-1}$$

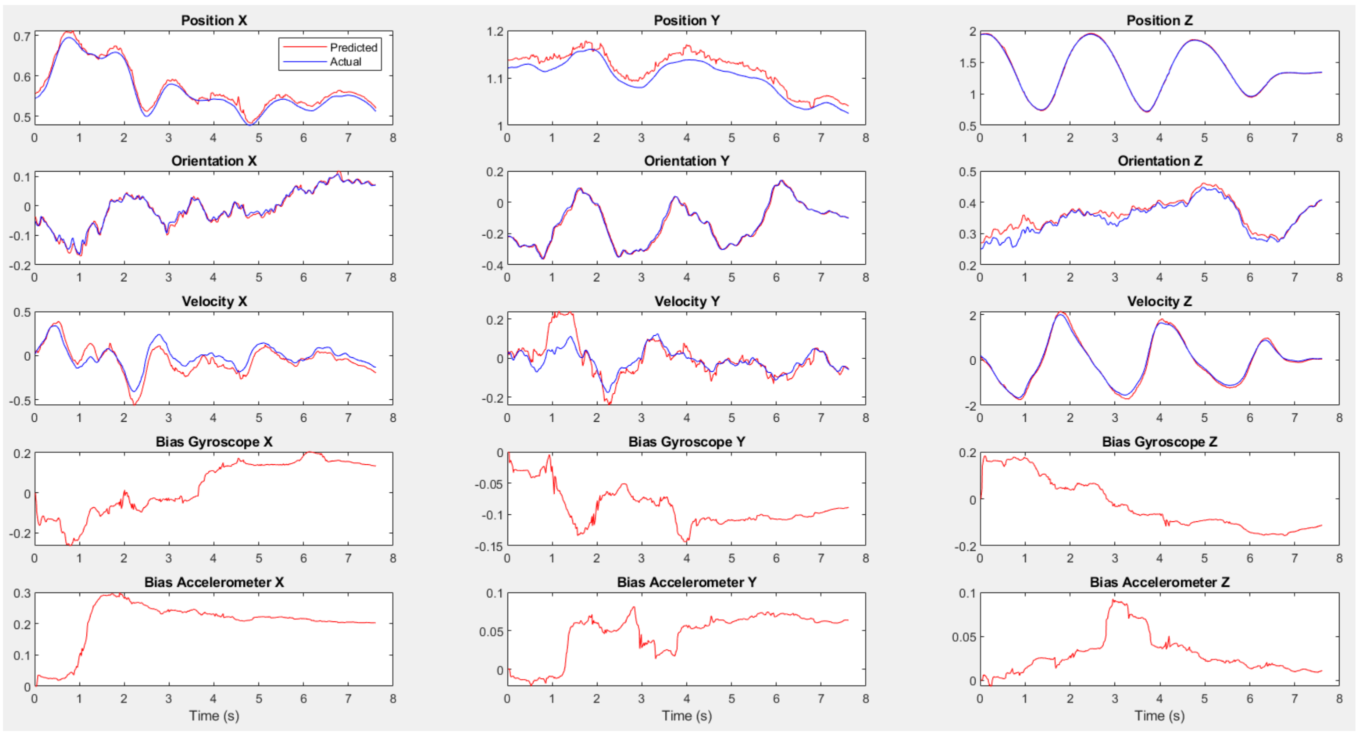
- **Current State Update**: The updated state vector $ucurr$ is computed using the Kalman gain, the difference between the actual measurement zt, the predicted measurement C, and the estimated mean state *uEst.*

PART-1 RESULTS:

- The results of part 1 of the UKF implementation are conclusive with the pre-determined values of Q and R.
- The related graphs based on state estimates calculated through the implementation of UKR with the measurement update from the optical pose and orientation have aligned well with a healthy jitter.
- The UKF has captured the nonlinearity of the given system very well with the selected number of sigma points.



P1: Outcomes of UKF for Dataset-1



P1: Outcomes of UKF for Dataset-4

PART-2:

The second part of the project involves using the velocity from the optical flow as the measurement update. The prediction step remains the same as in part 1. The update step here is non-linear due to the measurement being velocity which is a derived quantity due to the optical flow calculations. Also, the optical flow is defined in the camera frame while the rest are defined in terms of the world and body frames. We will make use of the adjoint matrix to bring everything to the camera frame to keep things simple.

- **Initialise parameters and calculate sigma points**: The initial parameters like alpha, beta, and kappa are initialized to calculate lambda, the spread of sigma points and further the sigma points are calculated. Here, the states need not be augmented since the noise is additive.

  $$\chi_t^{(0)} = \bar{\boldsymbol{\mu}}_t \qquad \chi_t^{(i)} = \bar{\boldsymbol{\mu}}_t \pm \sqrt{n+\lambda}\left[\sqrt{\bar{\boldsymbol{\Sigma}}_t}\right]_i$$

  Hence the value of n becomes 15. Sigma points are computed around the estimated mean state (**uEst**) using Cholesky decomposition to ensure the covariance matrix (**covarEst**) is positive definite.
- **Initialize Rotations and Translations**: Matrices for transformations between coordinate frames are initialized. These include the transformation from camera frame to body frame (**H_C2B**), extraction of rotation matrices (**R_C2B**, **R_B2C**), and initialization of skew-symmetric matrix for translation (**SkewT_C2B**). This is to implement the measurement model which is derived from the adjoint.

  $$\begin{bmatrix} {}^{C}\dot{\boldsymbol{p}}_C^W \\ {}^{C}\boldsymbol{\omega}_C^W \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_B^C & -\boldsymbol{R}_B^C \boldsymbol{S}(\boldsymbol{r}_{BC}^B) \\ \boldsymbol{0} & \boldsymbol{R}_B^C \end{bmatrix} \begin{bmatrix} {}^{B}\dot{\boldsymbol{p}}_B^W \\ {}^{B}\boldsymbol{\omega}_B^W \end{bmatrix}$$
- **Propagate Sigma Points through the Measurement Model**: Each sigma point is propagated through the measurement model to obtain predicted measurements (**Z_t**). Gaussian noise (**vt**) is added to account for measurement noise.

  $$\boldsymbol{g}_{\boldsymbol{\mu}} = \boldsymbol{R}_B^C \boldsymbol{l}(\boldsymbol{x}_2, \boldsymbol{x}_3) - \boldsymbol{R}_B^C \boldsymbol{S}(\boldsymbol{r}_{BC}^B) \;\; \boldsymbol{R}_C^B \, {}^{C}\boldsymbol{\omega}_C^W$$
- **Compute Mean and Covariance Matrices**: Weighted sums of the predicted measurements are computed to obtain the estimated measurement (**Zu_t**). Weighted sums of the predicted covariance and cross-covariance matrices are computed to obtain matrices **C_t** and **S_t**. The value *R* is added to the cross-covariance matrix to introduce sensor noise into the system.

  $$\boldsymbol{z}_{\mu,t} = \sum_{i=0}^{2n} W_i^{(m)} Z_t^{(i)}$$

  $$\boldsymbol{S}_t = \sum_{i=0}^{2n} W_i^{(c)} \left(Z_t^{(i)} - \boldsymbol{z}_{\mu k}\right)\left(Z_t^{(i)} - \boldsymbol{z}_{\mu,t}\right)^T + \boldsymbol{R}_t$$

  $$\boldsymbol{C}_t = \sum_{i=0}^{2n} W_i^{(c)} \left(\chi_t^{(i)} - \bar{\boldsymbol{\mu}}_t\right)\left(Z_t^{(i)} - \boldsymbol{z}_{\mu k}\right)^T$$

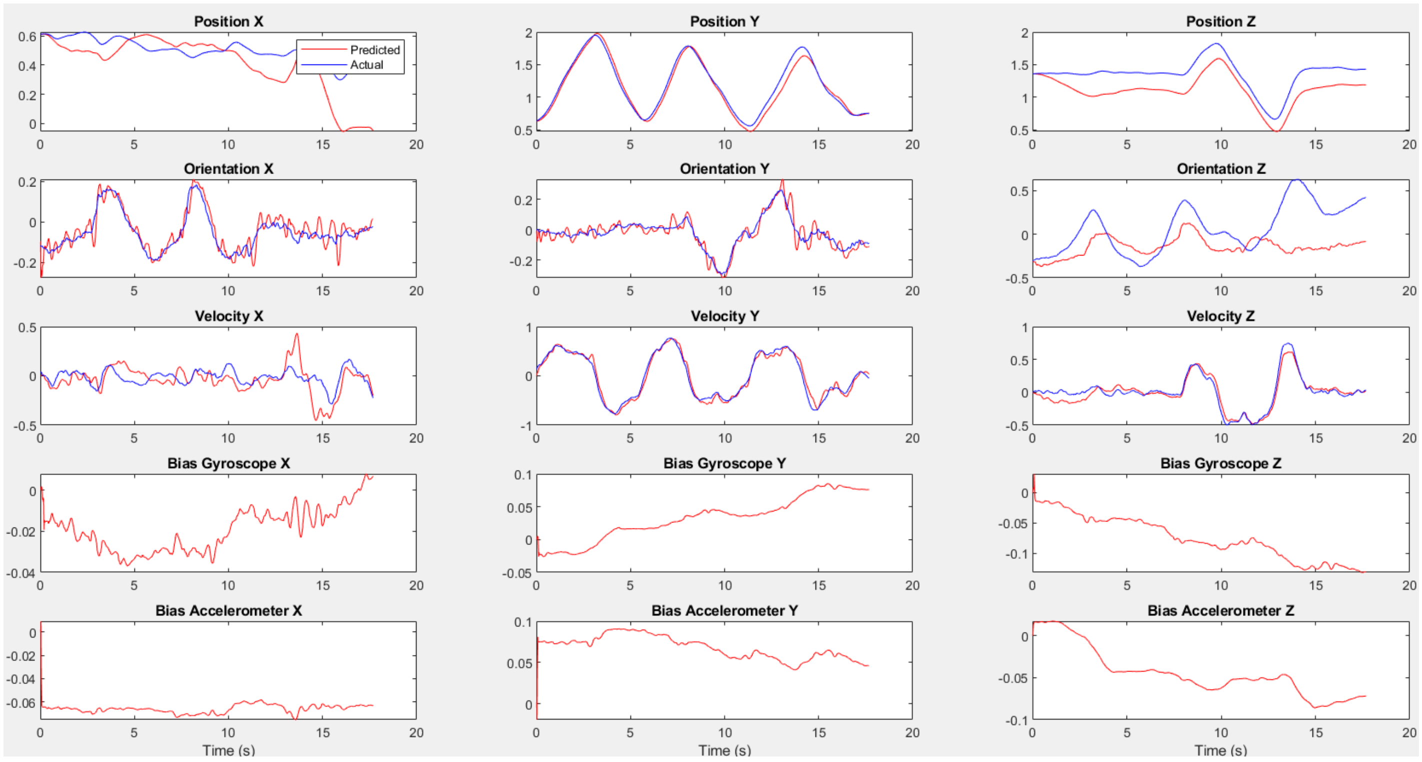
- **Compute *u*Curr and covarCurr**: The Kalman gain is computed using the cross-covariance matrix (**C_t**) and the measurement covariance matrix (**S_t**). The updated state estimate (*u*Curr) is computed using the Kalman gain, the difference between the measured and estimated measurements, and the estimated mean state. The updated covariance matrix (covarCurr) is computed using the Kalman gain and the predicted covariance matrix.

$$\mu_t = \bar{\mu}_t + K_t\,(z_t - z_{\mu,t})$$
$$\Sigma_t = \bar{\Sigma}_t - K_t\,S_t K_t{}^T$$
$$K_t = C_t\,S_t{}^{-1}$$

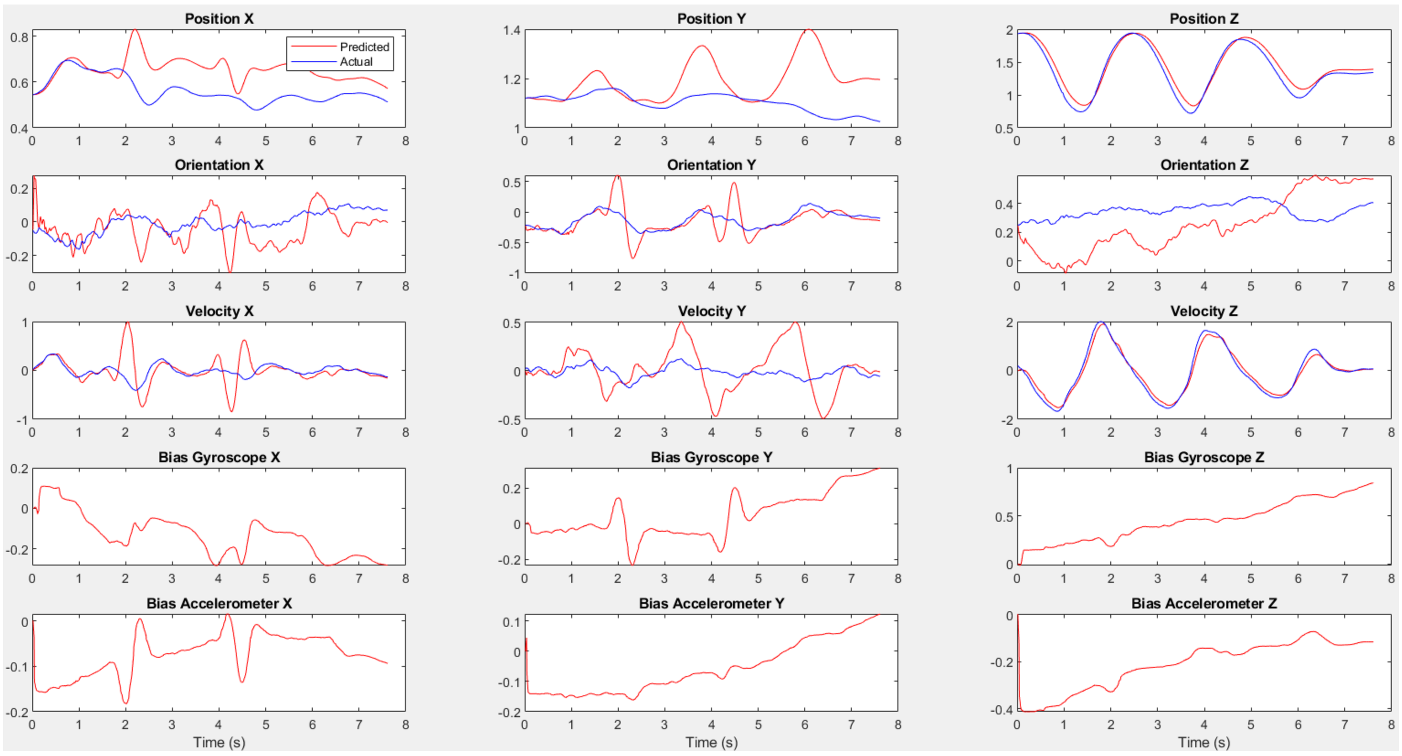
- Thus, the final uCurr and covarCurr are updated and sent back to the main function iteratively.

PART-2 RESULTS:

- The results from part 2 of the project with the measurement update being the velocity from the optical flow have been conclusive.
- The UKF implementation has performed well in capturing the non-linearity in the system although the noise modelled is heavy.
- The values of Q and R are to be modulated accordingly and the spread of sigma points can be modified to meet the characteristics of the model. Since the measurement update is non-linear and the quantities are derived, the UKF has struggled to keep up with the non-linearity but has performed well based on the current implementation.
- The nonlinearity can be observed well in the Part 2 and we can observe the performance characteristics of UKF and its ability to handle such non-linear models.



P2: Outcomes of UKF for Dataset-1



P2: Outcomes of UKF for Dataset-4

CONCLUSION:

The Unscented Kalman filter has proven to be an effective extension of the traditional Kalman filter for non-linear state estimation problems. Based on investigations from the above project we can observe certain aspects of the filter which could improve its performance. The UKF has proven to be robust in handling the degree of nonlinearity of the system and the way it handles noise. Critically tuning the UKF parameters has a major effect on the filter’s performance overall and is visible in the above outcomes.