Project 3: Vision and IMU Fusion with Unscented Kalman Filter

Robot Localization and Navigation
ROB-GY 6213
Version 1.0

Shantanu Ghodgaonkar
Univ ID: N11344563
Net ID: sng8399
Ph.No.: +1 (929) 922-0614

Contents

***Abstract*—This project investigates the application of an Unscented Kalman Filter (UKF) for sensor fusion, combining data from an Inertial Measurement Unit (IMU) and a vision-based system for robot state estimation. The UKF's ability to handle nonlinearities is leveraged to potentially improve accuracy compared to traditional Kalman Filters. The developed UKF framework integrates IMU data with vision-based pose and velocity estimation. The performance is evaluated in two scenarios: using visual pose estimation and using optical flow-derived velocity as measurements. The results are presented, including visualizations of estimated trajectories and quantitative analysis comparing the UKF's performance with potential baselines. The effectiveness of the UKF in handling nonlinearities is discussed, along with potential limitations and areas for future work.**

1. Summary

In essence, this project developed and evaluated an Unscented Kalman Filter (UKF) [7] for sensor fusion. The UKF aimed to improve robot state estimation by combining information from an Inertial Measurement Unit (IMU) and a vision-based system. The project capitalized on the UKF's ability to handle nonlinearities potentially leading to more accurate results compared to traditional Kalman Filters. The core of the project involved designing a UKF framework that could integrate IMU data (leveraged from Project 1 [2]) with vision-based pose and velocity estimates obtained in Project 2 [3]. This framework was then evaluated in two distinct scenarios: one utilizing visual pose estimation as the measurement and another using optical flow-derived velocity. Datasets 1 and 4 were employed for testing, maintaining consistency with the testing approach used in Project 1. The following sections will delve into the methodology employed, implementation details, the obtained results, and a discussion of the findings.

2. Introduction

Sensor fusion plays a crucial role in robot state estimation, particularly when dealing with environments where individual sensors have limitations. This project investigates the application of an Unscented Kalman Filter (UKF) [7] for sensor fusion, aiming to improve the accuracy of robot state estimation by combining information from an Inertial Measurement Unit (IMU) and a vision-based system.

Traditional Kalman Filters [6] are powerful tools for state estimation in linear systems. However, for systems with nonlinearities, the linearization step in KFs can lead to suboptimal performance. The UKF offers an advantage by employing a deterministic approach to handle nonlinearities. It utilizes a carefully chosen set of sigma points to capture the statistical properties of the state estimate and propagates them through the nonlinear process model. This allows the UKF to effectively incorporate nonlinearities into the state estimation process.

This project focuses on developing and evaluating a UKF framework that integrates IMU data (leveraged from Project 1 [2]) with vision-based pose and velocity estimation obtained in Project 2 [3]. The performance of the UKF is evaluated in two scenarios:

- Using visual pose estimation as the measurement.
- Using optical flow-derived velocity as the measurement.

3. Methodology

3.1. Inital Analysis of Given Code

Upon observation of the given skeleton code, the following inferences were made -

- The files titled *UKF_KalmanFilt_Part1.m* and *UKF_KalmanFilt_Part2.m* were akin to the *main* functions in a C Program. They performed the following functionality -

- Load a dataset (1 or4)
- Initialize the values of μ_t and Σ_t and z_t
- Declare a variable *savedStates* to store finally computed state at each dt
- A *for* loop to compute the dt, the predictions $\hat{\mu_t}$, $\hat{\Sigma_t}$ and perform the update using z_t to obtain μ_t and Σ_t using Kalman Gain K_t
- Plot each state at a given time dt

• The file titled *pred_step.m* was the declaration of the function responsible for performing the prediction step of the UKF. It took the following parameters as inputs -

- μ_{t-1} Previous state
- Σ_{t-1} Previous error covariance
- ω Angular velocity control input from IMU
- acc Linear Acceleration control input from IMU
- dt Small time difference

And provided the following outputs -

- $\hat{\mu}_t$ Current state estimate
- $\hat{\Sigma}_t$ Current error covariance estimate

• The file titled *upd_step.m* was the declaration of the function responsible for performing the update step of the UKF. It took the following parameters as inputs -

- z_t Current position and orientation computed using visual data from the camera
- $\hat{\mu}_t$ Current state estimate
- $\hat{\Sigma}_t$ Current error covariance estimate

And provided the following outputs -

- μ_t Current state
- Σ_t Current error covariance

• The file titled *plotData.m* was responsible to plot the state model.

3.2. Dataset structure

Each dataset had three variables inside it -

• $sampledData$ was a $1 \times n$ array of type $struct$ containing the sensor packet at each timestamp. It's components were as follows -

1) is_ready : True if a sensor packet is available, false otherwise
2) t : Timestamp for the sensor packet, different from the Vicon time
3) rpy : orientation of the body
4) omg : Body frame angular velocity from the gyroscope
5) acc : Body frame linear acceleration from the accelerometer
6) img : Undistorted image.
7) id : IDs of all AprilTags detected, empty if no tag is detected in the image
8) $p0$: Corners of the detect AprilTags in the image,
9) $p1$: the ordering of the corners, and the distribution of the tags
10) $p2$: the ordering of the corners, and the distribution of the tags
11) $p3$: the ordering of the corners, and the distribution of the tags
12) $p4$: the ordering of the corners, and the distribution of the tags

• $proj2Data$ was structure, defined as follows -

1) $time$: time of the data
2) $position$: Position of the robot frame with respect to the world
3) $angle$: Orientation of the robot frame with respect to the world
4) $linearVel$: Velocity of the camera frame with respect to the world frame expressed in the camera frame
5) $angVel$: Angular velocity of the camera frame with respect to the world expressed in the camera frame

3.3. Prediction Step for Parts 1 & 2

The prediction step for both parts of this project is the same. This is because the process model for both parts remains the same. Thus, we shall implement the same logic for both parts. Upon observation, we can see that the process model[2] (shown in Eq^n 1) shows that the noise is of the non-additive kind, *i.e.*, the noise is already accounted for within the process model. Thus, we must implement the UKF with Non-Additive Noise.

$$f(x,u,n) = \dot{x} = \begin{bmatrix} x_3 \\ G^{-1}(x_2)(\omega_m - x_4 - n_g) \\ g + R(x_2)(a_m - x_5 - n_a) \\ n_{bg} \\ n_{ba} \end{bmatrix} \quad (1)$$

Thus, we shall follow the three steps for the implementation of a UKF with Non-Additive Noise.

3.3.1. Step 1 : Computation of Sigma Points.

For non-additive noise, we first need to find $n' = n + n_q$ where n is the size of the state vector and n_q is the size of the noise vector. In our case, $n = 15$, $n_q = 12 \Rightarrow n' = 15 + 12 = 27$. Thus, we need to find $2n' + 1 = 2 \times 27 + 1 = 55$ Sigma Points.

We know that,

$$x \sim N(\mu, \Sigma) \quad q_t \sim N(0, Q_t) \quad (2)$$

Now, we shall form the augmented state matrix as shown below,

$$x_{aug} = \begin{bmatrix} x_{t-1} \\ q_t \end{bmatrix}$$

$$\mu_{aug,t-1} = \begin{bmatrix} \mu_{t-1} \\ 0 \end{bmatrix} \quad P_{aug} = \begin{bmatrix} \Sigma_{t-1} & 0 \\ 0 & Q_t \end{bmatrix}$$

Based on these augmented matrices we can find the 55 required sigma points as shown below -

$$\mathcal{X}^{(0)}_{aug,t-1} = \mu_{t-1}$$
$$\mathcal{X}^{(i)}_{aug,t-1} = \mu_{aug,t-1} \pm \sqrt{n' + \lambda'} \left[\sqrt{P_{aug}}\right]_i$$

To find the $\sqrt{P_{aug}}$ we can use the Cholesky Factorisation [4] method from MATLAB. We can put the above equation in a *for* loop and get 27 points using the "+" and 27 more using the "-" of the $\pm$. The "i" in $\left[\sqrt{P_{aug}}\right]_i$ represents the i^{th} column of $\sqrt{P_{aug}}$.

As for λ', it needs to be calculated using below Eq^n 3

$$\lambda' = \alpha^2(n' + k) - n' = -27 \tag{3}$$

α and k determine the sigma points spread. For a Gaussian prior distribution, good parameters are $\alpha = 0.001$ and $k = 1$.

3.3.2. Step 2 : Propagate Sigma Points through the non-linear function. Now, we must run a loop with 55 iterations (for all sigma points) and pass each sigma point through the process model function as given in Eq^n 1.

$$\mathcal{X}_t^{(i)} = f(\mathcal{X}_{aug,t-1}^{(i),x}, u_t, \mathcal{X}_{aug,t-1}^{(i),n}) \tag{4}$$

$$i = 0 \dots 2n' \quad \mathcal{X}_{aug,t-1}^{(i)} \begin{bmatrix} \mathcal{X}_{aug,t-1}^{(i),x} \\ \mathcal{X}_{aug,t-1}^{(i),n} \end{bmatrix}$$

The result is integrated by multiplication with the small time interval dt. Be careful while discretising the noise as the noise is modelled as shown below -

$$n_{ad} \sim N(0, dtQ_a) \qquad n_{bd} \sim N(0, dtQ_b)$$
$$n_{bad} \sim N(0, dtQ_{ba}) \qquad n_{bgd} \sim N(0, dtQ_g)$$

Thus, upon addition of the noise to the result of the process model function, we obtain the small change in the state vector during time interval dt. This can be added to the current sigma point in the iteration to obtain the current value of the state vector for that sigma point. This step is executed 55 times, once for each sigma point.

3.3.3. Step 3 : Compute the predicted mean and covariance matrix. This is the final step for the prediction. Firstly, we must compute the weights for the mean and covariance matrices separately -

$$W_0^{(c)'} = \frac{\lambda'}{n' + \lambda'} + (1 - \alpha^2 + \beta) = -964282 \tag{5}$$

$$W_i^{(c)'} = \frac{\lambda'}{2(n' + \lambda')} = 17857 \tag{6}$$

$$W_0^{(m)'} = \frac{\lambda'}{n' + \lambda'} = 964285 \tag{7}$$

$$W_i^{(m)'} = \frac{\lambda'}{2(n' + \lambda')} = 17857 \tag{8}$$

Where $\beta = 2$ is a good parameter for a Gaussian prior distribution. Using these weights, we shall find the weighted sum of all sigma points to finally obtain the estimated mean and covariance -

$$\bar{\mu}_t = \sum_{i=0}^{2n'} W_i^{(m)} \mathcal{X}_t^{(i)} \tag{9}$$

$$\bar{\Sigma}_t = \sum_{i=0}^{2n'} W_i^{(c)} \left(\mathcal{X}_t^{(i)} - \bar{\mu}_t\right) \left(\mathcal{X}_t^{(i)} - \bar{\mu}_t\right)^T \tag{10}$$

Thus, in such a manner we can perform the prediction step for a non-linear function using the Unscented Kalman Filter with Non-Additive Noise.

3.4. Update Step for Part 1

The update step for Part 1 is exactly the same as that in Project 1 [2]. The only difference is in the inputs. The update step shall take the position and orientation computed from the visual data obtained from the onboard camera. All logic remains the same.

3.5. Update Step for Part 2

The update step for Part 2 takes the velocity computed from the optical flow as the measurement input. This non-linear model does not inherently contain noise. Thus, we shall implement the Unscented Kalman Filter with Additive Noise as it shall be added in separately. Let us now look at the steps to implement this.

3.5.1. Step 1 : Compute Sigma Points. In this case, we need not find the augmented matrix because the noise is not an inherent part of the state. Thus, we can consider only the size of the state vector $n = 15$ and thus compute $2n + 1 = 31$ sigma points.

$$\mathcal{X}^{(0)} = \mu \tag{11}$$

$$\mathcal{X}^{(i)} = \mu \pm \sqrt{n + \lambda}[\sqrt{\Sigma}]_i \quad i = 1 \dots n \tag{12}$$

$$\lambda = \alpha^2(n + k) - n \tag{13}$$

α and k determine the sigma points spread. For a Gaussian prior distribution, good parameters are $\alpha = 0.001$ and $k = 1$.

3.5.2. Step 2 : Propagate Sigma Points through the non-linear function. Now, we shall propagate each of the 31 sigma points through the non-linear function, which will be described below. We have taken only velocity from the optical flow as our input. This is the linear velocity of the camera frame with respect to the world frame expressed in the camera frame and the angular velocity of the camera frame with respect to the world expressed in the camera frame. Using these two, we form the measurement model function $g(x_2, x_3, {}^B\omega_B^W)$ as given below,

$$g(x_t) = R_B^C \; l(x_2, x_3) - R_B^C \; S(r_{BC}^B) \; R_C^B \; {}^C\omega_C^W \tag{14}$$

Where, $l(x_2, x_3) = {}^B\dot{p}_B^W$ and represents the linear velocity of the body with respect to the world represented in the body frame. And ${}^C\omega_C^W$ was obtained from the optical flow. The above Eq^n 14 was formed using the equation for the adjoint (Fig 1)

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

Figure 1. Adjoint *(Excerpt from Lecture 11 PPT)*

To get the z_t value for each sigma point, we use the equation,

$$z_t = g(x_t) + v_t \tag{15}$$

Where $v_t \sim N(0, V)$ is the additive noise. Once we have computed all the z_t, we can move to the next step.

3.5.3. Step 3 : Compute mean and covariance matrices. This is the final step for the update. Firstly, we must compute the weights for the mean and covariance matrices separately -

$$W_0^{(c)} = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) = -937496 \tag{16}$$

$$W_i^{(c)} = \frac{\lambda}{2(n+\lambda)} = 31250 \tag{17}$$

$$W_0^{(m)} = \frac{\lambda}{n+\lambda} = -937499 \tag{18}$$

$$W_i^{(m)} = \frac{\lambda}{2(n+\lambda)} = 31250 \tag{19}$$

Where $\beta = 2$ is a good parameter for a Gaussian prior distribution. These weights, z_t, the estimated mean and the estimated covariance matrices are used to find the current mean (μ_t), current covariance (C_t), cross-covariance (S_t) and the Kalman Gain (K) through the following equations -

$$z_{\mu,t} = \sum_{i=0}^{2n} W_i^{(m)} Z_t^{(i)} \tag{20}$$

$$C_t = \sum_{i=0}^{2n} W_i^{(c)} (\mathcal{X}_t^{(i)} - \bar{\mu}_t)(Z_t^{(i)} - z_{\mu,t})^T \tag{21}$$

$$S_t = \sum_{i=0}^{2n} W_i^{(c)} (Z_t^{(i)} - z_{\mu,t})(Z_t^{(i)} - z_{\mu,t})^T + R_t \tag{22}$$

$$\mu_t = \bar{\mu}_t + K_t(z_t - z_{\mu,t}) \tag{23}$$

$$\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T \tag{24}$$

$$K_t = C_t S_t^- 1 \tag{25}$$

The only two new terms here are,

$$z_t = \text{Linear velocity from optical flow}$$
$$R_t = \text{Noise term}$$

These equations, directly implemented into MATLAB give us the update step for a UKF with Additive Noise and pairing this with the prediction step discussed earlier builds us the full-fledged Unscented Kalman Filter that can accurately provide the position, orientation and linear velocity of a quadrotor using its onboard IMU and camera.

4. Results

The implemented Unscented Kalman Filter (UKF) successfully achieved state estimation for the Micro Aerial Vehicle (MAV) in both configurations. The results showcase the UKF's ability to leverage sensor data from the onboard IMU and Camera for accurate state prediction and correction.

4.1. Part 1: UKF with Position and Orientation Measurements

As evident in Figures 2 to 7, the UKF effectively estimated the MAV's position throughout the simulation for *Dataset 1*. The close match between the estimated and actual trajectories demonstrates the filter's proficiency in utilizing position and orientation data from the camera for correction. Similar results were obtained for *Dataset 4*, which have been included in the zip file containing the script.

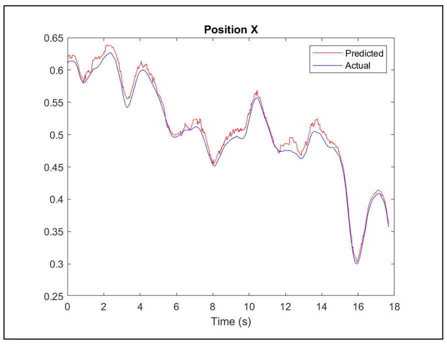

Figure 2. Position X.png

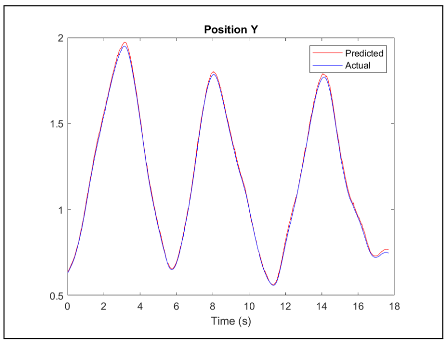

Figure 3. Position Y.png

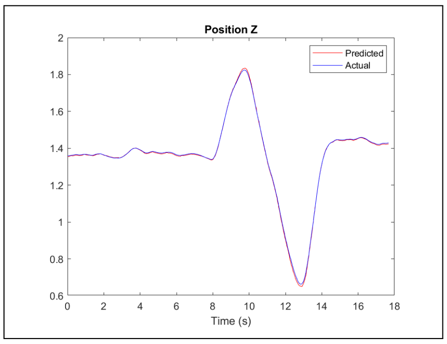

Figure 4. Position Z.png

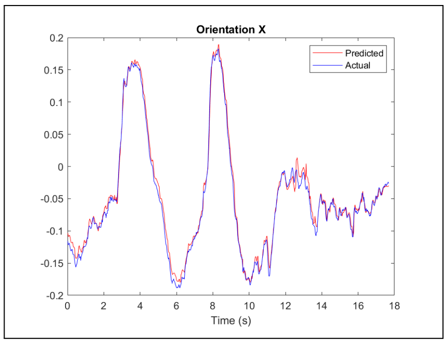

Figure 5. Orientation X.png

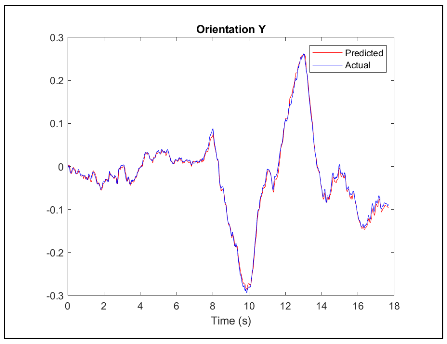

Figure 6. Orientation Y.png

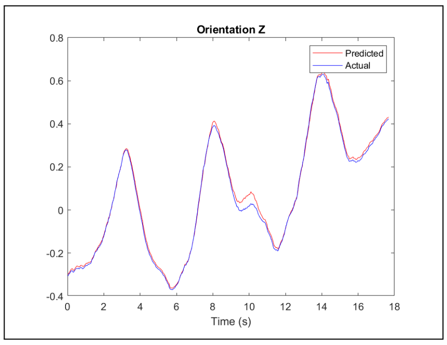

Figure 7. Orientation Z.png

Similarly, Figures 8 to 10 reveals a high degree of similarity between the estimated and actual MAV velocity for *Dataset 1*, signifying the UKF's capability in velocity estimation as well. Similar results were obtained for *Dataset 4*, which have been included in the zip file containing the script.

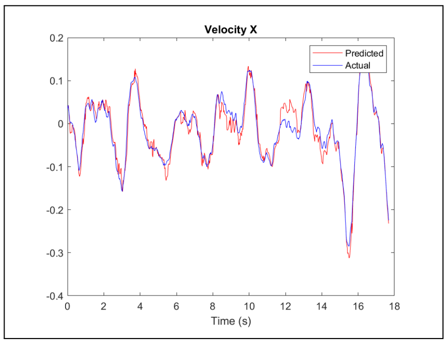

Figure 8. Velocity X.png

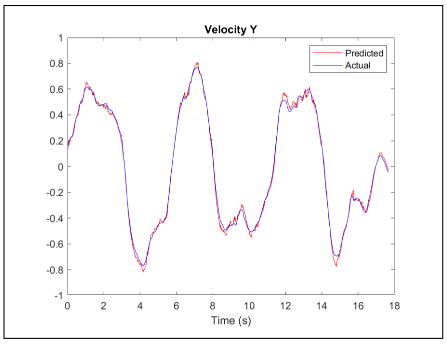

Figure 9. Velocity Y.png

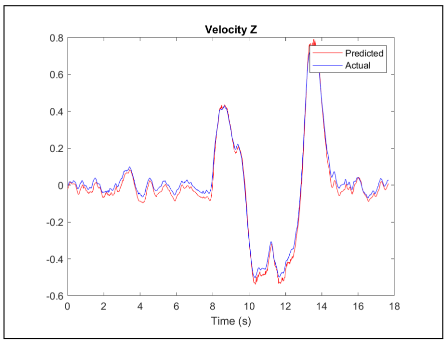

Figure 10. Velocity Z.png

The UKF also successfully estimated the sensor biases (accelerometer and gyroscope) during the filtering process, as shown in Figures 11 to 16. Accounting for these biases through the UKF's estimation process ensures that the filter utilizes the most accurate sensor information for state prediction.

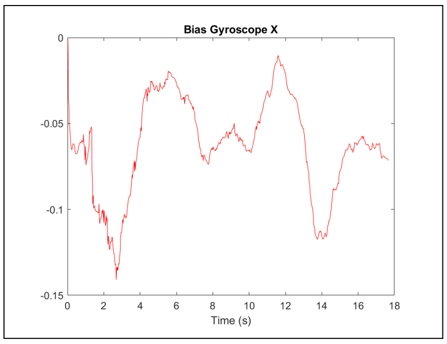

Figure 11. Bias Gyroscope X.png

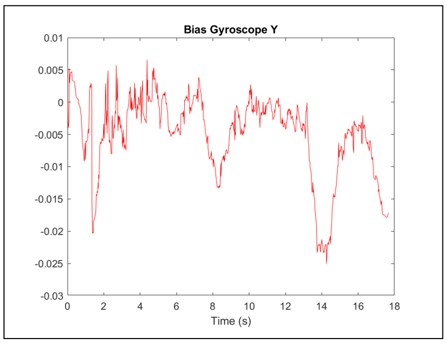

Figure 12. Bias Gyroscope Y.png

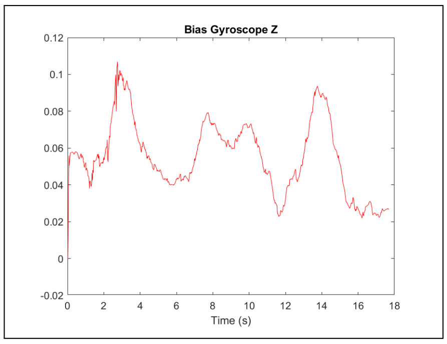

Figure 13. Bias Gyroscope Z.png

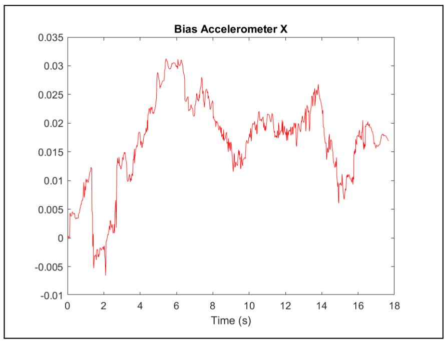

Figure 14. Bias Accelerometer X.png

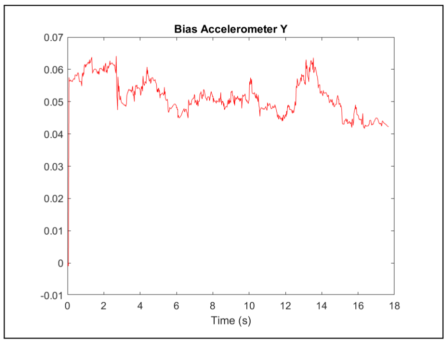

Figure 15. Bias Accelerometer Y.png

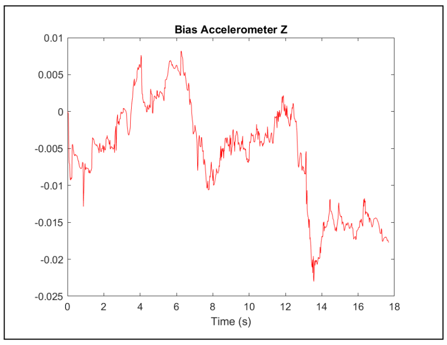

Figure 16. Bias Accelerometer Z.png

4.2. Part 2: UKF with Velocity Measurements

The trade-off between using richer measurement data and a limited set becomes evident when comparing the results of both configurations. As expected, 17 to 22 shows a greater discrepancy between the estimated and actual MAV positions compared to Part 1. Relying solely on velocity measurements for updates limits the UKF's ability to precisely track the MAV's position over time.

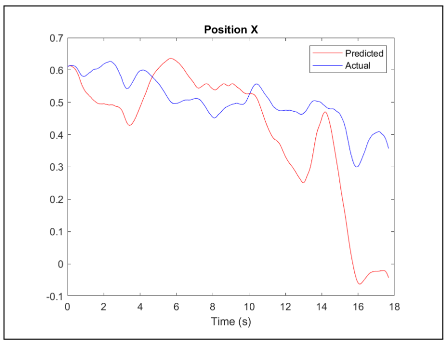

Figure 17. Position X.png

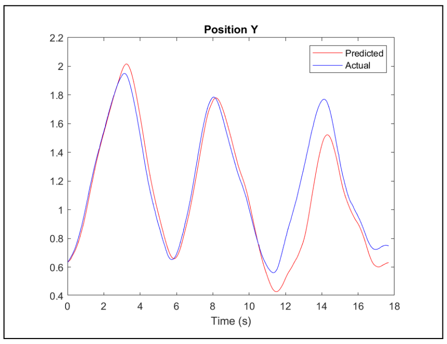

Figure 18. Position Y.png

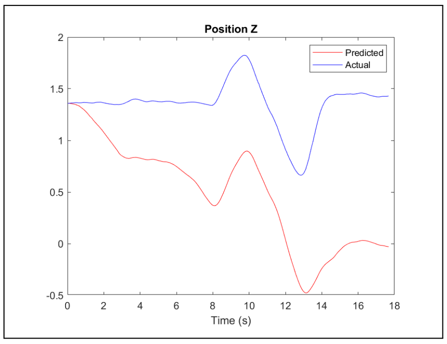

Figure 19. Position Z.png

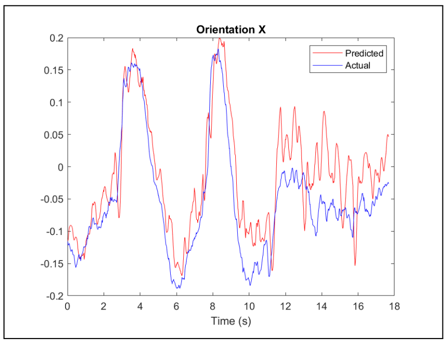

Figure 20. Orientation X.png

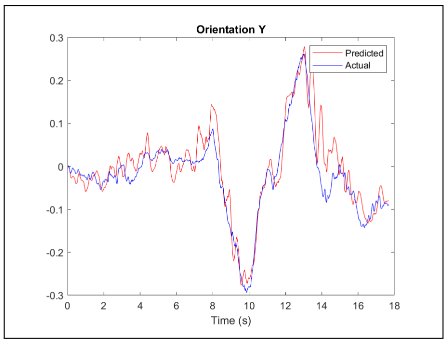

Figure 21. Orientation Y.png

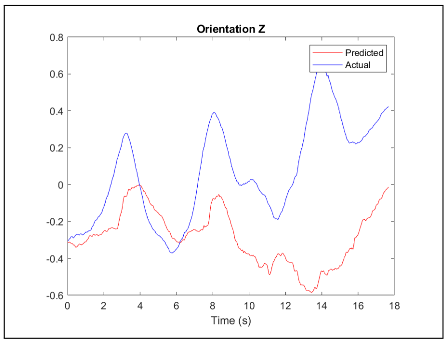

Figure 22. Orientation Z.png

However, 23 to 25 demonstrates that the UKF with velocity measurements maintains its proficiency in velocity estimation. The close match between the estimated and actual values highlights the filter's robustness in this aspect despite the limited measurement set.

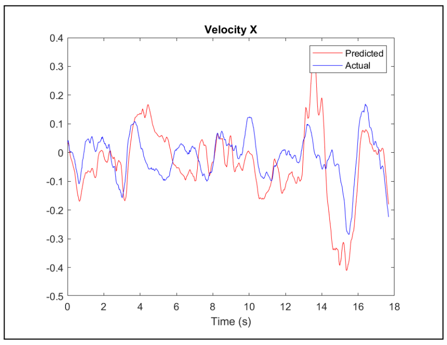

Figure 23. Velocity X.png

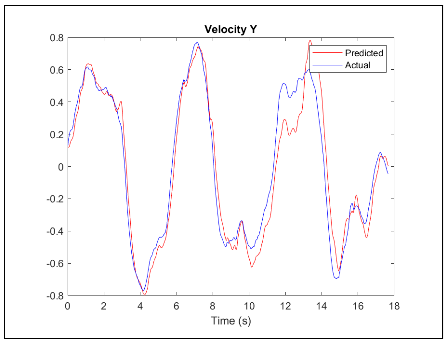

Figure 24. Velocity Y.png

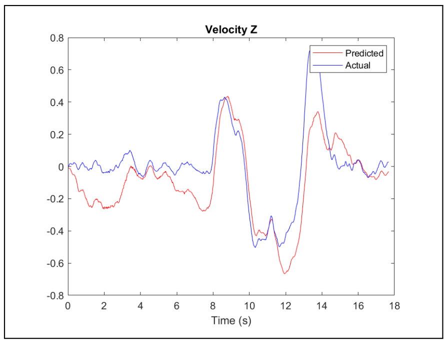

Figure 25. Velocity Z.png

The gyroscope and accelerometer biases are plotted as shown in below Figures 26 to 31.

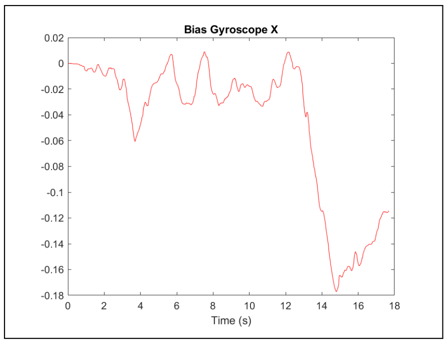

Figure 26. Bias Gyroscope X.png

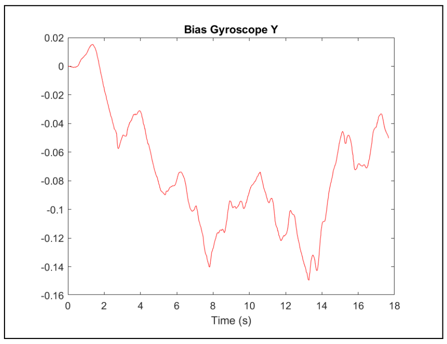

Figure 27. Bias Gyroscope Y.png

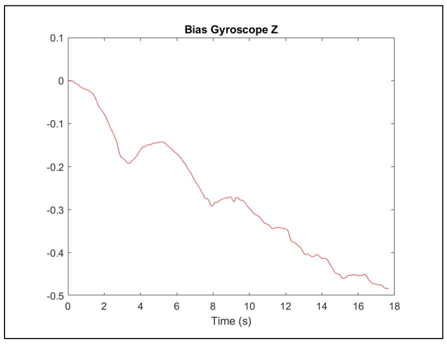

Figure 28. Bias Gyroscope Z.png

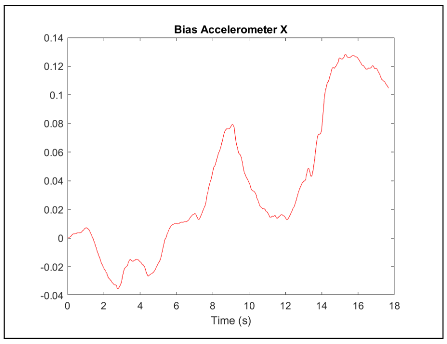

Figure 29. Bias Accelerometer X.png

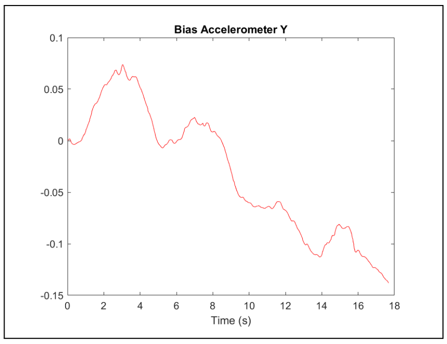

Figure 30. Bias Accelerometer Y.png

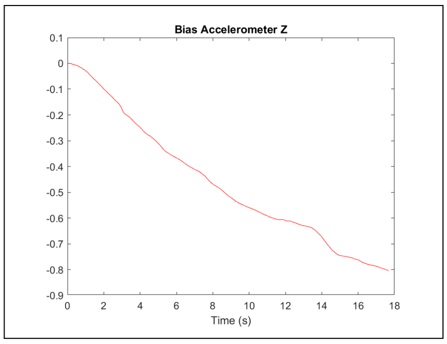

Figure 31. Bias Accelerometer Z.png

5. Conclusion

In conclusion, this project successfully implemented an Unscented Kalman Filter (UKF) for state estimation in a Micro Aerial Vehicle (MAV) simulation. The UKF effectively utilized sensor data from an onboard IMU and a camera system for state prediction and correction.

The two implemented filter versions demonstrated the UKF's versatility:

- *Part 1*: Using camera position and orientation for measurement updates provided accurate estimates of the MAV's complete state, including position, velocity, orientation, and sensor biases.
- *Part 2*: Utilizing only camera velocity for measurement updates showcased the UKF's capability to estimate the state with a reduced set of measurements.

The results highlight the UKF's potential for real-world MAV applications, where sensor data may be limited or noisy. Future work could involve incorporating additional sensor data or investigating alternative state estimation techniques for even more robust performance.

References

[1] Timothy D. Barfoot. *State Estimation for Robotics*. 1st. USA: Cambridge University Press, 2017. ISBN: 1107159393.

[2] Ghodgaonkar, Shantanu. *Implementation of an Extended Kalman Filter (EKF) for State Estimation*. 2024. URL: https://drive.google.com/file/d/1MTB0XjvszwhZJvR-bCVggfwa9PXGbHp4/view?usp=sharing.

[3] Ghodgaonkar, Shantanu. *Vision-Based 3D Attitude Estimation Using AprilTags*. 2024. URL: https://drive.google.com/file/d/1PdO5MHMplsqnmxrlCXx4kWT6H2ObG0Q3/view?usp=sharing.

[4] MathWorks. *Cholesky factorization*. 2006. URL: https://www.mathworks.com/help/matlab/ref/chol.html.

[5] Bruno Siciliano et al. *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 2010. ISBN: 1849966346.

[6] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN: 0262201623.

[7] E.A. Wan and R. Van Der Merwe. "The unscented Kalman filter for nonlinear estimation". In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*. 2000, pp. 153–158. DOI: 10.1109/ASSPCC.2000.882463.