RBE595 - Project 0 - Alohamora IMU Attitude Estimation

Pranay Katyal
Worcester Polytechnic Institute
Worcester, MA, USA
pkatyal@wpi.edu

***Abstract*—This project 0 reports aims to showcase my understanding and implementation of the project 0 - Alohamora. I work with IMU sensor and Vicon readings, and implement attitude estimation with pure gyro, pure acceleration, and by doing sensor fusion of the two using a complimentary filter. I also plot them against the true data from Vicon and showcase the comparison for all 6 of the datasets. Also try to set up the VizFlyt environment. [1][2][3]**

Keywords - IMU, Gyro, Complimentary Filter, Vicon, Sensor Fusion.

I. Problem Statement

The aim of this problem was to implement 3 methods to estimate the 3 dimensional attitude for a Drone. We are provided with the Data stream from an IMU sensor ArduIMU+ V2, which is a 6-dof inertial measurement unit, where we have readings from a 3-axis Gyroscope and 3-axis Accelerometer. We also have the ground truth data provided by the Vicon motion camera system with the Vicon markers, this provides the data in the form of Rotations.

II. Methodology

To start with the solution, we first need to get the sensor data into physical world values, as right now the raw data is in incorrect format. The data for IMU was provided from imuRaw1.mat which had its corresponding Vicon data viconRot1.mat, They both were stored in a dictionary format, and and timestamps ts attached to their readings.

A. Converting the Data

In imuRaw1, we had access to 'vals' and their corresponding 'ts'. Each column in 'vals' denotes data in the following order:

$$\begin{bmatrix} a_x & a_y & a_z & w_Z & w_x & w_y \end{bmatrix}^T$$

Since these values are not in their physical units, we need to first convert them into their physical counterparts. We were provided access to their conversion formulas and we simply had to implement them.

To convert Acceleration values to ms^{-2} we used:

$$\tilde{a}_x = \frac{a_x + b_{a,x}}{s_x}$$

Here $\tilde{a}_x$ represents a_x in Physical units, and we do the same for a_y and a_z. $b_{a,x}$ is the Bias and s_x is the Scale factor.

We also have access to IMUParams.mat file which represents a 2 x 3 Matrix where the 1^{st} Row denotes the Scale factor values, and 2^{nd} row denotes the biases (computed as the average biases of all sequences using Vicon).

To convert Angular Velocity to $rad\ s^{-1}$ we used:

$$\tilde{w} = \frac{3300}{1023} \times \frac{\pi}{180} \times 0.3 \times (w - b_g)$$

Here $\tilde{w}$ represents the value of w in Physical units and b_g is the bias, which can be calculated as the average of the first few hundred samples (with the Assumption that the IMU is at rest at the beginning).

From viconRot1.mat we get access to 'rots' and the corresponding timestamps 'ts', Here rots represent a $3 \times 3 \times N$ matrix that denotes the $Z-Y-X$ Euler angles rotation matrix as estimated by the Vicon motion capture system.

B. Rig Used

Although we were not provided with a physical rig, we did have access to what it could have looked like. We can clearly see how the rig was setup, where the IMU is, and how the Vicon markers were attached to the setup.

Fig. 1. Rig Used

Reflective Vicon markers are attached at known positions on the base, allowing the Vicon motion capture cameras to accurately track the 3D pose of the IMU during experiments. This arrangement ensured that the IMU orientation could be validated against the Vicon ground truth data while minimizing vibration and occlusion of the markers.

C. Sensor Calibration

One of the issues that we immediately encounter, is that the number of timestamps for both IMU data and Vicon data do not align, and they are also working for different amount of timestamps (although close but not exactly the same). the IMU data had 5645 timestamps, whereas Vicon data had 5561 timestamps, so we need to not only align them but also have

them same amount of timestamps. This arises from the issue that IMU and Vicon are not Hardware synchronized, so we need to synchronize them in software.

D. SLERP and mask

We start with making the timestamps same!, we do this by simply creating a mask, that uses the Vicon 'ts' and then use it to modify IMU 'ts'. This mask essentially does a very basic filter, it puts the IMU 'ts' in the range with limits of vicon 'ts' and we save this as Valid 'ts'.

The best way to align these timestamps, and also get the correct vicon data for those timestamps is to use Slerp. It is called as Spherical Linear Interpolation, and is a way to help us sample the correct Vicon data for the Valid 'ts'. We do this by first creating a Slerp object using the Scipy library's inbuilt function, that takes in the Vicon 'ts' and Vicon 'rots' in matrix form and then we use IMU 'ts', more specifically the Valid IMU 'ts' to sample the Valid vicon values from the Slerp object.

III. Implementation

A. Calculating Orientation from Gyro only

In this section I evaluate orientation from the IMU's Gyro readings, which provide us the Angular Velocity w. For this I utilized a numerical integration method known as dead reckoning. I Assumed that the initial orientation for IMU is same as Vicon since we need a starting angle for numerical integration to work.

The gyroscope provides angular velocity measurements $\tilde{w}$ along the three axes. To estimate orientation, we integrate the angular velocity measurements over time. We initialize the Euler angles with the first Vicon reading as ground truth, and then update them at each timestep using numerical integration. The incremental update is expressed as

$$\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix}_{t+1} = \begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix}_{t} + \begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}_{t} \delta t$$

where ϕ, θ, and ψ represent roll, pitch, and yaw, respectively, and $\dot{\phi}, \dot{\theta}, \dot{\psi}$ are the corresponding angular velocity components from the gyroscope. This simple dead-reckoning approach captures short-term orientation changes, but because biases and noise accumulate over time, the estimates drift significantly without external correction.

Although this method captures fast dynamics, it suffers from drift due to the accumulation of bias and noise in the gyroscope. Without correction from another sensor, the orientation estimate deviates significantly from ground truth over time.

B. Calculating Orientation from Acceleration only

The accelerometer provides measurements of both gravity and linear accelerations. When the IMU is relatively stationary or moving slowly, the accelerometer can be used to estimate the tilt (pitch and roll) by assuming that the measured acceleration corresponds primarily to gravity. The tilt angles are computed as:

$$\theta = \arctan 2\left(\frac{-a_x}{\sqrt{a_y^2 + a_z^2}}\right), \quad \phi = \arctan 2\left(\frac{a_y}{\sqrt{a_x^2 + a_z^2}}\right)$$

where θ and ϕ denote pitch and roll respectively. Yaw cannot be reliably recovered from the accelerometer alone, since gravity provides no information about rotation around the vertical axis. We do have formulation for it, in case IMU is not vertically aligned, but here I assumed it is for simplicity.

This method provides stable long-term estimates without drift, but is highly sensitive to dynamic motion. Rapid translational accelerations corrupt the gravity estimate, leading to noisy and unreliable attitude estimation during fast maneuvers.

C. Calculating Orientation from Sensor Fusion using Complimentary Filter

To leverage the strengths of both sensors, I implemented a complementary filter. The idea is to use the gyroscope for short-term orientation updates, while correcting long-term drift using accelerometer-based tilt estimates. The complementary filter blends the two sources as:

$$\hat{R}_t = \alpha\, \hat{R}_{acc,t} + (1-\alpha)\, \hat{R}_{gyro,t}$$

where $\hat{R}_{gyro,t}$ is the orientation propagated from gyroscope integration, $\hat{R}_{acc,t}$ is the orientation estimated from accelerometer tilt, and $\alpha \in [0,1]$ is a tuning parameter controlling the balance between fast response and drift correction. [4]

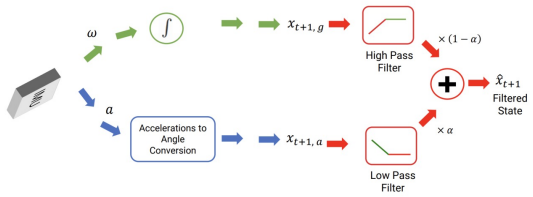

Fig. 2. Complimentary Filter

In practice, α was tuned experimentally (e.g., $\alpha = 0.9995$) to allow the gyroscope to dominate high-frequency dynamics, while the accelerometer provided a low-frequency correction. This significantly improved attitude estimation, especially for pitch and roll. Yaw remained dependent on the gyroscope due to the lack of absolute heading information from the accelerometer or Vicon system.

This fusion approach provided the most accurate results overall, effectively combining the complementary characteristics of both sensors. I did have some issues with the scaling of acceleration causing my complementary filter to not behave as good as it is supposed to, which is probably some issue with my implementation of the filter.

IV. Results

I successfully carried out the project, major learning curve for me, as before beginning this I had zero clue as to what IMU even was. But as a result of this project, I now have learnt about not only IMUs but also how they are used, what sensors are used to make an IMU, how we use those sensor readings to estimate attitude of a drone, how we can perform sensor fusion to improve results.

A. *Orientation Plots*

I also plotted $Z-Y-X$ Euler angles from each of the three implementations against the Vicon's data. which is shown below :

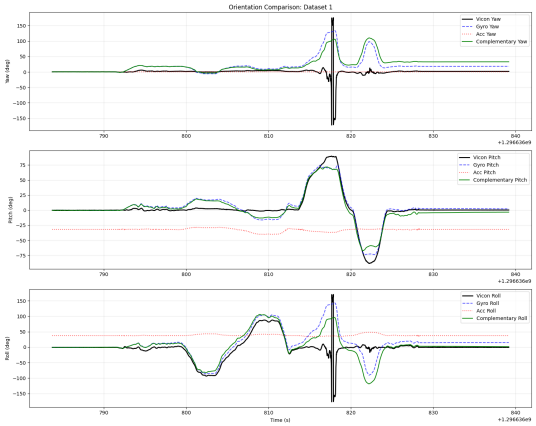

Fig. 3. Dataset 1

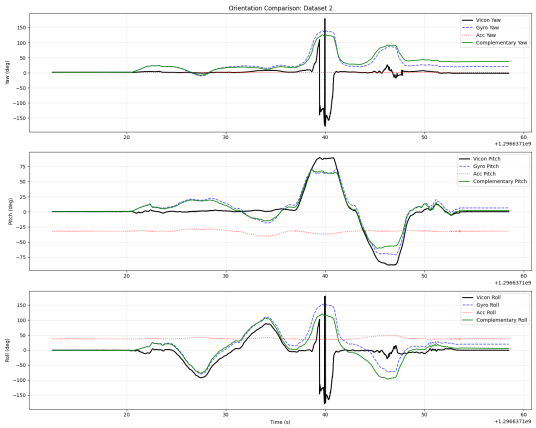

Fig. 4. Dataset 2

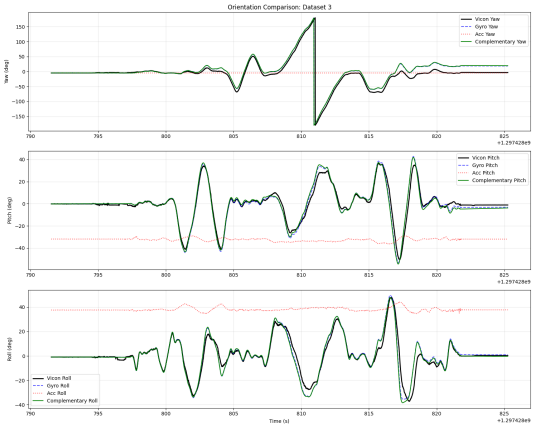

Fig. 5. Dataset 3

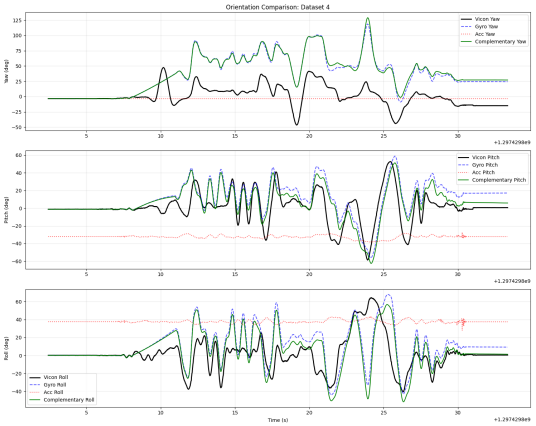

Fig. 6. Dataset 4

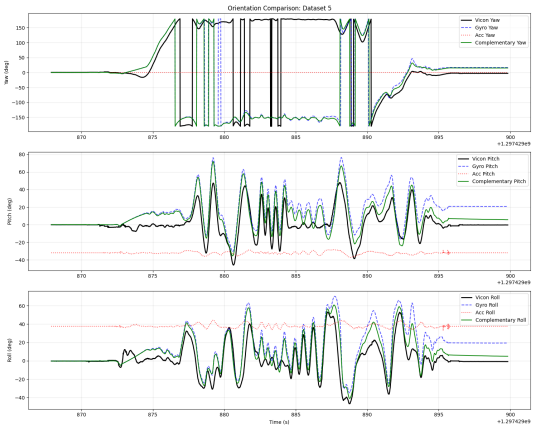

Fig. 7. Dataset 5

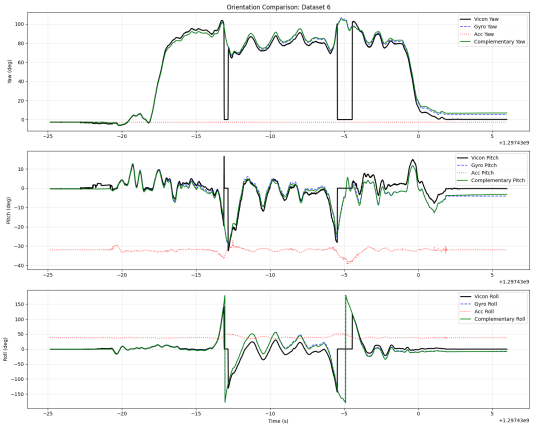

Fig. 8. Dataset 6

While some of the plot regions get tracked pretty well, there are definitely issues with the current methods and implementations, and this is why we probably have more advanced methods now that perform better!

B. Rotplot Results

I also used Rotplot.py, a script provided to plot these orientations in 3 dimensions.

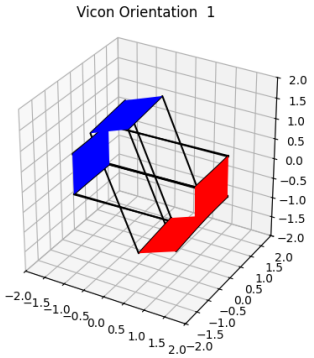

Fig. 9. Rotplot 1

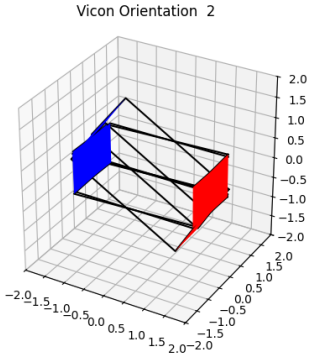

Fig. 10. Rotplot 2

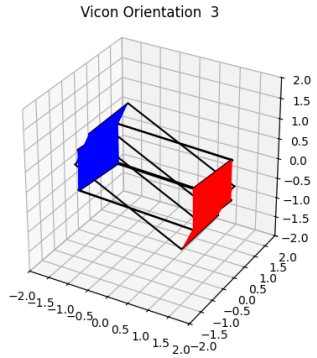

Fig. 11. Rotplot 3

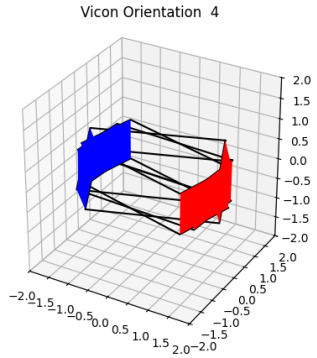

Fig. 12. Rotplot 4

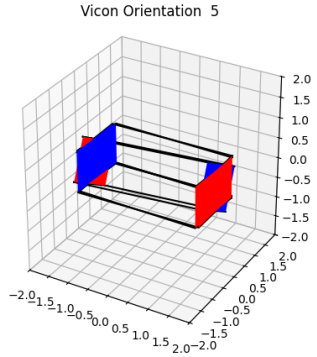

Fig. 13. Rotplot 5

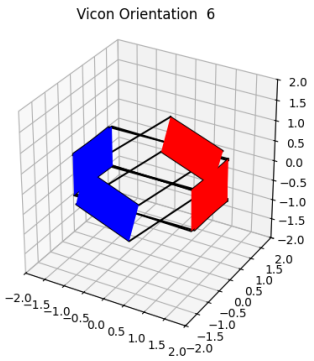

Fig. 14. Rotplot 6

C. Rotplot Videos

The Rotplot 3D visualizations videos were also made, for each of the Dataset. The video includes 4 sub-videos, showcasing the orientations from Gyro only, Acceleration only, Complementary Filter and Vicon. These provide a really good idea of how good the implementation is running with visuals that are far more intuitive.

You can access the video links from the google drive here. [5]. (check the Citation).

V. Issues

Some of the Issues I encountered while attempting this project.

1. the VizFlyt setup didn't work, had lots of compatibility issues. [6]

2. the Accelerations scale provided in IMUParams had scaling issues, which is causing complimentary filter to work bad. [7]

3. Dataset 6 had invalid rotation matrices which somehow caused my usual implementation to crash, but after skipping on those invalid ones it worked.[8]

References

[1] B. Douglas, "Drone control and the complementary filter," YouTube, 2016, accessed: 2025-08-28. [Online]. Available: https://www.youtube.com/watch?v=whSw42XddsU

[2] MathWorks, "Understanding sensor fusion and tracking, part 1: What is sensor fusion?" YouTube, 2017, accessed: 2025-08-28. [Online]. Available: https://www.youtube.com/watch?v=6qV3YjFppuc

[3] ——, "Understanding sensor fusion and tracking, part 2: Fusing a mag, accel, gyro estimate," YouTube, 2017, accessed: 2025-08-28. [Online]. Available: https://www.youtube.com/watch?v=0rlvvYgmTvI

[4] Perception and U. o. M. Robotics Group, "Enae788m: Class 2 part 2 – imu basics, attitude estimation using cf and madgwick," YouTube, 2019, accessed: 2025-08-28. [Online]. Available: https://www.youtube.com/watch?v=8hRoASoBEwY

[5] P. Katyal, "Project files for rbe595 - project 0 - alohamora imu attitude estimation," Google Drive, 2025, accessed: 2025-08-28. [Online]. Available: https://drive.google.com/drive/folders/1ZqmVCcCZM4RTusXnCexyhX69Aq9I9Xr-?usp=sharing

[6] K. Srivastava*, R. Kulkarni*, M. Velmurugan*, and N. J. Sanket, "Vizflyt: An open-source perception-centric hardware-in-the-loop framework for aerial robotics," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025, accepted for publication. [Online]. Available: https://github.com/pearwpi/VizFlyt

[7] Karooza, "Attitude estimation using imu sensors," Online Article, 2021, accessed: 2025-08-28. [Online]. Available: https://karooza.net/attitude-estimation-using-imu-sensors

[8] N. Sanket, "Attitude estimation using imu sensors (tutorial)," GitHub Pages, 2020, accessed: 2025-08-28. [Online]. Available: https://nitinjsanket.github.io/tutorials/attitudeest/imu.html