RBE595 - Project 1a - Madgwick Filter

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

Anirudh Ramanathan
Worcester Polytechnic Institute
Worcester, MA, USA
aramanathan@wpi.edu

***Abstract*—This report for Project 1a seeks to showcase our understanding and implementation of the project 1a - Madgwick Filter for Attitude Estimation. In the last assignment, we worked with IMU sensor and Vicon readings, and implemented attitude estimation with pure gyro, pure acceleration, and by doing sensor fusion of the two using a complementary filter. In this task, we further expand on the stability and precision of our system by implementing a Madgwick Filter We also plot the new data against the previous methods and true data from Vicon to showcase the comparison for all 6 of the datasets.**

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

I. Problem Statement

The main objective of the last project was to estimate the three-dimensional orientation (attitude) using acceleration and gyroscope measurements from a six-degree-of-freedom Inertial Measurement Unit (6-DoF IMU). The three previously implemented methods to estimate orientation were performed by the Gyroscope, the Accelerometer, and a fusion of these measurements by the Complementary Filter. The objective of this project was to extend the concept of sensor fusion by filters using the high-accuracy and memory-efficient Madgwick Filter.[1], [2], [3], [4], [5], [6]

The Madgwick filter differs from the Complementary Filter in that it uses a gradient descent method on accelerometer-based orientations to minimize the error between measured gyroscope data and the overall estimated orientation. Moreover, it utilizes the Quaternion for calculations instead of Euler angles.

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:

$$\tilde{a}_x = (a_x * s_x) + b_{a,x}$$

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.

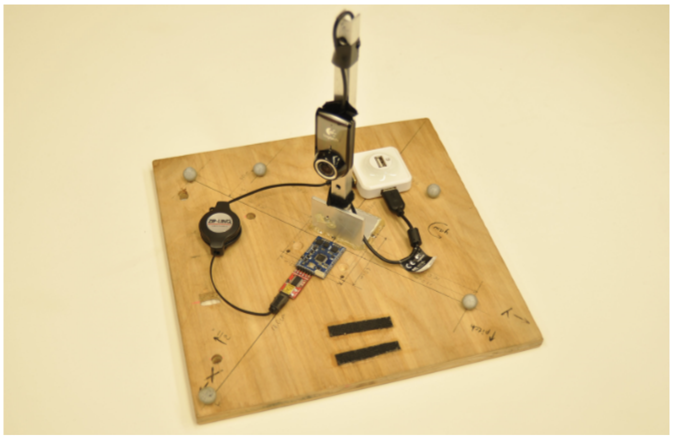
Fig. 1. Rig Used

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 we evaluate orientation from the IMU's Gyro readings, which provide us the Angular Velocity w. For this we utilized a numerical integration method known as dead reckoning. We 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, considering the fact that we assumed IMU is vertical. We do have formulation for it, in case IMU is not vertically aligned, but here we 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, we 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. [1]

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.

IV. Calculating Orientation from Sensor Fusion using Madgwick Filter

While the Complementary Filter fuses gyroscope and accelerometer information in Euler space, it has two main limitations:

- the accelerometer signal is corrupted by translational accelerations, which reduces reliability of long-term correction
- the use of Euler angles introduces singularities (gimbal lock)

To overcome these issues, we implemented the **Madgwick Filter**, which works in quaternion space and performs a gradient-descent correction step based on accelerometer data. This avoids singularities and improves long-term stability.

A. Initialization

The filter starts with an initial orientation. For training datasets, this is taken from the first Vicon reading, converted to a quaternion using For test datasets without Vicon, the initial orientation is assumed to be zero (identity quaternion). We use the formula below to get the initial quaternion. `R.from_euler('zxy', initial_orientation, degrees=True).as_quat(scalar_first=True)`

B. Gyroscope Update

At each time step, the body angular velocity $\omega = [\omega_x, \omega_y, \omega_z]^T$ is first expressed as a quaternion

$$\omega_q = \begin{bmatrix} 0 & \omega_x & \omega_y & \omega_z \end{bmatrix}^T$$

and the quaternion derivative from the gyroscope is computed as

$$\dot{q}_\omega = \tfrac{1}{2}\, q \otimes \omega_q$$

where $\otimes$ denotes quaternion multiplication. This corresponds to dead-reckoning integration of gyroscope data.

$$q_1 \otimes q_2 = \begin{bmatrix} w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2 \\ w_1x_2 + x_1w_2 + y_1z_2 - z_1y_2 \\ w_1y_2 - x_1z_2 + y_1w_2 + z_1x_2 \\ w_1z_2 + x_1y_2 - y_1x_2 + z_1w_2 \end{bmatrix}$$

C. Accelerometer Correction

To prevent drift, a correction step is applied using normalized accelerometer measurements $\hat{a} = [a_x, a_y, a_z]^T / \|a\|$.

The objective is to minimize the error between the expected gravity vector from the quaternion and the measured gravity vector from the accelerometer. When the IMU is stationary or moving slowly, the accelerometer primarily measures gravity, which should appear as $[0, 0, 1]^T$ in the global reference frame. Using the current quaternion estimate $q = [w, x, y, z]$, we can predict what this gravity vector should look like in the sensor frame by rotating it through the inverse quaternion transformation.

The residual function f captures the mismatch between this predicted gravity direction and the actual accelerometer measurement:

$$f = \begin{bmatrix} 2(xz - wy) - a_x \\ 2(wx + yz) - a_y \\ 2(0.5 - x^2 - y^2) - a_z \end{bmatrix}$$

Each component represents the error along one sensor axis. When $f = 0$, the quaternion estimate perfectly aligns with the measured gravity direction.

To minimize this error using gradient descent, we need the Jacobian matrix J, which describes how small changes in each quaternion component affect the residual error:

$$J = \begin{bmatrix} -2y & 2z & -2w & 2x \\ 2x & 2w & 2z & 2y \\ 0 & -4x & -4y & 0 \end{bmatrix}$$

Each column of J represents the partial derivatives of f with respect to one quaternion component (w, x, y, z respectively). This matrix encodes the sensitivity of the gravity prediction error to changes in orientation.

The gradient of the error function is computed as:

$$\nabla f = J^T f$$

This gradient points in the direction of steepest increase in error. To reduce the error, we move in the opposite direction, scaled by a tuning parameter β:

$$\dot{q}_\nabla = -\beta \frac{\nabla f}{\|\nabla f\|}$$

The normalization ensures that the correction magnitude is independent of the error scale, providing stable convergence behavior. The parameter β controls how aggressively the filter corrects for accelerometer-based errors versus trusting the gyroscope integration.

D. Quaternion Update

The Madgwick filter combines information from both sensors by fusing the gyroscope-based quaternion derivative with the accelerometer-based correction term. The total quaternion derivative represents the complete rate of change in orientation:

$$\dot{q} = \dot{q}_\omega + \dot{q}_\nabla$$

where $\dot{q}_\omega$ captures the rotational dynamics from the gyroscope, and $\dot{q}_\nabla$ provides the gravity-based correction from the accelerometer.

This combined rate of change is integrated forward in time using Euler's method:

$$q_{t+1} = q_t + \dot{q}\,\Delta t$$

where Δt is the time step between consecutive IMU measurements. This integration step propagates the orientation estimate from the previous time step to the current one, incorporating both the predicted motion and the measured correction.

However, numerical integration can introduce small errors that cause the quaternion to deviate from unit length. Since valid rotation quaternions must have unit magnitude, normalization is essential:

$$q_{t+1} = \frac{q_{t+1}}{\|q_{t+1}\|}$$

This normalization step ensures that the quaternion remains on the unit sphere in 4D space, preserving its validity as a rotation representation. Without this step, accumulated numerical errors would eventually cause the quaternion to represent invalid transformations, leading to incorrect orientation estimates.

The normalized quaternion q_{t+1} now serves as the updated orientation estimate for the current time step, ready to be converted to Euler angles for visualization or used as the starting point for the next iteration of the filter.

V. Results

We successfully carried out the project, major learning curve for us, was getting to learn about the Madgwick filter and how to implement it. But as a result of this project, we now have learnt about Madgwick Filters, and how to utilize them to get way better estimates of attitude as compared to other methods. We also learnt about how to apply quaternion and use them.

The results for complimentary filter now perform better at values like $\alpha = 0.99$ because we were able to fix our acceleration value issues from project 0. This also further improved the Madgwick filter.

This implementation effectively balances the short-term accuracy of the gyroscope with the long-term stability of the accelerometer. The tuning parameter β (set to 0.01 in our experiments) controls the correction strength. With proper tuning, the Madgwick filter tracked the Vicon ground truth more accurately than the gyroscope-only and Complementary Filter approaches, especially for roll and pitch, while avoiding the drift observed in yaw. Even though the recommended value by the Author of the Madgwick filter is 0.041, upon testing around multiple values like $0.0, 0.8, 0.2, 0.1$, we found that 0.01 was a clear winner for us.[7]

A. Orientation Plots

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

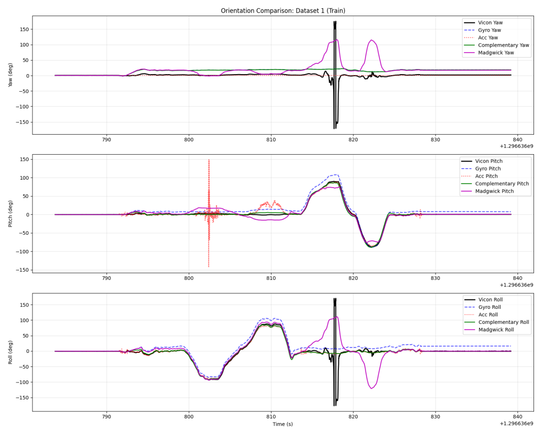

Fig. 2. Dataset 1

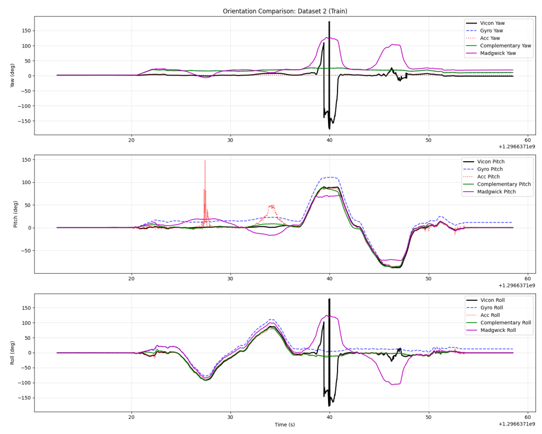

Fig. 3. Dataset 2

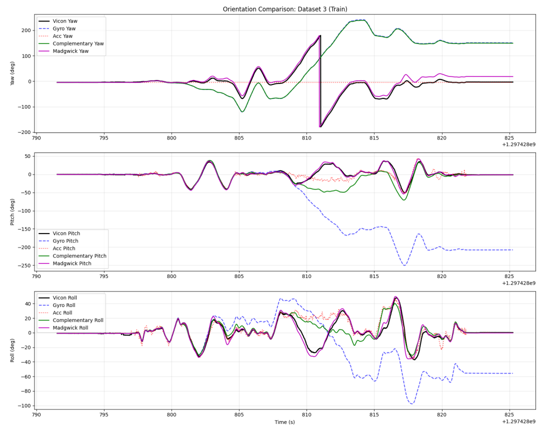

Fig. 4. Dataset 3

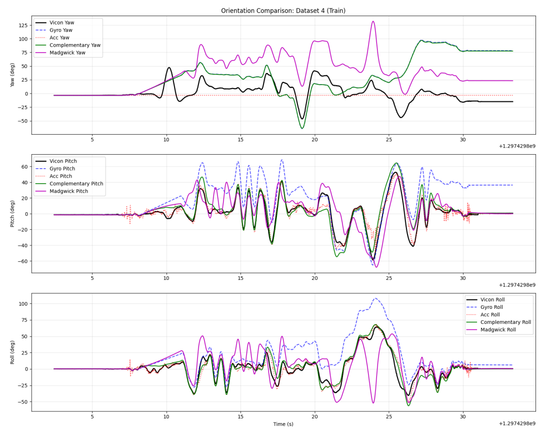

Fig. 5. Dataset 4

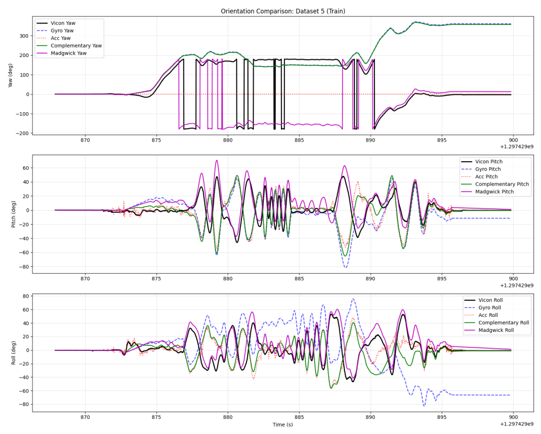

Fig. 6. Dataset 5

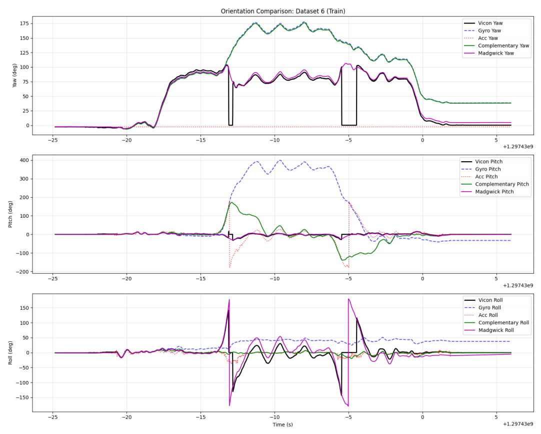

Fig. 7. Dataset 6

B. Test set - Orientation plots

The following Datasets were the Test sets that did not have the Vicon data accessible this was part of the imuRaw7,8,9,10 files although it is hard to verify their correctness, The videos from rotplot do show us that the Madgwick Filter outperforms others.

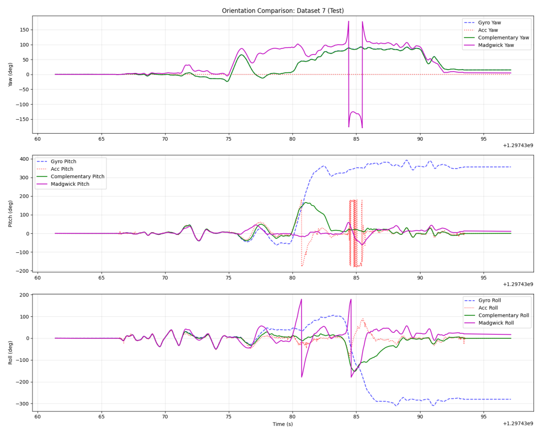

Fig. 8. Dataset 7

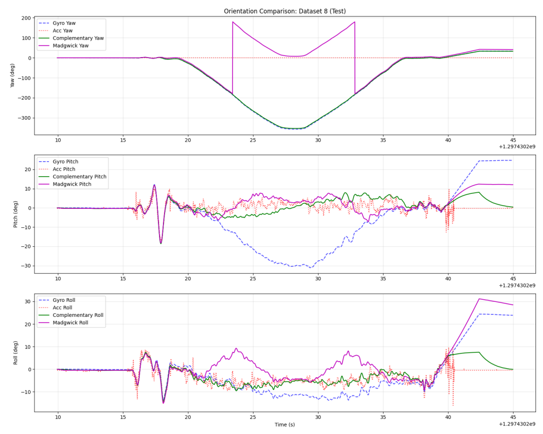

Fig. 9. Dataset 8

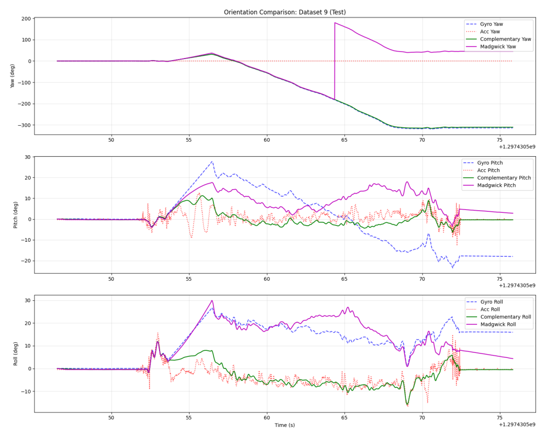

Fig. 10. Dataset 9

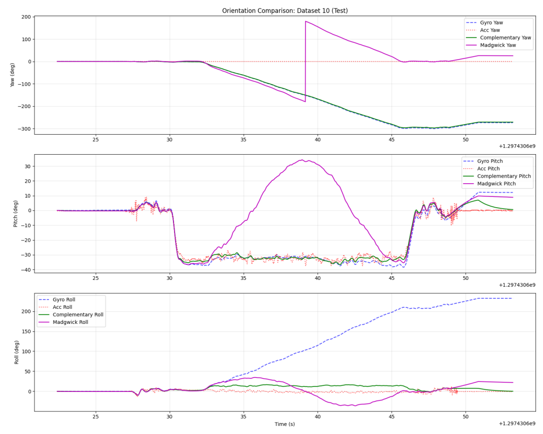

Fig. 11. Dataset 10

For the Training sets 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 like Madgwick Filter now, that outperformed all of them.

The test sets assumed that the initial orientations are Zeros, therefore that may play some role on some of the plots not following the trend, but considering the fact how accurate is Madgwick, I would still consider it working as optimal as it can. Performance can vary with tuning, right now the $\beta = 0.01$, the paper recommends 0.041, but for us this value works better.

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. [8]. (check the Citation).

References

[1] 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

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

[3] 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

[4] 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

[5] ——, "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

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

[7] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," Conference Paper, IEEE International Conference on Rehabilitation Robotics (ICORR), 2011, [Online]. Available: https://drive.google.com/file/d/1LLJ5fZFMdWO4IAT66aDA-JIpTM7FhE47/view?usp=sharing.

[8] P. Katyal and A. Ramanathan, "Project files for rbe595 - project 1a - madgwick filter," Google Drive, 2025, accessed: 2025-08-28. [Online]. Available: https://drive.google.com/drive/folders/1GL-p1g7so_SICBe3jr1nLcd5VnRqMIyi?usp=sharing