Attitude Estimation and Control for a Quadcopter UAV with concepts from manifold theory

M.Tech. Project Stage II

Submitted in partial fulfillment of requirements for the degree of

Master of Technology

By

Arjun Sadananda
Roll No.: 213236002

under the guidance of

Prof. Ravi Banavar

Systems and Control Engineering
Indian Institute of Technology, Bombay

June, 2024

Dissertation Approval

This dissertation entitled

Attitude Estimation and Control for a Quadcopter UAV with concepts from manifold theory

by

Arjun Sadananda
Roll No.: 213236002

is approved for the degree of
Master of Technology in Systems and Control Engineering

Digital Signature
Ravi N Banavar (i93001)
04-Jul-24 10:58:16 AM

....

Prof. Ravi Banavar
(Supervisor)

Digital Signature
SUKUMAR SRIKANT (i11119)
03-Jul-24 10:58:43 PM

....

Prof. Sukumar Srikant
(Examiner I)

Digital Signature
Kavi Arya Arya (i01015)
03-Jul-24 10:15:52 PM

....

Prof. Kavi Arya
(Examiner II)

Date: July 3, 2024
Place: IIT Bombay

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Digital Signature
Arjun Sadananda (213236002)
04-Jul-24 12:34:13 PM

....

Arjun Sadananda
Roll No.: 213236002

Date : July 3, 2024

Acknowledgement

I express my heartfelt gratitude to **Prof. Ravi Banavar** for his unwavering guidance and support throughout this project and beyond. His mentorship played a pivotal role in shaping my M.Tech journey, and I am deeply grateful for his invaluable insights and encouragement.

I extend my sincere thanks to **Prof. Sukumar Srikanth** for his constructive feedback and valuable inputs.

I am indebted to the **e-Yantra project, Prof. Kavi Arya and team** for their continuous support and resources that were instrumental in the successful completion of this research endeavor.

I would like to acknowledge **Viyom Vivek** for insightful discussions that helped clarify my concepts and deepen my understanding.

My heartfelt thanks go to **my family** for their unwavering support, understanding, and encouragement throughout this journey.

Lastly, I am grateful for the stimulating environment provided by my guides and peers at **IIT Bombay**, which has been instrumental in my growth and learning.

Abstract

This project addresses the development and implementation of attitude estimation and control strategies for a quadcopter UAV, leveraging concepts from manifold theory. An extension of the Manifold (Multiplicative) Extended Kalman Filter (MEKF) is provided, incorporating a TRIAD-aided approach to improve robustness in the presence of magnetic disturbances. The TRIAD-aided MEKF is rigorously tested, demonstrating its robustness against the (inertial and magnetic) noises and disturbances caused by the quadcopter.

The project also explores the design of a geometric nonlinear controller for attitude tracking, specifically focused on stabilisation under harsh initialization conditions.

To facilitate the implementation and testing of these algorithms, a custom flight controller firmware is developed for the Teensy 4.0/4.1 Development Boards. This firmware is designed to be modular and accessible, allowing for easy integration and testing of new algorithms. It serves as a platform for future research and development in the field of UAV control systems. This work contributes to the field by enhancing the practical applicability of manifold-based estimation and control techniques for quadcopter UAVs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Control Quoting from [1] (2010): "There are several university-level projects [2], [3], [4], [5], and commercial products related to the development and application of quadrotor UAVs. Despite the substantial interest in quadrotor UAVs, little attention has been paid to constructing nonlinear control systems for them, particularly in designing nonlinear tracking controllers. Linear control systems such as proportional-derivative controllers or linear quadratic regulators are widely used to enhance the stability properties of an equilibrium [2], [4], [5]. A nonlinear controller is developed for the linearized dynamics of a quadrotor UAV with saturated positions in [6]. Backstepping and sliding mode techniques are applied in [7]. Since these controllers are based on Euler angles, they exhibit singularities when representing complex rotational maneuvers of a quadrotor UAV, thereby fundamentally restricting their ability to track nontrivial trajectories." This is also reflected in the fact that the controllers implemented in the most popular open-source flight controllers (Betaflight, iNav, Arducopter) are still at the core, PIDs with Euler angles.

This project was initiated to compare the open-source flight controllers with the Geometric Nonlinear controller proposed by [1]. The control objective chosen was to stabilize the quadcopter under harsh initialization.

However, before we move to harsh initialization, it is natural to start with the most fundamental mode of control in a flight controller: "Angle Mode" with a pilot. Therefore, we first implement Attitude Tracking Mode in this M.Tech Project. The stabilisation under harsh initialization has been implemented on hardware but requires more tuning and testing.

A major outcome of this project is the development of flight controller firmware on which new controllers and estimators can be easily implemented. The flight controller firmware developed can be found at https://github.com/arjun-sadananda/TeensyPilot.git. A very important part of any Flight Controller is the Attitude Estimator, and therefore we implement the popular MEKF for attitude estimation. Additionally, we take it a step further and include the magnetometer to aim for "headless mode" piloting which presents its own set of challenges.

Estimator The first question while designing an estimator is, what attitude representation shall we work with? Considering that the special orthogonal group SO(3) has dimension three, we ideally seek a continuous and non-singular representation expressed by 3 parameters. However, in 1964 it has been shown that “...it is topologically impossible to have a global 3-dimensional parametrization without singular points for the rotation group”.

Knowing this, we would not be wrong to say that unit quaternions are the most convenient representation we have and that we will have for orientations [8].

Since a quaternion (without the unit norm constraint) is of dimension 4, one tends to think at first on a 4 × 4 covariance matrix, and in the direct application of the Kalman Filter [9]. Given that all predictions are contained in the surface defined by the unit constraint, the covariance matrix shrinks in the orthogonal direction to this surface, which leads to a singular covariance matrix after several updates. A second perspective was firstly approached in reference [10] and was later named as “Multiplicative Extended Kalman Filter”. In this approach an “error-quaternion” [11] is defined which is then transformed to a 3-vector. We use this vector to build the covariance matrix, and we talk about a “3×3 representation of the quaternion covariance matrix”. However, there are still details in this adaptation that are currently being developed. Namely, the “covariance correction step” [12].

Reference [8] presents a new viewpoint by noticing that unit quaternions live in a manifold (the unit sphere in $\mathbb{R}^4$). This work uses basic concepts from manifold theory to define the mean and covariance matrix of a distribution of unit quaternions. With these definitions an EKF-based estimator is developed, arriving at the concepts of “multiplicative update” and “covariance correction step” in a natural and satisfying way.

In this M.Tech Project, we extend the (Manifold Extended Kalman Filter) MEKF estimator [8] to include a second reference vector, the magnetometer (along with the accelerometer). However, the performance of this extended estimator was found to be unreliable in practice primarily due to the inconsistency in the magnetometer readings. [13] describes how magnetic disturbances influence attitude estimation and explains common methods used to decouple attitude (pitch and roll) estimation from magnetic disturbances.

To tackle this Martin and Salaun in the development of an invariant nonlinear observer [14], proposed a simple solution by creating another inertial vector as the cross-product of the gravitational acceleration and the Earth’s magnetic field. This provides a decoupling of the attitude from the magnetic measurements.

A recent work [15] addresses this by constructing a new second reference vector by removing the component of magnetometer reading along the accelerometer reading and using this reference vector in the measurement model in the quaternion EKF. [15] also compares its approach with three other estimators [16] [17] [18] and through experimental analysis shows EKF to be the preferred algorithm when various practical problems are encountered.

An interesting observation to be made here is that the two approaches mentioned above are related to the second and third columns of the rotation matrix generated by the TRIAD [19]. It is known that TRIAD (constructed from two vectors) is suboptimal because it ignores one piece of information from one of the unit vectors [20]. This works in our favor since we wish to decouple the magnetometer readings from influencing the attitude estimate.

Therefore we introduce the idea of "pre-processing the magnetometer vector" by using the second or third column of the rotation matrix generated by TRIAD as the second vector in the extended Manifold EKF. As expected this improved the performance drastically. An alternate approach to achieving this could be tuning the Measurement Noise Covariance Matrices but this reduces the responsiveness (of the estimator) to changes in the yaw axis. Reference [21] also takes a very similar but yet different approach to what has been implemented this M.Tech Project.

We summarize this work as follows:

- In Chapter 2 we review the Manifold EKF as described in [8] and the TRIAD. Then we present the extension and modifications made to the estimation algorithms. Finally, we present the results of the experiments conducted to study the effects of these modifications.
- In Chapter 3 we review the Attitude Tracking Controller as described in [1]. We also define our control strategy for stabilizing a quadcopter under harsh initialization. Additionally, we describe the observations, challenges, and solutions found during the hardware implementation of the controller,
- A major outcome of this project is a bare-bones flight controller firmware developed from scratch. This firmware is an excellent platform to develop and test new estimators and controllers without delving into relatively complex firmware like Arducopter, iNav, or Betaflight. Chapter 3 also provide a brief overview of this firmware and the hardware used in the project.
- Finally chapter 4 shares some conclusions and future scope of the project.

Chapter 2

Robust 3D Orientation Estimation with TRIAD aided MEKF

2.1 MEKF Description

In this section we review the Manifold EKF as described in [8].

2.1.1 Preliminaries

The orientation representation used in the MEKF is a unit quaternion, which lives in a manifold (S^3).

To define the probability distribution and its evolution using the Kalman filter, we will need to define charts for the manifold. [11] discusses the three-component "attitude error representations", namely Orthographic, Rodrigues Parameters, Modified Rodrigues Parameters, and Rotation Vector. All four are charts φ that map a point $\boldsymbol{q}$ in the manifold with a point $\boldsymbol{e}$ in $\mathbb{R}^3$

$$\mathbb{R}^3 \ni \boldsymbol{e} = \varphi(\boldsymbol{q})$$

All four charts provide the same second-order approximation for a point $\boldsymbol{e} \in \mathbb{R}^3$ near the origin, to a quaternion $\boldsymbol{q} \in \mathrm{S}^3$:

$$\varphi^{-1}(\boldsymbol{e}) \approx (1 - ||\boldsymbol{e}||^2/8, \boldsymbol{e}/2)^T$$

We notice that a chart φ will inevitably produce a deformation of the space. However, for the quaternions in the neighborhood of the identity quaternion, the charts behave like the identity transformation. This is desirable for the Kalman Filter as this means that the space around the identity quaternion closely resembles the Euclidean space. But this just happens in the neighborhood of the identity quaternion. However, we can extend this property for any quaternion $\bar{\boldsymbol{q}}$. Any quaternion $\boldsymbol{q}$ can be expressed as a "deviation" from $\bar{\boldsymbol{q}}$ through

$$\boldsymbol{q} = \bar{\boldsymbol{q}} * \boldsymbol{\delta}^{\bar{q}}$$

where $*$ represents quaternion product. Then, we define a chart $\varphi_{\bar{q}}$ for $\bar{\boldsymbol{q}}$ as

$$\boldsymbol{e}^{\bar{q}} = \varphi_{\bar{q}}(\boldsymbol{q}) = \varphi(\delta^{\bar{q}})$$

We refer to the Euclidean space associated with the chart $\varphi_{\bar{\boldsymbol{q}}}$ as $\bar{\boldsymbol{q}}$-centered chart. Thus $\varphi_{\bar{\boldsymbol{q}}}^{-1}$ takes a point $\boldsymbol{e}^{\bar{\boldsymbol{q}}}$ in $\bar{\boldsymbol{q}}$-centered chart and maps it to $\boldsymbol{q}$ in the manifold through

$$\boldsymbol{q} = \varphi_{\bar{\boldsymbol{q}}}^{-1}(\boldsymbol{e}^{\bar{\boldsymbol{q}}}) = \bar{\boldsymbol{q}} * \varphi^{-1}(\boldsymbol{e}^{\bar{\boldsymbol{q}}})$$

Given a unit quaternion $\bar{\boldsymbol{q}}$ and a chart φ, we will define the expected value of a distribution of unit quaternion in the $\bar{\boldsymbol{q}}$-centered chart as

$$\bar{\boldsymbol{e}}^{\bar{\boldsymbol{q}}} = E[\boldsymbol{e}^{\bar{\boldsymbol{q}}}]$$

and its covariance matrix as

$$\mathbf{P}^{\bar{\boldsymbol{q}}} = E[(\boldsymbol{e}^{\bar{\boldsymbol{q}}} - \bar{\boldsymbol{e}}^{\bar{\boldsymbol{q}}})(\boldsymbol{e}^{\bar{\boldsymbol{q}}} - \bar{\boldsymbol{e}}^{\bar{\boldsymbol{q}}})^T]$$

2.1.2 Motion Equations and Measurement Model

We can predict the value of the random variables that describe the state of our system through the following motion equation:

$$\frac{d\boldsymbol{\omega}'(t)}{dt} = \mathbf{q}^{\omega}(t)$$

$$\frac{d\boldsymbol{q}(t)}{dt} = \frac{1}{2}\boldsymbol{q}(t) * \begin{bmatrix} 0 \\ \boldsymbol{\omega}'(t) \end{bmatrix}$$

$\mathbf{q}^{\omega}(t)$ is process noise with covariance matrix Q_{ω}.

We assume we have sensors giving measurements of angular velocity $\boldsymbol{\omega}_t$ and of a reference vector $\boldsymbol{v}_t$ whose value $\boldsymbol{v}_r$ (r for reference) expressed in the external reference frame S is known. The two sensors used for this are a gyroscope and an accelerometer. We assume the gyroscope is calibrated to ensure that the biases are zero. We know that the accelerometer measurement $\boldsymbol{a}_t$ in the external reference frame must measure $\boldsymbol{a}_r := -\boldsymbol{g}$

The measurement model relates measurements $\boldsymbol{a}_t$ and $\boldsymbol{\omega}_t$ with the states $\boldsymbol{q}_t$ and $\boldsymbol{\omega}'_t$ [1]

$$\boldsymbol{a}_t = \mathbf{R}^T(\boldsymbol{q}_t)\boldsymbol{a}_r + \mathbf{r}_t^a$$

$$\boldsymbol{\omega}_t = \boldsymbol{\omega}'_t + \mathbf{r}_t^{\omega}$$

Where $\mathbf{R}^T(\boldsymbol{q}_t)$ is the transpose of the rotation matrix corresponding to the quaternion $\boldsymbol{q}_t$.

$\mathbf{r}_t^{\omega}$ and $\mathbf{r}_t^a$ are measurement noise with zero mean and covariance R_a R_{ω}.

[1] external disturbances have been removed from the accelerometer measurement model

2.1.3 Attitude Estimator Description

The state of the system at time t is defined by orientation $\boldsymbol{q}_t \in \mathbb{S}^3$ (unit quaternion) and angular velocity in body frame $\boldsymbol{\omega}'_t$ in rad/sec.

Our knowledge about the state at a time t_{n-1}, having included measurement up to a time t_{n-1} is described by a distribution expressed in the $\bar{\boldsymbol{q}}_{n-1|n-1}$-centered chart. We assume this distribution has a mean

$$\bar{\boldsymbol{x}}^{\bar{q}_{n-1|n-1}}_{n-1|n-1} = \begin{bmatrix} \bar{\boldsymbol{e}}^{\bar{q}_{n-1|n-1}}_{n-1|n-1} = 0 \\ \bar{\boldsymbol{\omega}}'_{n-1|n-1} \end{bmatrix}$$

and 6x6 covariance matrix $\mathbf{P}^{\bar{q}_{n-1|n-1}}_{n-1|n-1}$

Prediction

We can predict the values of the random variables that describe the state through the motion equation. Therefore the prediction step of the Kalman filter is as follows

$$\boxed{\bar{\boldsymbol{\omega}}'_{n|n-1} = \bar{\boldsymbol{\omega}}'_{n-1|n-1}}$$

$$\boldsymbol{\delta}^{\boldsymbol{\omega}}_{\boldsymbol{n}} = \begin{bmatrix} cos(\frac{||\bar{\boldsymbol{\omega}}'_{n|n-1}||\Delta t_n}{2}) \\ \frac{\bar{\boldsymbol{\omega}}'_{n|n-1}}{||\bar{\boldsymbol{\omega}}'_{n|n-1}||} sin(\frac{||\bar{\boldsymbol{\omega}}'_{n|n-1}||\Delta t_n}{2}) \end{bmatrix}$$

$$\boxed{\bar{\boldsymbol{q}}_{n|n-1} = \bar{\boldsymbol{q}}_{n-1|n-1} * \boldsymbol{\delta}^{\boldsymbol{\omega}}_{\boldsymbol{n}}}$$

$$\mathbf{F_n} = \begin{bmatrix} \mathbf{R}^T(\boldsymbol{\delta}^{\boldsymbol{\omega}}_{\boldsymbol{n}}) & I\Delta t_n \\ 0 & I \end{bmatrix}$$

$$\boxed{\mathbf{P}^{\bar{q}_{n|n-1}}_{n|n-1} = \mathbf{F}_n[\mathbf{P}^{\bar{q}_{n-1|n-1}}_{n-1|n-1} + \mathbf{Q}_n]\mathbf{F}^T_n}$$

with

$$\mathbf{Q}_n = \begin{bmatrix} Q_\omega \frac{(\Delta t_n)^3}{3} & -Q_\omega \frac{(\Delta t_n)^2}{2} \\ -Q_\omega \frac{(\Delta t_n)^2}{2} & Q_\omega \Delta t_n \end{bmatrix}$$

Measurement Prediction

$$\bar{\mathbf{a}}_{n|n-1} = \mathbf{R}^T(\boldsymbol{q}_{n|n-1})\mathbf{a_r}$$

$$\bar{\boldsymbol{\omega}}_{n|n-1} = \bar{\boldsymbol{\omega}}'_{n-1|n-1}$$

$$\boxed{\bar{\mathbf{z}}_{n|n-1} = \begin{bmatrix} \bar{\mathbf{a}}_{n|n-1} \\ \bar{\boldsymbol{\omega}}_{n|n-1} \end{bmatrix}}$$

$$\mathbf{H}_n = \begin{bmatrix} [\bar{\boldsymbol{a}}_{n|n-1}]_\times & 0 \\ 0 & I \end{bmatrix}$$

$$\boxed{\mathbf{S}_{n|n-1} = \mathbf{H}_n \mathbf{P}^{\bar{q}_{n|n-1}}_{n|n-1} \mathbf{H}_n^T + \begin{bmatrix} R_a & 0 \\ 0 & R_\omega \end{bmatrix}}$$

where $[\boldsymbol{v}]_\times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}$

Kalman Gain

$$\mathbf{K}_n = \mathbf{P}^{\bar{q}_{n|n-1}}_{n|n-1} \mathbf{H}_n^T \mathbf{S}^{-1}_{n|n-1}$$

Update

A measurement arrives at time t_n

$$\boldsymbol{z}_n = \begin{bmatrix} \boldsymbol{a}_n \\ \boldsymbol{\omega}_n \end{bmatrix}$$

$$\begin{aligned} \bar{\boldsymbol{x}}^{\bar{q}_{n|n-1}}_{n|n} &= \bar{\boldsymbol{x}}^{\bar{q}_{n|n-1}}_{n|n-1} + \mathbf{K}_n(\boldsymbol{z}_n - \bar{\boldsymbol{z}}_{n|n-1}) \\ \mathbf{P}^{\bar{q}_{n|n-1}}_{n|n} &= (I_6 - \mathbf{K}_n \mathbf{H}_n) \mathbf{P}^{\bar{q}_{n|n-1}}_{n|n-1} \end{aligned}$$

where $\bar{\boldsymbol{x}}^{\bar{q}_{n|n-1}}_{n|n-1} = (\bar{\boldsymbol{e}}^{\bar{q}_{n|n-1}}_{n|n-1} = 0, \bar{\boldsymbol{\omega}}'_{n|n-1})^T$.

Finally, we need to obtain the updated unit quaternion $\bar{\boldsymbol{q}}_{n|n}$ and compute the mean and covariance in the $\bar{\boldsymbol{q}}_{n|n}$-centered chart.

$$\begin{aligned} \bar{\boldsymbol{q}}_{n|n} &= \varphi^{-1}_{\bar{\boldsymbol{q}}_{n|n-1}}(\bar{\boldsymbol{e}}^{\bar{\boldsymbol{q}}_{n|n-1}}_{n|n}) \\ &= \bar{\boldsymbol{q}}_{n|n-1} * \varphi^{-1}(\bar{\boldsymbol{e}}^{\bar{\boldsymbol{q}}_{n|n-1}}_{n|n}) \\ &= \bar{\boldsymbol{q}}_{n|n-1} * \bar{\boldsymbol{\delta}}_n \end{aligned}$$

Knowing that the Kalman update could produce any point in the $\bar{\boldsymbol{q}}_{n|n-1}$-centered chart we will need to "saturate" to the closest point contained in the image of each chart. The point $\bar{\boldsymbol{e}}^{\bar{\boldsymbol{q}}_{n|n-1}}_{n|n}$ in the $\bar{\boldsymbol{q}}_{n|n-1}$-centered chart is the origin in the $\bar{\boldsymbol{q}}_{n|n}$-centered chart.

The reference [8] also explains the “covariance correction step” as the update of the covariance matrix to the new chart. But we shall skip this since it has also been shown in [8] that we will obtain essentially the same accuracy in our estimations with or without this update step.

2.2 3D Orientation Estimator Description

The orientation estimated by the above-described estimator is prone to drift away from the true orientation by a rotation in the yaw axis (or the azimuth rotation). This is because the accelerometer is insensitive to rotations in the horizontal plane.

2.2.1 MEKF2

To tackle this issue we now introduce a second reference vector, the magnetometer

$$\boldsymbol{m}_t = \mathbf{R}^T(\boldsymbol{q}_t)\boldsymbol{m}_r + \mathbf{r}_t^m$$
$$\boldsymbol{a}_t = \mathbf{R}^T(\boldsymbol{q}_t)\boldsymbol{a}_r + \mathbf{r}_t^a$$
$$\boldsymbol{\omega}_t = \boldsymbol{\omega}_t' + \mathbf{r}_t^\omega$$

$\mathbf{r}_t^\omega$, $\mathbf{r}_t^a$ and $\mathbf{r}_t^m$ are measurement noise with zero mean and covariance R_ω, R_a and R_m. The measurement at time t_n is

$$\boldsymbol{z}_n = \begin{bmatrix} \boldsymbol{m}_n \\ \boldsymbol{a}_n \\ \boldsymbol{\omega}_n \end{bmatrix}$$

The state and the motion equations remain unchanged. Therefore the Prediction step remains unchanged. The measurement prediction and update steps change as follows

Measurement Prediction

$$\bar{\boldsymbol{m}}_{n|n-1} = \mathbf{R}^T(\boldsymbol{q}_{n|n-1})\boldsymbol{m}_r$$
$$\bar{\boldsymbol{a}}_{n|n-1} = \mathbf{R}^T(\boldsymbol{q}_{n|n-1})\boldsymbol{a}_r$$
$$\bar{\boldsymbol{\omega}}_{n|n-1} = \bar{\omega}'_{n-1|n-1}$$

$$\bar{\boldsymbol{z}}_{n|n-1} = \begin{bmatrix} \bar{\boldsymbol{m}}_{n|n-1} \\ \bar{\boldsymbol{a}}_{n|n-1} \\ \bar{\boldsymbol{\omega}}_{n|n-1} \end{bmatrix}$$

$$\mathbf{H}_n = \begin{bmatrix} [\bar{\boldsymbol{m}}_{n|n-1}]_\times & 0 \\ [\bar{\boldsymbol{a}}_{n|n-1}]_\times & 0 \\ 0 & I \end{bmatrix}$$

$$\mathbf{S}_{n|n-1} = \mathbf{H}_n \mathbf{P}_{n|n-1}^{\bar{q}_{n|n-1}} \mathbf{H}_n^T + \begin{bmatrix} R_m & 0 & 0 \\ 0 & R_a & 0 \\ 0 & 0 & R_\omega \end{bmatrix}$$

Kalman Gain and update equations remain unchanged (the dimensions have changed)

$$\mathbf{K}_n = \mathbf{P}_{n|n-1}^{\bar{q}_{n|n-1}} \mathbf{H}_n^T \mathbf{S}_{n|n-1}^{-1}$$

$$\bar{\boldsymbol{x}}_{n|n}^{\bar{q}_{n|n-1}} = \bar{\boldsymbol{x}}_{n|n-1}^{\bar{q}_{n|n-1}} + \mathbf{K}_n(\boldsymbol{z}_n - \bar{\boldsymbol{z}}_{n|n-1})$$
$$\mathbf{P}_{n|n}^{\bar{q}_{n|n-1}} = (I_6 - \mathbf{K}_n\mathbf{H}_n)\mathbf{P}_{n|n-1}^{\bar{q}_{n|n-1}}$$

Chart update and obtaining the updated quaternion steps also remain unchanged.

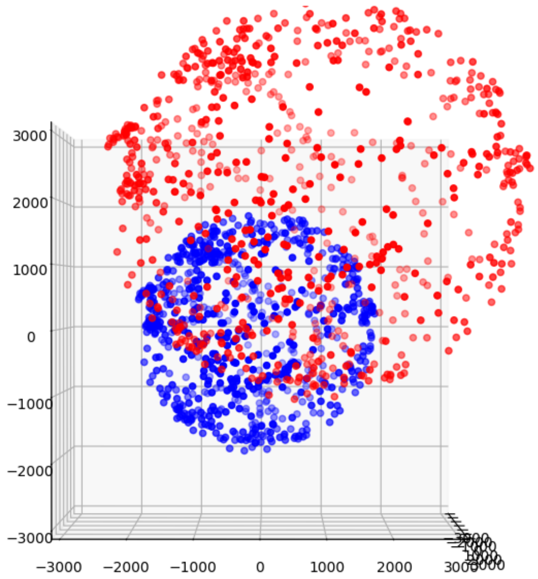


Figure 2.1: Side view of raw and calibrated magnetometer readings recorded for calibration

2.2.2 TRIAD aided MEKF2

The above implementation fixes the drift problem but introduces another issue of inconsistencies between the two reference vectors. In practice, we find that the accelerometer is relatively more consistent and measures gravity vector when not accelerating, but the magnetometer tends to give inconsistent readings at different orientations. This is also reflected in the fact that magnetometers are harder to calibrate and are easily distorted by the effects of soft and hard iron around the sensor. Figure 2.1 shows the measurements before and after calibration.

Figure 2.2 & 2.3 shows the predicted vector measurements and the actual vector measurements as a 3D needle in a ball in both body frame and inertial frame. The large residual vector (the difference between the two vectors shown) indicates the inconsistency. Figure 2.2 & 2.3 also shows the poorly estimated orientation when the object has been approximately rotated by 90 degrees in the roll axis.

This issue of magnetometer disturbance affecting the inclination (roll and pitch) estimation has been discussed in reference [13]. Reference [13] also discusses different methods of decoupling attitude (roll and pitch) estimation from magnetic disturbance. Reference [14] and [15] construct and use a new reference vector in their respective estimators. These new reference vectors are essentially the second and third columns of the TRIAD estimated while using the accelerometer for the first vector and magnetometer for the second vector.

TRIAD

$$\mathbf{c}_1 = \mathbf{a}_n$$

$$\mathbf{c}_2 = \frac{\mathbf{a}_n \times \mathbf{m}_n}{|\mathbf{a}_n \times \mathbf{m}_n|}$$

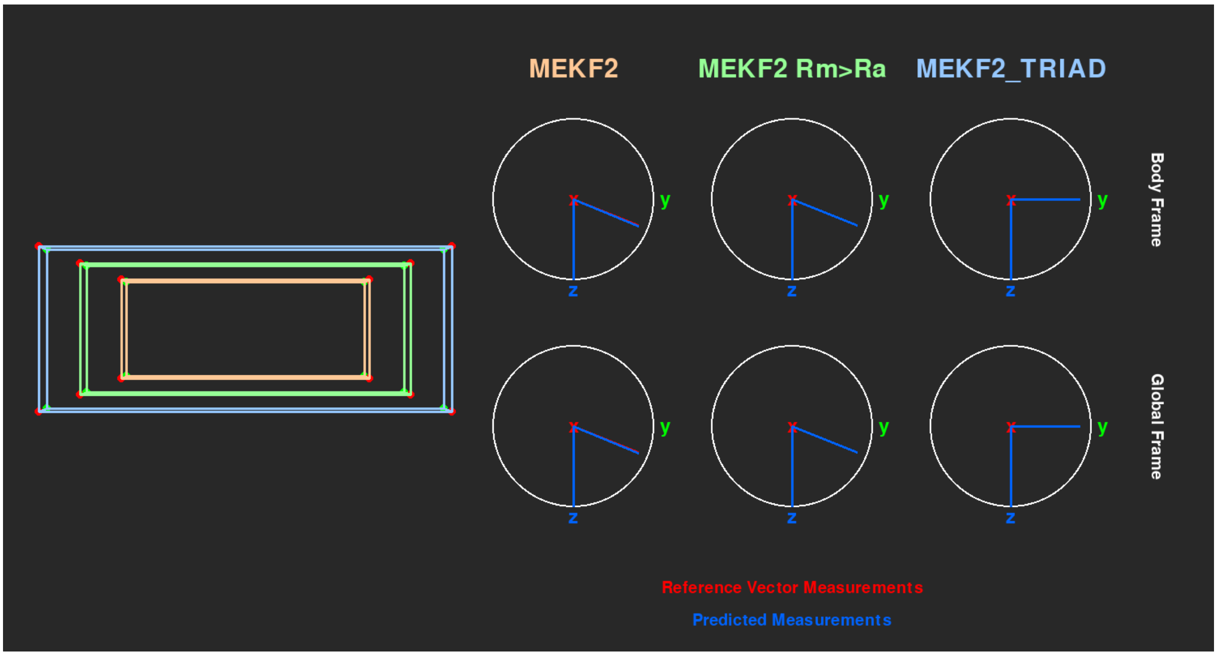


Figure 2.2: Cubes and 3D needle in sphere visualization of measured and predicted vectors; for the three estimators being compared; At identity orientation

$$\mathbf{c}_3 = \mathbf{c}_1 \times \mathbf{c}_2$$

$$\mathbf{R} = \begin{bmatrix} \mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 \end{bmatrix}$$

This version of TRIAD is sub-optimal because it ignores one piece of information from one of the unit vectors. But this works in our favor since this is exactly how we decouple the attitude (roll and pitch) estimation from magnetic disturbance.

In the MEKF2 estimator described above we replace m_n with

$$m_n = c_3$$

from the TRIAD. Alternatively, the second column of the TRIAD can also be used.

2.3 Experimental Setup and Observations

2.3.1 Hardware Description

Figure 2.4 shows the hardware used in the implementation. This sensor module allows both I2C and SPI protocols to access the sensor registers.

2.3.2 Firmware Implementation Details

The code for the estimator can be found in https://github.com/arjun-sadananda/TeensyPilot.git. The details of this firmware are explained in the next chapter. For this section, the main two modules of this firmware that are relevant are the TP_Sense and TP_Estimator classes.

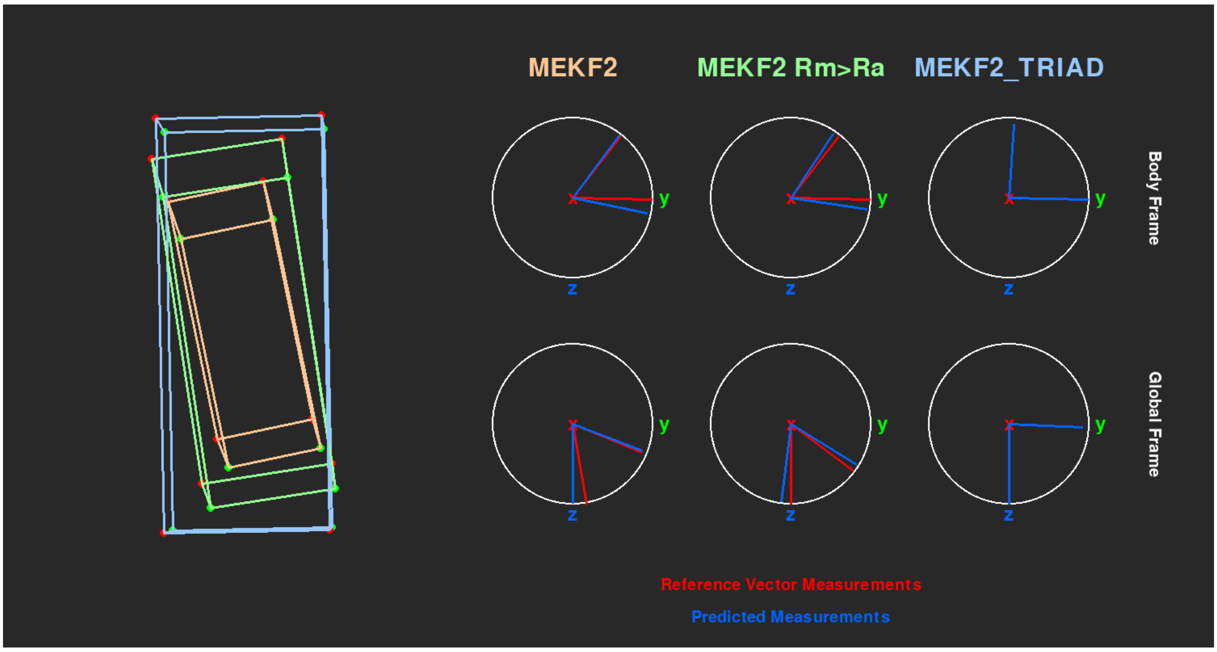


Figure 2.3: Cubes and 3D needle in sphere visualization of measured and predicted vectors; for the three estimators being compared; At 90 degree roll

LSM9DS1 Module	
Teensy 4.1 Developement Board	

Figure 2.4: Estimator Hardware Components

When it comes to reading sensors, we have both protocols (I2C and SPI) implemented in this firmware. When all three x three sensors are being read the reading time for I2C is about $\tilde{6}$00 microseconds with 400kHz clock and is about $\tilde{2}$5 microseconds with 8MHz clock SPI read. Now coming to the hardware implementation of the estimator following are some points to be noted:

- The a_r and m_r reference vectors are found during the calibration phase before the start of the estimator. This is done by taking an average of 3000 readings for each sensor.

- Q_ω R_ω R_a R_m are diagonal matrices with all diagonal entries equal, therefore we shall only declare these variables as scalars in the implementation. We set $Q_w = 10.0$, $R_w = 0.001$, $R_a = 0.01$, $R_m = 0.01$ unless stated otherwise.

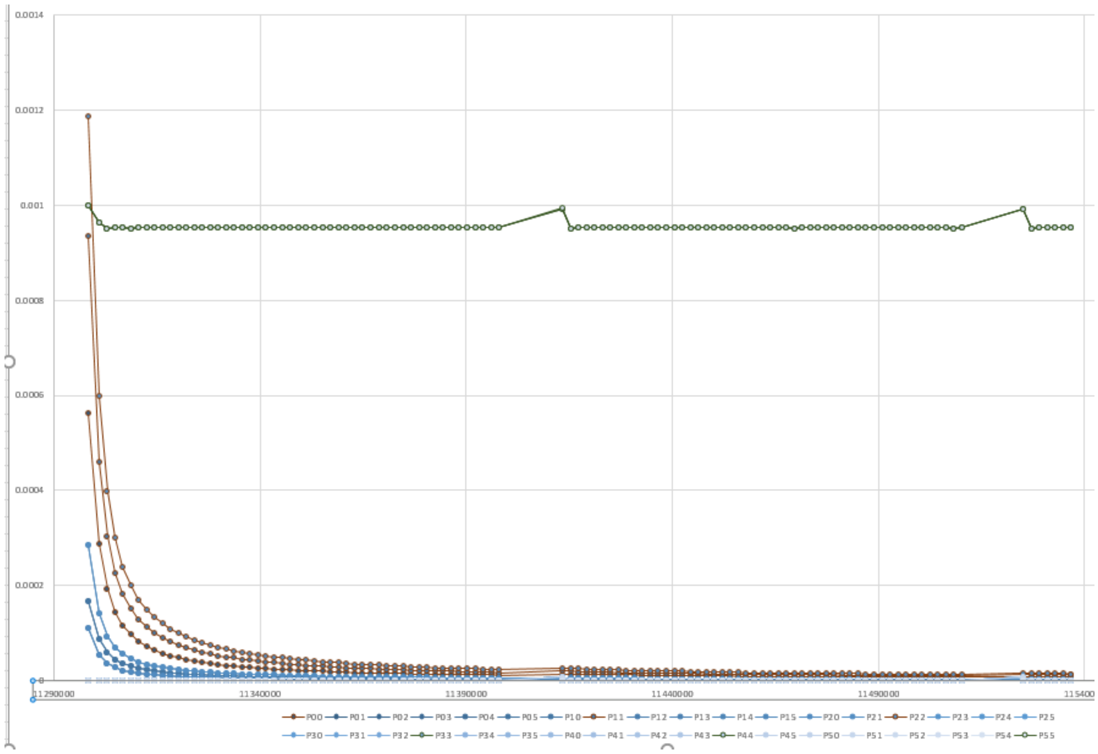


Figure 2.5: State Covariance - time plot for MEKF2

2.3.3 Observations

The experiment compares the following three cases,

- MEKF2
- MEKF2 with $Rm = 5.0e - 2 = 0.05$
- TRIAD aided MEKF2

Figures 2.5 2.6 2.7 show the evolution of the components of the state covariance matrices in the three cases. As expected the settling time for the second case with Rm>Ra is significantly larger than the other two cases. We also observe that Kalman filter reaches steady state in about .1 seconds.

Figure 2.2 and 2.3 show the estimator at two different orientations for all the three cases. It is clear from the 3D needle in sphere visual that there is a large non zero residual when the orientation is at 90 degree roll from identity orientation in the case of MEKF without the use of TRIAD. It is also clear from the cube visualization that the estimated orientation is far from the true orientation of about 90 degrees roll.

Challenges in Estimation during Flight

https://github.com/plusk01/adaptive-gyro-filtering

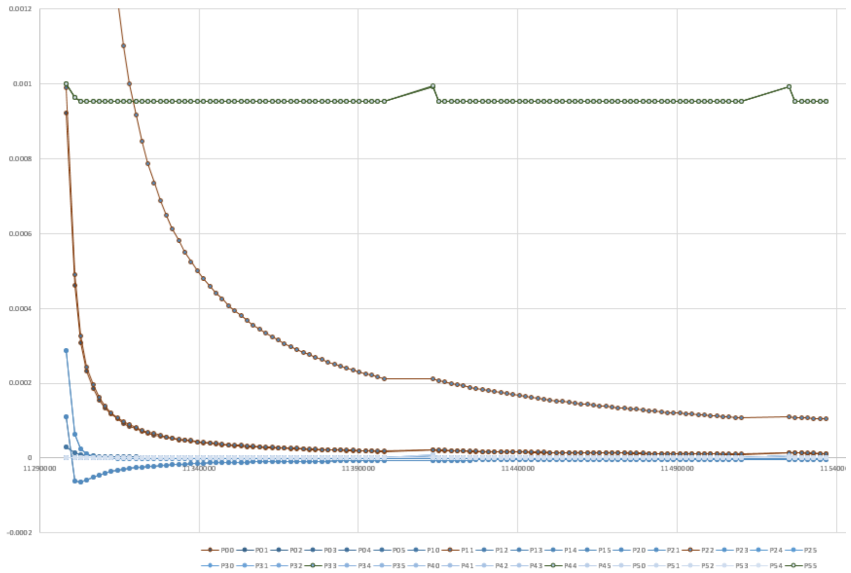


Figure 2.6: State Covariance - time plot for MEKF2 with larger measurement noise covariance for magnetometer than accelerometer

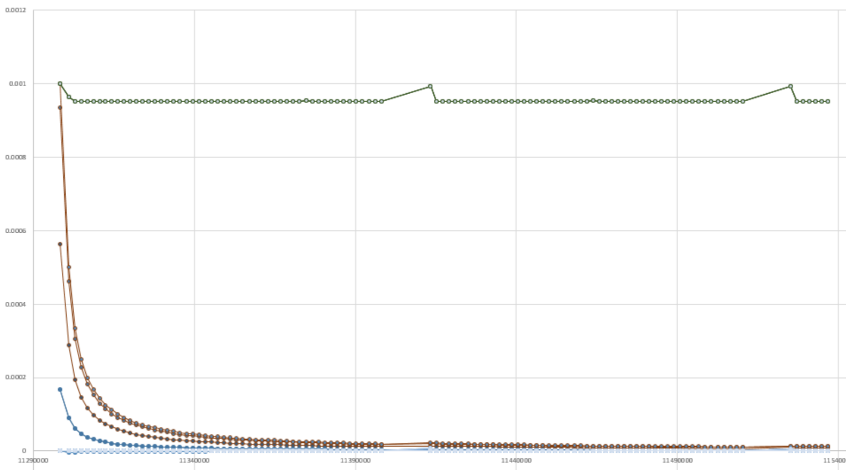


Figure 2.7: State Covariance - time plot for TRIAD aided MEKF

Signal conditioning is an important part of a control strategy. In the context of multirotor control, the on-board gyro provides angular rate measurements which are directly used to drive the actual angular rate to the desired angular rate. In this setting, gyro noise is fed into the actuators, creating visible attitude fluctuations (i.e., "wiggles") even when the vehicle is meant to be hovering. Additionally, noisy gyro signals are directly related to hot motors (which can lead to motor failure, but is also an indication of performance). The majority of gyro noise comes from vibrations, caused by the spinning of the motors.

We shall discuss this further in the next chapter.

Chapter 3

Geometric Attitude tracking control on SE(3)

3.1 Definitions and Model

In this section, we introduce the reader to basic notation and terminology used in quadrotor modeling and control. We also map the available physical inputs to the control variables used in our algorithms.

Define:

$m \in \mathbb{R}$	the total mass
$d \in \mathbb{R}$	distance between motor and center of mass
$J \in \mathbb{R}^3$	the inertia matrix w.r.t body-fixed frame

$\mathbf{x} = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix}^T \in \mathbb{R}^3$

the position vector of the center of mass in the inertial frame

$R = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} \in \mathsf{SO}(3)$

the rotation matrix from the body-fixed frame to the inertial frame. b_i is the transformed body frame coordinates represented in the NED frame coordinates {North-East-Down Frame}

$\mathbf{v} = \begin{pmatrix} v^1 & v^2 & v^3 \end{pmatrix}^T \in \mathbb{R}^3$

the velocity vector of the center of mass in the inertial frame

$\Omega = \left(\omega^1, \omega^2, \omega^3\right)^T \in \mathbb{R}^3$

the angular velocity in the body-fixed frame

The configuration of this quadrotor UAV is defined by the location of the center of mass and the attitude with respect to the inertial frame. Therefore, the configuration manifold is the special Euclidean group SE(3), which is the semidirect product of $\mathbb{R}^3$ and the special orthogonal group $\mathsf{SO}(3) = \{R \in \mathbb{R}^{3x3} | R^T R = I, \det R = 1\}$. The mapping of Ω from $\mathbb{R}^3$ to $\mathfrak{so}(3)$ (the lie algebra of $\mathsf{SO}(3)$) is defined by the *hat map*.

$$\hat{\Omega} = \begin{bmatrix} 0 & -\omega^3 & \omega^2 \\ \omega^3 & 0 & -\omega^1 \\ -\omega^2 & \omega^1 & 0 \end{bmatrix} \in \mathfrak{so}(3)$$

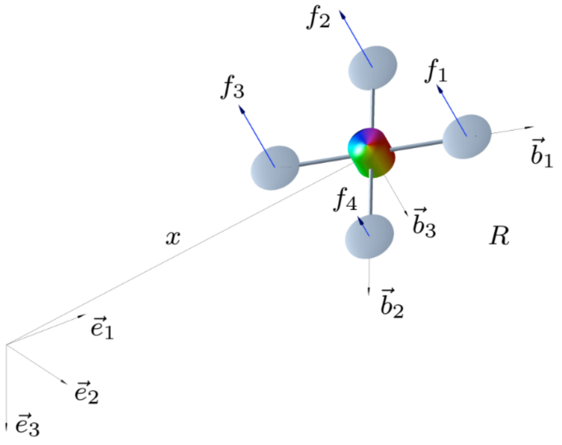


Fig. 1. Quadrotor model

Figure 3.1: Quadcopter Model

The *hat map* $\hat{}: \mathbb{R}^3 \rightarrow \mathfrak{so}(3)$ is defined by the condition that $\hat{x}y = x \times y$ for all $x, y \in R$.

Control Input Remapping (Kinematics)

We assume that the thrust of each propeller is directly controlled, and the direction of the thrust of each propeller is normal to the quadrotor plane. The first and third propellers are assumed to generate a thrust along the direction of -b3 when rotating clockwise; the second and fourth propellers are assumed to generate a thrust along the same direction of -b3 when rotating counterclockwise.
The mapping from individual motor thrusts f_i to overall thrust f and moments $M \in \mathbb{R}^3$ is given by

$$\begin{bmatrix} f \\ M_1 \\ M_2 \\ M_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -d & 0 & d \\ d & 0 & -d & 0 \\ -c_{\tau f} & c_{\tau f} & -c_{\tau f} & c_{\tau f} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}$$

This mapping is invertible when $8c_{\tau f}d^2 \neq 0$ which is true since d and $c_{\tau f}$ are not zero. Therefore we use f and $M \in \mathbb{R}^3$ as the control inputs. A detailed explanation of the assumptions made and the conventions used to generate this mapping can be found in [1]. For the × motor configuration used in this project (instead of the + motor configuration explained above) the mapping changes to

$$\begin{bmatrix} f \\ M_1 \\ M_2 \\ M_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ d' & -d' & -d' & d' \\ d' & d' & -d' & -d' \\ -c_{\tau f} & c_{\tau f} & -c_{\tau f} & c_{\tau f} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}$$

where $d' = \frac{d}{\sqrt{2}}$.

Equations of Motion

The equations of motion of the quadrotor UAV can be written as

$$\dot{x} = v \tag{3.1}$$

$$m\dot{v} = mge_3 - fRe_3 \tag{3.2}$$

$$\dot{R} = R\hat{\Omega} \tag{3.3}$$

$$J\dot{\Omega} + \Omega \times J\Omega = M \tag{3.4}$$

3.2 Control Objectives

Attitude Tracking

The quadcopter is controlled by a pilot manually. The pilot provides roll, pitch, yaw angles and thrust. The angles are then mapped to an attitude tracking command $R_d(t) \in \mathsf{SO}(3)$. The control objective is to achieve

$$\lim_{t\to\infty} R(t) = R_d(t) \tag{3.5}$$

Stabilization under harsh initialization

The quadcopter is manually thrown into the air and the controller is initialized mid-air. Thus initializing the system with some

$$\mathbf{x}(0) = \mathbf{x_0} \in \mathbb{R}^3; \quad R(0) = R_0 \in \mathsf{SO}(3)$$

$$\mathbf{v}(0) = \mathbf{v_0} \in \mathbb{R}^3; \quad \Omega(0) = \Omega_0 \in \mathbb{R}^3$$

The control objective is to achieve a stable hover:

$$\lim_{t\to\infty} x_3(t) = x_{3d} \in \mathbb{R} \tag{3.6}$$

$$\lim_{t\to\infty} v(t) = 0 \in \mathbb{R}^3 \tag{3.7}$$

This control objective must be achieved using feedback from only the onboard sensors available on the quadcopter. The onboard sensors comprise an Inertial Measurement Unit - IMU (a 3-axis gyroscope, a 3-axis accelerometer, and a 3-axis magnetometer), a barometer, a time-of-flight sensor, and an optical flow sensor.

3.3 Controller Description

3.3.1 Stabilization under harsh initialization

The control strategy has two stages.
Stage 1: Stabilize Attitude and Altitude, where the objectives are:

$$\lim_{t\to\infty} R(t) = I \in \mathsf{SO}(3) \tag{3.8}$$

$$\lim_{t\to\infty} x_3(t) = x_{3d} \in \mathbb{R} \tag{3.9}$$

We shall see that these two objectives are decoupled and therefore can be developed and tested independently.

The exponential stability of the "attitude-controlled flight mode" under some initial conditions (almost global) has been shown in [1]. Therefore $R(t)$ will converge to I. When $R(t)$ and $\Omega(t)$ are close to I and 0 respectively Stage 2 is enabled. The objective in Stage 2: Stabilize Linear Velocity and Altitude, is exactly as defined in equations 3.6 & 3.7.

3.3.1.1 Stage 1: Stabilize Attitude and Altitude

To design the controller for the objective defined in 3.8, we first define errors associated with the attitude dynamics of the quadrotor UAV. The **Attitude Tracking Error** and the **Angular Velocity Tracking Error** for an arbitrary smooth attitude tracking command $R_d(t) \in \mathsf{SO}(3)$ is defined in [1].

$$e_R = \frac{1}{2}(R_d^T R - R^T R_d)^\vee \tag{3.10}$$

$$e_\Omega = \Omega - R^T R_d \Omega_d \tag{3.11}$$

where, $\hat{\Omega_d} = R_d^T \dot{R_d}$. Setting the control command, $R_d(t)$ to I, we get

$$e_R = \frac{1}{2}(R - R^T)^\vee$$

$$e_\Omega = \Omega$$

The *vee map* $\vee : \mathfrak{so}(3) \to \mathbb{R}^3$ is the inverse of the hat map. The nonlinear controller for this stage is:

$$M = -k_R e_R - k_\Omega e_\Omega + \Omega \times J\Omega \tag{3.12}$$

where k_R and k_Ω are positive constants.

In this attitude-controlled mode, it is possible to ignore the translational motion of the quadrotor UAV; consequently the reduced model for the attitude dynamics is given by equations 3.18, 3.19, using the controller expression 3.15. It has been shown in [1] that if the initial conditions satisfy

$$\psi(R_0) < 2$$

$$||\Omega_0|| < \frac{2}{\lambda_M(J)} k_R(2-\psi(R_0))$$

then $(e_R, e_\Omega) = (0,0)$ is an exponentially stable equilibrium of the reduced closed loop dynamics.

In addition, asymptotically tracking a quadrotor altitude (3.9) is achieved by the following control law for the thrust magnitude:

$$f = \frac{k_z(x_3 - x_{3d}) + k_v \dot{x}_3 + mg}{e_3.Re_3}$$

where k_z and k_v are positive constants. Exponential stability of attitude-controlled flight mode with altitude tracking has been shown in [1].

3.3.1.2 Stage 2: Stabilize Linear Velocity and Altitude

To design the controller for the objective defined in 3.6 and 3.7, we fuse the ideas of position-controlled flight mode and velocity-controlled flight mode described in [1]. The **position tracking error** and the **velocity tracking error** are defined as follows:

$$e_x = (x_3 - x_{3d})e_3$$

$$e_v = v$$

The nonlinear controller for this stage, described by control expressions for the thrust magnitude and the moment vector, are:

$$f = (k_x e_x + k_v e_v + mge_3).Re_3$$

$$M = -k_R e_R - k_\Omega e_\Omega + \Omega \times J\Omega$$

where k_x, k_v, k_R, k_Ω are positive constants. Following the prior definition of the attitude error and the angular velocity error 3.13, 3.14

$$e_R = \frac{1}{2}(R_c^T R - R^T R_c)^\vee$$

$$e_\Omega = \Omega - R^T R_c \Omega_c$$

and $R_c(t) \in \mathsf{SO}(3)$ and $\Omega_c \in \mathbb{R}^3$ are given by

$$R_c = [b_{1_c}; b_{3_c} \times b_{1_c}; b_{3_c}], \quad \hat{\Omega}_c = R_c^T \dot{R}_c$$

where, $b_{3_c} \in \mathsf{S}^2$ is defined by

$$b_{3_c} = -\frac{-F}{||-F||}$$

where, $F = k_x e_x + k_v e_v + mge_3 \in \mathbb{R}^3$. b_{1_c} is selected to be orthogonal to b_{3_c}, thereby guaranteeing $R_c \in \mathsf{SO}(3)$. We assume that $||-F|| \neq 0$. We set b_{1_d} to e_1, since the condition to switch to this (Stage 2) controller (from Stage 1) already ensures that b_1 is "close to" e_1. Now b_{1_c} is constructed by projecting b_{1_d} onto plane normal to b_{3_c} and normalising it:

$$b_{1_c} = -\frac{1}{||b_{3_c} \times b_{1_d}||}(b_{3_c} \times (b_{3_c} \times b_{1_d}))$$

3.3.2 Attitude Tracking

This controller has already been defined above. For an arbitrary smooth attitude tracking command $R_d(t) \in \mathsf{SO}(3)$

$$e_R = \frac{1}{2}(R_d^T R - R^T R_d)^\vee \tag{3.13}$$

$$e_\Omega = \Omega - R^T R_d \Omega_d \tag{3.14}$$

where, $\hat{\Omega}_d = R_d^T \dot{R}_d$. The nonlinear controller is:

$$M = -k_R e_R - k_\Omega e_\Omega + \Omega \times J\Omega \tag{3.15}$$

The only new point to be noted is that the transmitter roll, pitch, yaw angle commands need to be mapped to a $R_d(t) \in \mathsf{SO}(3)$ by one of the standard Euler to rotation matrix mapping. Note that the sensitivity and limits for the manual pilot can also be adjusted in this step.

3.4 Hardware Implementation Details and Observations

3.4.1 Hardware Setup

Table 3.1 lists all the components used in building the quadcopter. The figure shows the assembled product. For building the circuit board, a perf board was used. For mounting different parts on the drone, 3D printing, and acrylic sheets were used. Figure 3.2 shows the wiring diagram for all the electronic components used on the drone. This is using the LSM9DS1 in SPI mode. Figure 3.3 shows the fully assembled quadcopter. This is the second version of the quadcopter that has major upgrades and additions over the first version like switching from I2C to SPI and interfacing the ELRS Receiver.

3.4.2 Flight Controller Firmware Architecture

Figure 3.4 is a simplified flow chart of the Firmware Architecture. There is no parallelism of any sort implemented either through RTOS or through threading. All the tasks are performed in series as shown in the flowchart. This loop runs at 9174Hz when SPI is used for reading the sensor measurements and no displaying or SD card logging is done.

The latest version of this firmware can be found at https://github.com/arjun-sadananda/TeensyPilot.git

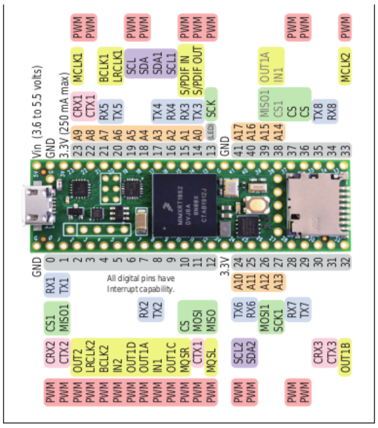
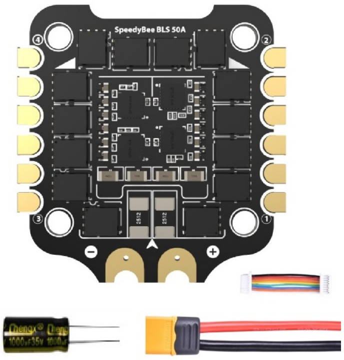
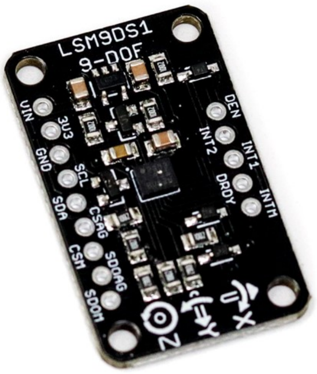
FRAME: GEPRC CL35 Cinelog 35 V2 	MICROCONTROLLER: Teensy 4.1 Development Board 	MOTOR: SPEEDX2 2105.5-2650KV(6S)
PROPELLERS: HQProp D-T90MM 	ESC: SpeedyBee 50A 30×30 4-in-1 	BUCK CONVERTER: 7–26V to 5V 3A BEC
MAG,ACC,GYRO SENSOR: LSM9DS1 	ELRS TRANSMITTER AND RECEIVER 	DISPLAY 2.4-inch 240×320 TFT

Table 3.1: Hardware Components

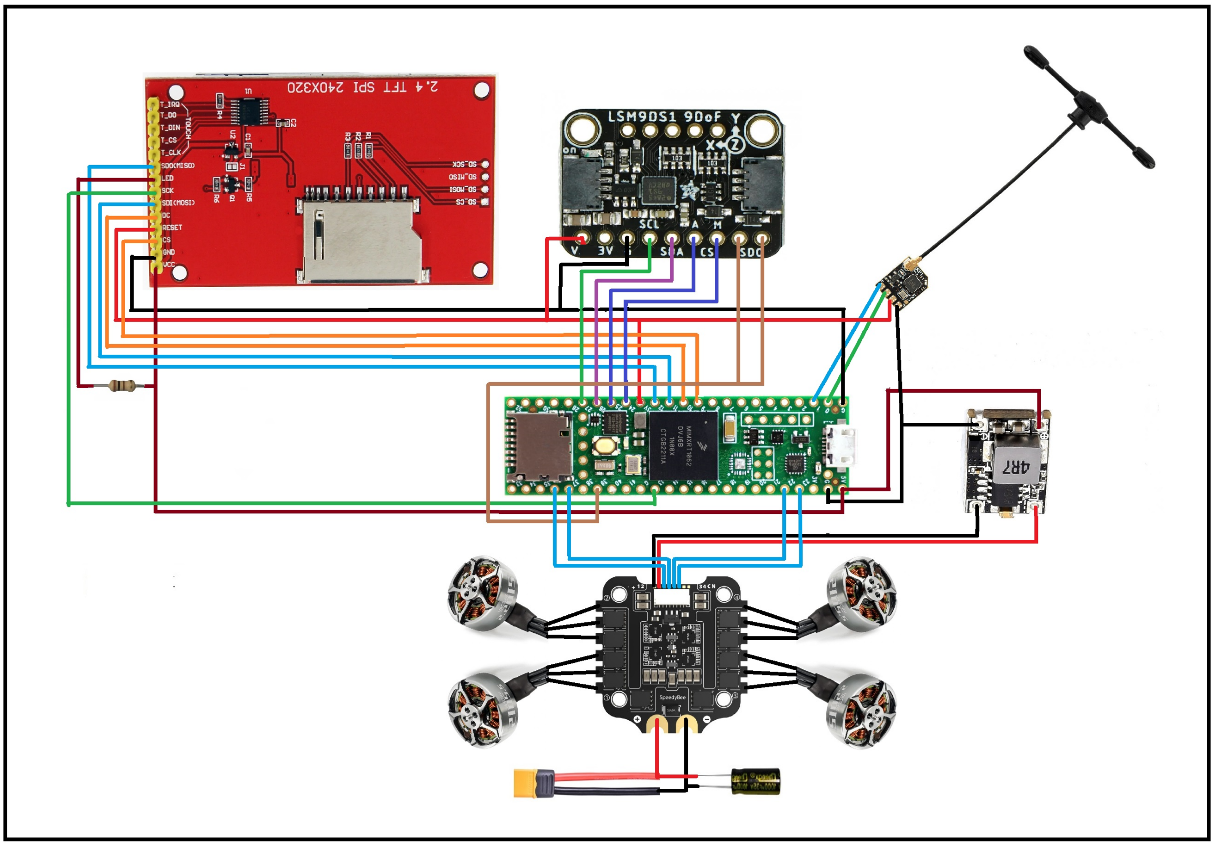


Figure 3.2: Electronic Wiring Diagram

3.4.3 Observations - Challenges and Solutions in Practice

The firmware developed that implements the estimator and controller described achieves the desired result of stable flight in Attitude Tracking Mode. The pilot can command a thrust and a desired orientation ($R_d(t)$) and the controller successfully tracks the desired orientation. The stabilisation under harsh initial conditions still needs tuning and testing to achieve the desired result, this is left as future scope.

There are some important observations made during the development and we shall go over them in this section.

Disturbances - Integral/ Manual Compensation

In practice, we find multiple sources of fixed disturbances that the above controller doesn't consider. These disturbances cause the quadcopter to stabilize at an attitude different from the commanded attitude.

Therefore the equations of motion should involve the fixed uncertainties in the translational dynamics and the rotational dynamics denoted by Δ_x and $\Delta_R \in \mathbb{R}^3$, respectively in the following equations of motion

$$\dot{x} = v \tag{3.16}$$

$$m\dot{v} = mge_3 - fRe_3 + \Delta_x \tag{3.17}$$

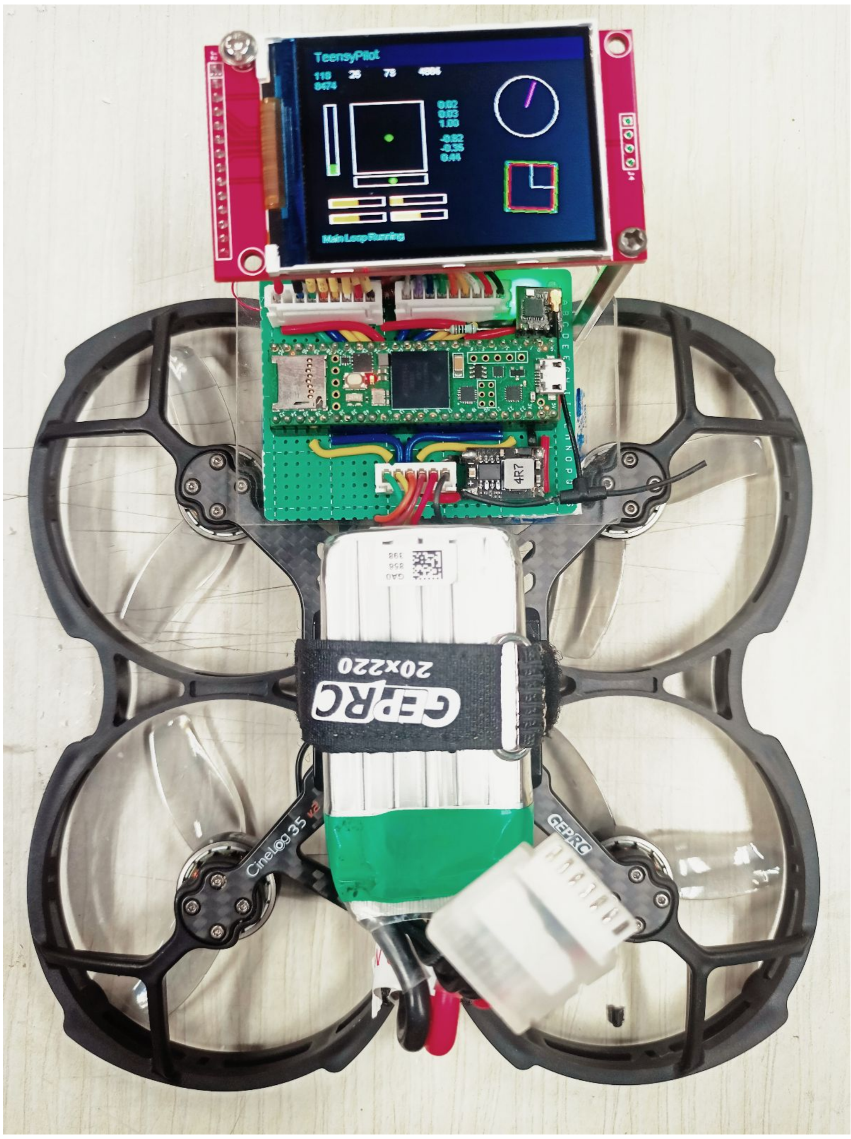


Figure 3.3: Fully Assembled Quadcopter

$$\dot{R} = R\hat{\Omega} \tag{3.18}$$

$$J\dot{\Omega} + \Omega \times J\Omega = M + \Delta_R \tag{3.19}$$

Figure 3.5 demonstrates the effect of disturbance on the controller performance. This plot is logged from a flight after some manual compensation is included to compensate for the difference in the thrust produced by the motors. The desired orientation commanded is identity (1,0,0,0) quaternion, but the quadcopter stabilizes at an orientation different from identity despite already including some compensation for each motor.

The two methods used to tackle this issue by manually tuning a scaling or offset parameter for each motor to compensate for the difference in thrust produced by each motor. A cleaner solution to this problem is to add an integrator that integrates the e_R, which does tackle the issue when thrust is close to hover, but causes new issues while manually piloting. If the controller is turned on while still not in flight, we face problems related to integral

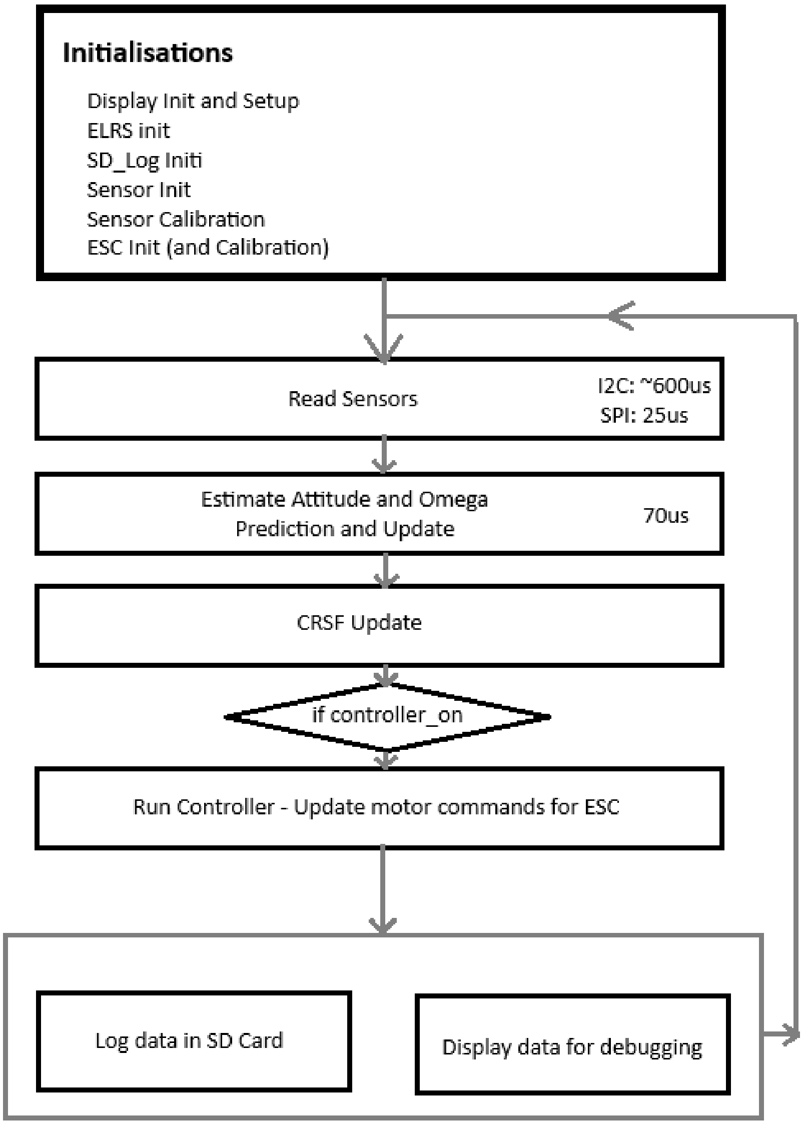


Figure 3.4: Firmware: Flowchart

wind-up.

Therefore for the manually piloted attitude tracking problem, more care must be taken while introducing the intregator. The final solution to this problem is yet to be found and is left for future scope. At present, only manual compensation is being used in the released firmware.

Loop Rate vs Control Rate - OneShot42

There are multiple options for the protocol used to send motor rate commands to the ESC. The first flight controller use the standard PWM, where the motor throttle percentage is encoded as pulse width of 1000 microseconds (for 0% throttle) to 2000 microseconds (or 100% throttle). This means the fastest rate at which the ESC can receive commands is at

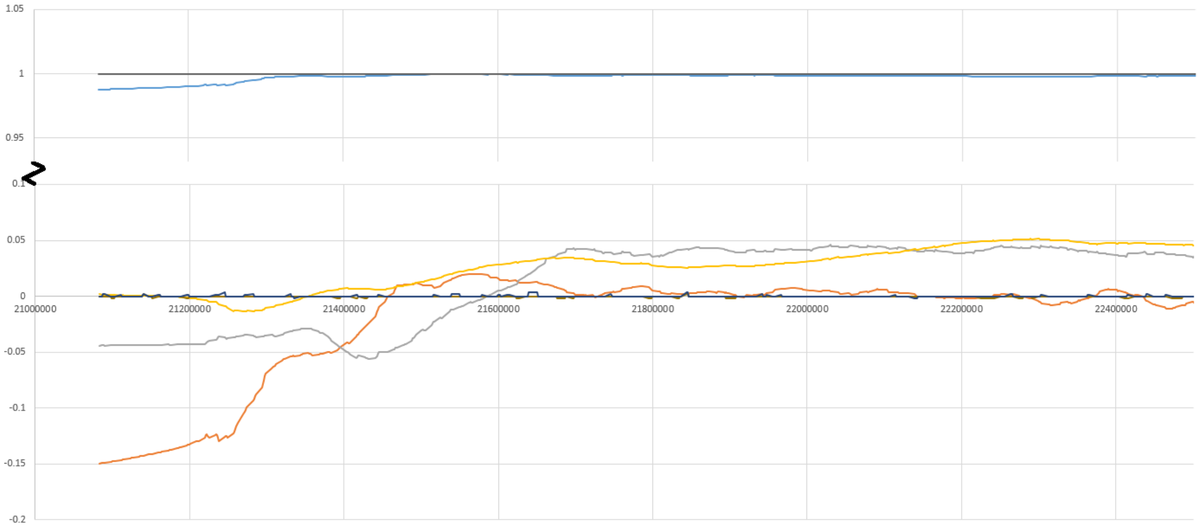


Figure 3.5: Effect of Disturbance: estimated quaternion, desired quaternion vs time

490Hz (by default it runs at 50Hz). Therefore even if our flight controller loop is 9kHz, the ESC will only receive commands at 490Hz. To tackle this there are many new protocols that the ESC understands. Oneshot125, Oneshot42 and Multishot are three such protocols that are analog and store the speed in pulse width, but the time period varies from 25us to 250us. Since our controller runs at 9kHz the protocol chosen is OneShot42 which can have a theoretical max rate of 12kHz.

In our implementation the we simply generate the PWM signals at a fixed rate of 10kHz, but the proper way to implement OneShot42 is to sync it with the rate at which the motor commands are generated. This is left as a future scope. An alternate upgrade to DShot600, which is a digital protocol is also left as a future scope.

Gyro (RPM) Noise - Adaptive Notch Filter

Upgrading the ESC protocol from Standard PWM to OneShot42 at 12kHz revealed an underlying issue that went unnoticed while using the slower standard PWM at 50Hz. The motor rpm induces noise into the gyro measurements which then propagates to the controller output. This causes the jittering of the motors.

Figure 3.6 and 3.7 show the omega estimated and the control command sent to the motors. The standard approaches to tackle this issue are filtering the gyro measurements. There are two major filters that are usually implemented. First is the RPM filter, which takes the rpm measurements from the ESC and applies a filter to counter the possible noise entering the gyro due to the motor rpm. The second filter is an adaptive notch filter that does a FFT dynamically and applies a notch filter at the noise frequency identified from the FFT. This is yet to be implemented and is left as a future scope.

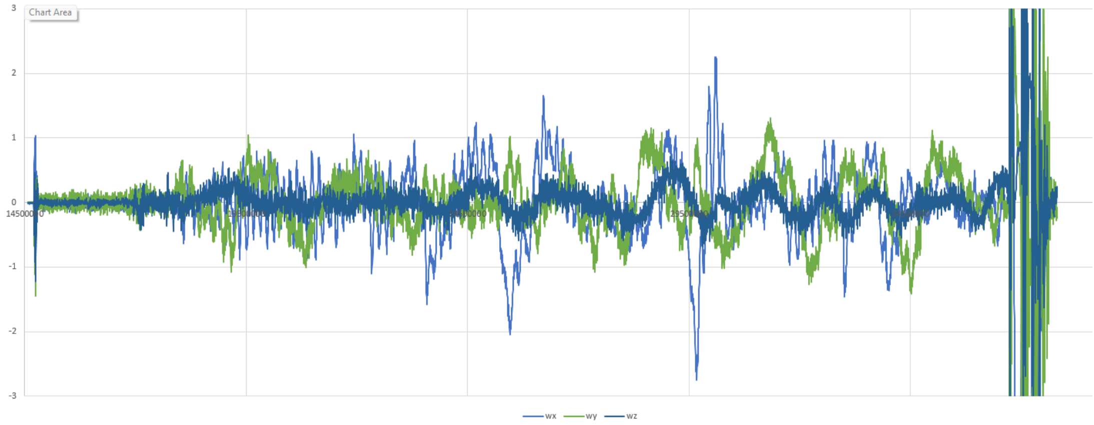


Figure 3.6: Effect of motor rpm on gyro measurement: ω vs time

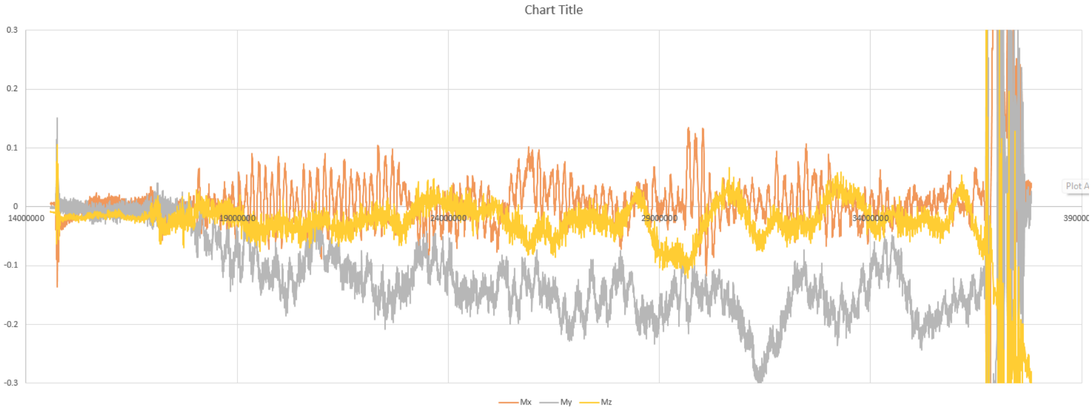


Figure 3.7: Effect of ω estimate on control: M vs time

Chapter 4

Conclusion and Future work

We have successfully developed a flight controller from scratch that implements Geometric Attitude Tracking Flight mode on hardware. We also implemented a robust Manifold EKF estimator that decouples magnetometer disturbances from Attitude (roll, pitch) estimation. The robustness of this estimator was tested by using these attitude estimates for Attitude Tracking and was found to be robust against the (inertial and magnetic) noises and disturbances caused by the quadcopter. We also did a few test runs with initializing the quadcopter with soft and harsh initialization.

The flights revealed some very interesting ch allenges that need to be tackled before moving to the next phase of implementing Stage 2 of Stabilisation under harsh initialization. The immediate challenges that need to be tackled before moving to the next phase of this project are: - the disturbances in dynamics. - implementing adaptive notch filter or RPM filter for gyro noise induced by motor rpm.

After this is done, the next step would be to run the controller under harsh initialization, collect logs, and tune the controller till we achieve reliable attitude stabilization. Finally, we can move on to the next phase of introducing a 1D LIDAR, altitude estimators, and optical flow sensors to implement Stage 2 of Stabilization under harsh initialisation. This will make the quadcopter completely autonomous.

At this stage, we would reach a point where a comparative study can be made between the Throw mode of ArduCopter against this geometric controller.

References

[1] T. Lee, M. Leok, and N. H. McClamroch, “Control of complex maneuvers for a quadrotor uav using geometric methods on se (3),” *arXiv preprint arXiv:1003.2005*, 2010. 1, 3, 15, 17, 18

[2] M. Valenti, B. Bethke, G. Fiore, J. How, and E. Feron, “Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery,” in *AIAA guidance, navigation, and control conference and exhibit*, p. 6200, 2006. 1

[3] P. Pounds, R. Mahony, and P. Corke, “Modelling and control of a large quadrotor robot,” *Control Engineering Practice*, vol. 18, no. 7, pp. 691–699, 2010. 1

[4] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin, “Quadrotor helicopter flight dynamics and control: Theory and experiment,” in *AIAA guidance, navigation and control conference and exhibit*, p. 6461, 2007. 1

[5] P. Castillo, R. Lozano, and A. Dzul, “Stabilization of a mini-rotorcraft having four rotors,” in *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3, pp. 2693–2698, IEEE, 2004. 1

[6] N. Guenard, T. Hamel, and V. Moreau, “Dynamic modeling and intuitive control strategy for an" x4-flyer",” in *2005 International Conference on Control and Automation*, vol. 1, pp. 141–146, IEEE, 2005. 1

[7] S. Bouabdallah and R. Siegwart, “Backstepping and sliding-mode techniques applied to an indoor micro quadrotor,” in *Proceedings of the 2005 IEEE international conference on robotics and automation*, pp. 2247–2252, IEEE, 2005. 1

[8] P. Bernal-Polo and H. Martínez-Barberá, “Kalman filtering for attitude estimation with quaternions and concepts from manifold theory,” *Sensors*, vol. 19, no. 1, p. 149, 2019. 2, 3, 4, 7

[9] I. Bar-Itzhack and Y. Oshman, “Attitude determination from vector observations: Quaternion estimation,” *IEEE Transactions on Aerospace and Electronic Systems*, no. 1, pp. 128–136, 1985. 2

[10] E. J. Lefferts, F. L. Markley, and M. D. Shuster, “Kalman filtering for spacecraft attitude estimation,” *Journal of Guidance, control, and Dynamics*, vol. 5, no. 5, pp. 417–429, 1982. 2

[11] F. L. Markley, “Attitude error representations for kalman filtering,” *Journal of Guidance, Control, and Dynamics*, vol. 26, pp. 311–317, 2003. 2, 4

[12] M. W. Mueller, M. Hehn, and R. D'Andrea, "Covariance correction step for kalman filtering with an attitude," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 9, pp. 2301–2306, 2017. 2

[13] B. Fan, Q. Li, and T. Liu, "How magnetic disturbance influences the attitude and heading in magnetic and inertial sensor-based orientation estimation," *Sensors (Switzerland)*, vol. 18, 1 2018. 2, 9

[14] P. Martin and E. Salaun, "Invariant observers for attitude and heading estimation from low-cost inertial and magnetic sensors," in *2007 46th IEEE Conference on Decision and Control*, pp. 1039–1045, IEEE, 2007. 2, 9

[15] Y. Chen and H. Rong, "A customized extended kalman filter for removing the impact of the magnetometer's measurements on inclination determination," *Sensors*, vol. 23, 12 2023. 2, 9

[16] S. O. Madgwick, S. Wilson, R. Turk, J. Burridge, C. Kapatos, and R. Vaidyanathan, "An extended complementary filter for full-body marg orientation estimation," *IEEE/ASME Transactions on Mechatronics*, vol. 25, pp. 2054–2064, 8 2020. 2

[17] J. Wu, Z. Zhou, J. Chen, H. Fourati, and R. Li, "Fast complementary filter for attitude estimation using low-cost marg sensors," *IEEE Sensors Journal*, vol. 16, pp. 6997–7007, 9 2016. 2

[18] Y. S. Suh, "Attitude estimation using inertial and magnetic sensors based on hybrid four-parameter complementary filter," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, pp. 5149–5156, 7 2020. 2

[19] M. D. Shuster and S. D. Oh, "Three-axis attitude determination from vector observations," *Journal of guidance and Control*, vol. 4, no. 1, pp. 70–77, 1981. 2

[20] F. L. Markley, "Optimal attitude matrix from two vector measurements," *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 3, pp. 765–768, 2008. 2

[21] H. Kinatas and C. Hajiyev, "Triad-aided multiplicative ekf for small satellite attitude estimation and magnetometer calibration," *IEEE Sensors Journal*, vol. 23, pp. 27161–27168, 11 2023. 3