BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

A Data Driven Toggling Gain Complementary Filtering Approach for Orientation Estimation

Samnun Azfar
200041130
Ramisa Zaman Audhi
200041131
Mir Md Inzamam
200041240

Department of Computer Science and Engineering
Islamic University of Technology
May, 2025

A Data Driven Toggling Gain Complementary Filtering Approach for Orientation Estimation

Samnun Azfar
200041130
Ramisa Zaman Audhi
200041131
Mir Md Inzamam
200041240

Department of Computer Science and Engineering
Islamic University of Technology
May, 2025

Declaration of Candidate

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by **Samnun Azfar**, **Ramisa Zaman Audhi**, and **Mir Md Inzamam** under the supervision of **Dr. Abu Raihan Md. Mostafa Kamal**, Professor, Department of Computer Science and Engineering and co-supervision of **Mohammad Ishrak Abedin**, Lecturer, Department of Computer Science and Engineering, Islamic University of Technology, Dhaka, Bangladesh. It is also declared that neither this thesis nor any part of it has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others have been acknowledged in the text and a list of references is given.

Dr. Abu Raihan Md. Mostafa Kamal
Professor
Department of Computer Science and Engineering
Islamic University of Technology (IUT)
Date: May 29, 2025

Samnun Azfar
Student ID: 200041130
Date: May 29, 2025

Ramisa Zaman Audhi
Student ID: 200041131
Date: May 29, 2025

Mohammad Ishrak Abedin
Lecturer
Department of Computer Science and Engineering
Islamic University of Technology (IUT)
Date: May 29, 2025

Mir Md Inzamam
Student ID: 200041240
Date: May 29, 2025

Dedicated to the cats of IUT

Contents

List of Figures

List of Tables

List of Abbreviations

MARG	Magnetic Angular-Rate Gravity
INS	Inertial Navigation System
CF	Complemetary Filtering/Complemetary Fusion
EKF	Extended Kalman Filter
ML	Machine Learning
BROAD	Berlin Robust Orientation Assessment Dataset
RANSAC	RANdom SAmple Consensus
IMU	Inertial Measurement Unit
MEMS	Micro-Electro-Mechanical Systems

Abstract

One of the three goals of an Inertial Navigation System(INS) is to estimate the 3D orientation of INS given accelerometer, magnetometer and angular rate gyroscope readings. Complementary fusion is one of the most robust, mathematically simple and fast algorithms for fusing the accelerometer, magnetometer and angular rate gyroscope readings to estimate the 3D orientation. But complementary fusion of sensor data suffers from linear accelerations and constant gain problems. This thesis aims to solve the aforementioned issues in a data driven approach. Our thesis proposes a data driven toggling scheme for the gain parameter and also proposes a data driven noise removal approach through a tree based machine learning model. The variable gain decides trust of fusion between the sensors and the denoising ensures extraneous linear accelerations and magnetic disturbances are eleminated. Furthermore, the proposed algorithm is tested on a real-world dataset, and also we look forward to construct a device to further test our proposed algorithm on a real-world scenario.

Chapter 1

Introduction

1.1 Inertial Navigation and MARG Sensors

Inertial Navigation refers the process of determining the *position*, *velocity* and *orientation* of an object to relative to its initial navigation parameters, without the assistance of any position fixing devices located externally [37]. Any system estimating position, orientation or velocity of an object using Inertial Sensing is referred to as the Inertial Navigation System(INS).

The INS uses a multitude of sensors to predict the position, orientation or the velocity. Two of these sensors are common to almost all of these systems, which are the Gravity Sensor(senses the direction of gravitational acceleration vector) and the Gyrosope(Angular Rate Sensor). Optionally some systems include the Magnetometer sensor(senses the direction of the Magnetic North)[33].Together they are referred to as the **MARG** sensors.

The INS faces significant amount of challenges while solving its 3 problems - **position estimation**,**veloicty estimation** and **orientation estimation**. Low cost MARG sensors are subject to electrical disturbances and noises [19][21].The angular-rate sensor is subject to random noises and drift(a phenomenon where low frequency noises integrate over time and skew the estimates)[19]. Accelerometers suffer from linear-accelerations, where the linear acceleration becomes harder to distinguish from the gravitational vector[21].

1.2 Orientation Estimation and MARG Sensors

Orientation Estimation is one of the 3 problems that an INS has to solve. Orientation of a 3d object could be represented by Euler Angles, Rotation Matrices, or quaternions[39]. An orientation estimation algorithm is fed with the MARG sensor readings and it estimates the orientation of an object in 3D space, representing it in Euler angles, quaternions or through a rotation matrix.

Any orientation estimation algorithm has to overcome the shortcomings of MARG sensors mentioned in 1.1. Some popular algorithms are the Madgwick filter[28], Mahony filter[29], EKF filter[30], Complementary filter[20]. Each algorithm employs distinct methodologies to mitigate the effects of sensor disturbances and enhance orientation estimation accuracy.

Our focus in this thesis would be to work with orientation estimation through the utilization of MARG sensor readings.

1.3 Orientation Estimation through Complementary Fusion

The complementary filter is one of the most basic form of fusion where the orientation estimates of the Angular Rate-Gyros are fused with the estimates generated from the accelerometer and if available the magnetometer. The basic complementary filtering[17][15] is represented as:

$$R = (1 - \alpha)R_g + \alpha R_{am} \tag{1.1}$$

Where R is the rotation estimation, R_g is the rotation estimation from the rate-gyros and R_{am} is the rotation estimate from the accelerometer and the magnetometer.R being the rotation estimate can be represented in multiple forms such as Euler angles, quaternion or as a rotation matrix.

1.3.1 Key Strengths of CF

- Reduced Calculation and computation, simple mathematical model.
- Faster convergence compared to Madgwick and EKF Filters[41].

1.3.2 Weaknesses of CF

- The value of the gain α is constant.
- Cannot efficiently differentiate between linear acceleration and gravitational acceleration.

1.4 Our Goal- Overcoming the negatives of CF: A data driven approach

As discussed in section 1.3, complementary filters (CF) offer notable advantages over existing orientation estimation algorithms. The objective of this thesis is to address the limitations of CF by utilizing data-driven approaches. Specifically, our goals are to:

- Develop an ML model to dynamically adjust the parameter α based on MARG sensor measurements, enabling differentiation between low and high linear acceleration scenarios.Such methods have been studied in previous researches in [42].
- Train a predictive model to estimate and compensate for noise in MARG sensor data. Similar approaches found as [9] and [14].

Through these improvements, we aim to improve the performance of CF, mitigating its weaknesses and placing it as a robust and competitive algorithm for orientation estimation.

1.5 Organization

After this introduction , we elaborate on some of the related works that have been done on this field in chapter 2. Then we describe our proposed methodology in chapter 3. Finally, we show some of the results we have achieved so far in chapter 5. The results which were acheived in chapter 5 are further elaborated and broken down in chapter 6. Lastly, chapter 8 discusses what we look forward to do in testing our proposed algorithm on real world physical scenarios instead of working on error metrics over a synthetic dataset.

Chapter 2

Related Works

From Virtual Reality domain and smartphone-based pedestrian navigation to autonomous aerial vehicles and human motion analysis, a wide range of applications across multiple domains rely on orientation estimation utilizing the **IMUs (Inertial Measurement Units)** incorporated in the devices. IMUs provide interpolated data collected from three sensors **(accelerometer, magnetometer and gyroscope)** and provide vital information regarding the angular rate, attitude/orientation or specific force for the body. The inherent drawbacks of MEMs IMU sensors, such as noise, bias, and drift over time, make orientation estimation difficult and need the application of complex algorithms to fuse sensor data and generate precise and trustworthy orientation estimations. It directly influences subsequent computations like velocity and position estimation through *dead reckoning*. In dead reckoning, errors in orientation estimation can accumulate over time, leading to significant deviations in position tracking. Therefore it is all the more essential to enhance the accuracy and reliability for benchmarking the overall performance of the navigation and tracking system.

2.1 Approaches and algorithms

Several literature have addressed these problems in different ways i.e in *mathematical/deterministic* ways or through *data-driven ways using machine learning or deep learning*, or a *hybrid* of them. For example, Vertzberger and Klein note that smartphone IMUs incorporate high-amplitude noise and time-varying bias, and that achieving accurate estimation requires the involvement of all nine IMU signals (all three axes of the three inertial sensors) with time-varying weights. [42]. In their work they did not incorporate the magnetometer results. In practice, the filter must adjust to changing dynamics and varying noise, which fixed fusion weights often fail to address.

Below are the approaches that have been used to address the challenges imposed by orientation estimation on the basis of a timeline:

2.1.1 Mathematical approaches

Classical Filters (Complementary filters [20], Kalman Filters[30], Gradient Descent[28]). A broad class of AHRS solutions use deterministic algorithms such as **complementary filter[20]**, **extended Kalman Filters (EKF)[30]**, **unscented Kalman Filters (UKF)[45]** and **gradient descent [28]** methods. These exploit known kinematic gyroscope readings and external reference vectors from accelerometer (gravity) and magnetometer (magnetics) in closed form. For instance, the QUEST algorithm[38][5] and Kalman filters solve Wahba's problem (erroneous rotation matrix calculation)[44] optimally, but assume stationary noise models. As stated in [42], Kalman filters are largely dependent on the tuning of the process and measurement covariance, and incur heavy computational cost due to calculations referring to inverse matrices. Conversely, complementary filters follow a linear interpolative approach and fuse gyro predictions with accelerometer/magnetometer corrections in the frequency domain. CFS (Madgwick [28], Mahony [29] have intuitive fixed gains (alphas) and comparatively low computation, yet they suffer from slow convergence under rapid motions or shocks. Madgwick's filter [28] uses a gradient-descent step calculation based on accelerometer and magnetometer Jacobians, but requires manual tuning of the step gain (alpha). To address these problems regarding dynamic motions, variable-gain CFs have been proposed. Ding Duong Quoc et al. demonstrated the performance of this adaptive-gain CF [13] which dynamically changed the filter constant) and observed it to perform significantly better than a fixed-gain CF in roll/pitch accuracy under varying motion. Likewise, Shao et al. design a variable-gain complementary filtering method for aircraft angle-of-attack estimation, proving in simulation that continuously adaptive filters yield far more stable estimates than conventional CFs. It dynamically adjusts the alpha gain based on inertial network variance patterns, reducing the error by half compared to the fixed-gain designs in aerodynamic sensing applications. [35]. Similarly, Broughton et al. (2019) develop a robust complementary filter for UAVs that monitors the times when accelerometer readings are reliable. Based on the 'steadiness' model, it decides when to incorporate gravity measurements and constrains gyro biases using a Gaussian random-walk model. In Monte-Carlo tests of aggressive maneuvers, this filter has been able to track roll more accurately than standard CFs. [1]

Kalman filters make use of probabilistic models to recursively predict and update orientation estimates [30]. Sabatini's quaternion-based EKF incorporates sensor biases and mitigates motion and noise disturbances by adapting measurement noise covariances, which addresses the challenge of adaptability in human movement analysis [34]. The Unscented Kalman Filter avoids linearization by propagating sigma points, offering better performance when noises are Gaussian. [45]. However, these Kalman-based approaches require careful covariance tuning and are complex because of inverse matrices. In practice, researchers often trade theoretical optimality for robustness by using simpler linear filter (CF or gradient-descent) or by applying dual-filter schemes as demonstrated in [12]. In this work, a two-step EKF is performed on a 9 DOF device (acc, mag, gyro) which models disturbances explicitly and results in $<1^\circ$ static error, stabilizing performance against magnetic disturbances. But despite high accuracy, EKF incur heavy computational performance overhead. But still, [42] surveys that the algorithm in [38] used for EKF [30] is probably the most accepted approach despite its sensitivy to noise assumptions.

Madgwick's gradient descent filter formulated orientation estimation as an optimization using Jacobians to minimize the error between predicted and measured sensor vectors [28]. It takes only 227 operations per update, which is significantly lighter than EKF implementations and enables high sampling rates in resource-constrained platforms. While the filter converges well under quasi-static conditions, it exhibits slow convergence under rapid dynamic motions because of the erroneous accelerometer bias from the fixed gain [47]. As opposed to the variable gain solutions previously discussed, an enhanced gradient descent algorithm was proposed by Wilson et. al (2019) which decoupled the magnetic variance from the calculation and improved the robustness for the convergence [47]. In order to mitigate magnetic disturbances and accelerate convergence, several enhancements have been proposed. Madgwick's extended complementary filter for full body MARG orientation (2020) dynamically adjusts gain parameters to decouple magnetometer influence, achieving up to 33% heading error reduction under fast-changing environments while maintaining computational efficiency [27].

Zhang et al. (2020) propose an **adaptive sparse-interpolation CF (ASICF)** for MEMS IMUs. They use a quaternion-based complementary filter for fusing gyro and accel, but samples asses the 'trustworthiness' of the accel measurements (via its variation) from the successive samples, and if the data are too noisy or dynamic, it performs an interpolation step rather than using the raw sample [49]. Basically, the algorithm performs an *adaptive data-skip mechanism* and then fills the gaps produced from the dropped samples using interpolation. They evaluated ASICF on multiple datasets and report significantly lower attitude errors under large disturbances (20% smaller error than the Valenti CF [49]. Its performance depends directly on being able to detect outliers, and upon detection, it can handle sudden motions by effectively smoothing or skipping bad accel readings.

Guo et al. (2023) introduce a **variable gain CF** for aircraft angle-of-attack estimation using a distributed IMU network plus flush-air sensors [35]. They derive a flight-phase-dependent blending factor: the filter coefficient (analogous to alpha) is allowed to vary with the change rate of the angle-of-attack, placing more weight on inertial data at high dynamics. Simulation results for a high-speed UAV show this VGCF yields significantly smaller AoA error than constant gains (approximately 0.0058° vs. 0.0017° RMSE i.e > 2x improvement). However, it is quite domain-specific even though it exemplifies adaptive gain analogous to our approach.

In summary, classical filters are computationally efficient and theoretically grounded, yet they often require hand tuned gains or multiple stages to handle non-ideal IMU behavior and mitigate erroneous values.

2.1.2 Machine-Learning Enhanced Filters

The overhead for manual tuning is reduced by introducing data-driven machine learning approaches in orientation estimation. One approach is to make the filter gain *adaptive* using sensor data. Our approach is strongly analogous to the approach followed by Vertzberger and Klein (2022). They introduced a **hybrid adaptive complementary filter** that learns axis-specific accelerometer weights via neural networks [42]. They integrate gyro data classically (quaternion integration) and then apply a complementary update in each axis whose weight is predicted by a small neural network based on estimated linear accelerations. The method was evaluated on a smartphone IMU (60 two-minute sequences of walking activities in pocket, hand etc. with VI-SLAM Ground truth [42]) and compared against fixed gain filters like Madgwick, Mahony, AEKF etc. The learned filter yielded the lowest roll/pitch errors (10-37% better than classic filters on average). It adapted to dynamic motion via data-driven weight tuning, which outperformed its corresponding fixed gain filters. However, it requires offline labeled training, added complexity for neural net and most importantly, the yaw is not addressed (no magnetometer fusion). In a similar spirit, Maton et al. tune a CF gain indirectly; they embed filter gains in Mamdani fuzzy inference system (FIS) and use a genetic algorithm to optimize the membership functions so as to minimize velocity error against ground truth (obtained via MOtion CAPture devices). In an experimental robot dataset, GA-tuned gains reduce velocity and position errors.[31] While it doesn't specifically require explicit vision-based orientation, the tuning, again, is done offline and needs to be redone for new scenarios. Another line of work trains small neural models to train corrections; minimize noise. For example, Al-Sharman et al. (2019) train a neural model to enhance filter outputs by learning correction terms [36]. Brossard et al. introduce deep CNN not end-to-end, but to *denoise* gyroscope signals before integrating. Evaluated on EuRoC and TUM-VI datasets, the learned filter *outperforms state-of-the-art methods* and even beats top visual-inertial odometry algorithms in attitude accuracy. [9]. But it requires large ground-truth datasets for training and the open-loop mode still drifts (converge slower) due to the complex design. Weber et al. 's RIANN employs a neural network which was trained on diverse motion datasets for performing real-time attitude estimation from 6-DoF IMU data without requiring brute force tuning. It demonstrated generalization across diverse environments and sampling rates, but domain-specific optimization is required for peak performance. [46]. Hybrid approaches, although sets a promising direction for data-based fusion algorithms, might impose challenges upon encountering new motions in the environment i.e it comes with the risk of overfitting.

2.1.3 Deep Learning Enhanced Filters

Beyond hybrids, fully deep end-to-end methods have been explored. These treat orientation estimation as a regression or sequence problem using powerful neural architectures. Convolutional networks (CNN) and recurrent networks (LSTM, GRU, transformers) have been trained to map IMU time-series to orientation quaternions. Golroudbari and Sabour proposed an end-to-end CNN + bi-directional LSTM that inputs raw 6-DOF (mag exclusive) IMU measurements and directly regresses quaternion angles [3]. It was evaluated on 7 different datasets (120+ hours of motion including EuRoC, TUM-VI) and reduced orientation root mean square errors by 20-50% vs. Madgwick and EKF baselines without per-dataset fine-tuning. Their network uses 1D convolution later for extracting local motion features over 0.5s windows, followed by bi-LSTM layers for modeling temporal context and finally aligning with the quaternion normalization layer. [3]. Tedaldi et al. (2014) fused gyroscope and accelerometer streams using multi-head attention. Demonstrating faster convergence (<0.2s) and 25% lower drift over 1 minute trajectories compared to Madgwick. [40] Brossard et al. (2020, NDAG 27) train dilated convolution networks to denoise gyroscope signals and perform open-loop dead reckoning. On EuRoC and TUM-VI benchmarks, their method surpasses top visual-inertial odometry systems in attitude estimation accuracy without any visual input. These attention maps reveal that during high dynamic or rapid motion settings, the model attends more to gyroscope results, paralleling the alpha toggling concept. [9] In [43], they used PCA and trained a stacked denoising autoencoder on synthetic IMU noise to correct accelerometer and gyroscope biases online. Over 120 seconds, it resulted in a 40% reduction in Allan-variance-derived (statistical analysis for deriving nature of a drift, source) drift. This [36] employs a 1D U-net architecture to remove magnetic disturbances from MARG data, lowering heading error by 30% in indoor environments only with dynamic ferromagnetic interference as noise. These architectures can be refurbished to serve the purpose of noise prediction models. The common limitation of these deep-learning based approaches is the requirement for large ground-truth datasets and computational resources, as well as the "black-box" nature of these models. Nonetheless they demonstrate that fully learned orientation estimators can be highly accurate and adapt to unknown motion contexts, although these are quite heavy for low-cost sensors.

2.2 Gaps and Opportunities

Across these studies, most *adaptive* methods still rely on either **fixed fusion structures** with heuristic tuning or offline learning. For example, fixed-gain complementary filters (Mahoni [29], Madgwick [28] are cheap but cannot adjust to rapid dynamic motions, whereas hybrid approaches like [42] or [49] adapt to motion but require pre-training or brute force tuning. Pure learning methods in [9], [3] achieve excellent accuracy but need vast training data and loser interpretability. The genetic or fuzzy tuning of [31] shows that optimizing gain helps, but remains static once deployed. Notably, **none** of the reviewed works use machine learning to **predict sensor noise *online*** or to dynamically **toggle the complementary filter co-efficient in *real time***. This constitutes a gap; a **a model based filter** whose parameters (measurement noise or blending factorα) adapt instantaneously to the current operating conditions based on learned patterns. Our XGBoost approach might be able to fill this gap by learning to estimate noise levels (and thus adjust α) on the fly, combining the strengths of data-driven adaptivity with the structure of a complementary filter. This addresses dynamic disturbances in a principled way that is not covered by the existing literature [42] [9].

2.3 Cohesion with Approach

We are following two main approaches for conducting the orientation estimation. The first one involves an XGBoost trained Toggle Engine that sets α-gain=0 when it detects motion letting the gyro integration alone drive the rapid changes, and α-gain

> 0 during rest to fuse accelerometer and magnetometer values for drift correction. In coherence with this, Meyer et al. [1] detect dynamic vs. steady states via gyro-bias and mag-rate thresholds, improving heading error by 33% under fast maneuvers. Likewise, Shao et al. [35] dynamically adjusted α based on inertial network variance, halving angle-of-attack error compared to its fixed gain substitutes. Most importantly, the literature that corresponds closest to our approach, [42] by Vertzberger and Klein embed a neural net to predict axis-specific gains, outperforming static α-filters across human motions from pedestrian data. [40] shows multi-head attention naturally attends to gyro during high dynamics, which is analogous to setting α->0 when motion is detected by our Toggle Engine. Similar approaches have been followed in [18], [47], [3] but all of them one way or other fine-tuned the parameters required offline as a part of pre-processing, whereas we are following a approach that predicts the motion and dynamically adjusts the α gain online. Again, the hybrid or ML/DL-based models rely on multilayer perceptron, but we use XGBoost. The tree-based model is far lighter and feasible for embedding in a micro-controller, whereas the neural networks require more memory and FLOPs. For our second approach, we are essentially denoising using XGBoost. We are trying to predict the noise (the quaternion error between the complementary filter output and the ground truth) and at runtime, we subtract this derived noise from the CF-derived vectors before the next fusion step. In coherence with this, [49] predicts accelerometer noise via XGBoost, cutting Allan variance by 40% vs. ARMA. [27] suffered from slow convergence as stated in [28], but the fixed β (read α), cannot adapt to noise fluctuations, Our XGBoost noise regressor can suggest β adjustments via equivalent covariance tuning. Again, survey papers related to [30] state that static covariance assumptions limit filter robustness; which motivates our use of XGBoost for predicting time-variance noise statistics for EKF covariance updates. In [49], they train a 1-D U-net to remove magnetometer disturbances in indoor MARG, which is essentially equivalent to an XGBoost regressor per axis for disturbance correction, feeding EKF or Complementary signals with cleaner signals. So rather than static or CNN-based denoising, our XGBoost regressor directly target the CF quaternion-error preserving low compute and interpretability while achieving deep-net level noise reduction. Neural networks (CNN, LSTM, BiLSTM) being used in [3] and others while providing high accuracy, are black-boxes with millions of parameters, whereas we can ensure interpretability by incorporating a simple if-else based tree and the outputs can be linked to the inputs, which is why it easier for our model to run on low power embedded models in real time.

Chapter 3

Proposed Methodology

In this chapter, we propose a data-driven methodology designed to overcome the limitations of the Complementary Filter (CF). Thus the following section will correspond to chronological steps executed to improve the α prediction and noise elimination.

3.1 Research Design

We will go through four steps, where we introduce our mathematical background for the filter design and model training. The steps are listed below:

- Modified Complementary Filter Design based on the works of [42].
- Training Dataset
- Training the ML model for movement prediction
- Training the ML model for noise prediction

These steps are elaborated in later sections.

3.2 Choice of the Machine Learning Model

For our research, it is important that we run the model in every iteration to remove noises from the sensor readings and prediction of movement. Thus we need a model which will have very little inference time. Thus a tree based model would best suit our needs.

Also it is imperative that we have improved bias-variance tradeoff and our model is robust to outliers. Tree based models are prone to overfitting[8]. Thus we would need a robust learning method to minimize overfitting of the XGBoost trees. Thus RANSAC(Random Sample Consensus)[16] algorithm would be our preffered choice for this thesis.

3.2.1 Choice of Extreme Gradient Boosting Regression and Classification Models

XGBoost (Extreme Gradient Boosting) has become a preferred choice for both regression and classification tasks due to its robust performance, scalability, and advanced regularization techniques. The key reasons include:

- **High Predictive Accuracy:** XGBoost consistently achieves impressive performance on structured/tabular datasets in both academic benchmarks and real-world competitions due to its ability to minimize bias and variance simultaneously [11]. Such accuracy is required for our experimentation for accurate movement prediction.
- **Regularization Capabilities:** Unlike traditional gradient boosting, XGBoost includes L1 and L2 regularization terms, which help prevent overfitting—a common issue in high-dimensional data [11].
- **Handling of Missing Values:** XGBoost can handle missing data internally during training without requiring preprocessing imputation, improving ease of use and robustness [11].
- **Efficient Computation:** With its use of histogram-based algorithms and parallel processing, XGBoost is highly efficient and scalable to large datasets [23]. BROAD dataset with almost 2,400,000 records combined, needs a highly efficient and paralellizable model training process.
- **Flexibility:** It supports a variety of objective functions and custom loss functions, making it suitable for a wide range of regression and classification problems [11].

3.2.2 Choice of Random Sample Consensus(RANSAC) for Regression

Random Sample Consensus (RANSAC) is an iterative algorithm designed to estimate the parameters of a mathematical model from a dataset that may contain outliers. RANSAC operates by selecting random subsets of the data to fit a model and then

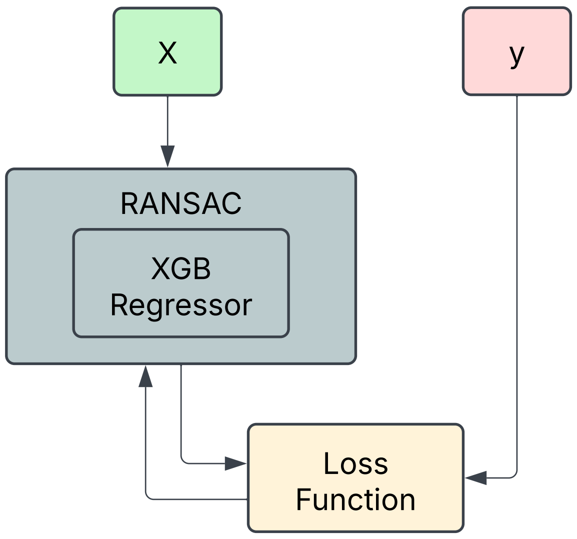


Figure 3.1: Regression Model Training Pipeline with RANSAC

determining the number of inliers—data points that fit the model within a predefined tolerance. The model with the highest number of inliers is considered the best fit. This approach is particularly effective in scenarios where the dataset is contaminated with outliers, as it focuses on the consensus of the inlier data points while disregarding the outliers [16].

XGBoost can be sensitive to outliers in the training data, which may lead to overfitting and reduced generalization performance. Integrating RANSAC with an XGBoost regressor combines the strengths of both methodologies to enhance model robustness and predictive performance. The regression training pipeline with RANSAC is visualized in Figure 3.1:

- **Robustness to Outliers**: RANSAC's ability to identify and exclude outliers ensures that the XGBoost model is trained on cleaner data, leading to more reliable predictions.
- **Enhanced Generalization**: By focusing on inliers during the training process, the combined model is less likely to overfit to noise and anomalies, improving its performance on unseen data.
- **Complementary Strengths**: While XGBoost excels at capturing complex, nonlinear relationships in data, RANSAC provides a mechanism to mitigate the influence of outliers, resulting in a more robust and accurate model.

This hybrid approach has been explored in various studies, demonstrating its effectiveness in improving model performance in the presence of outliers [26].

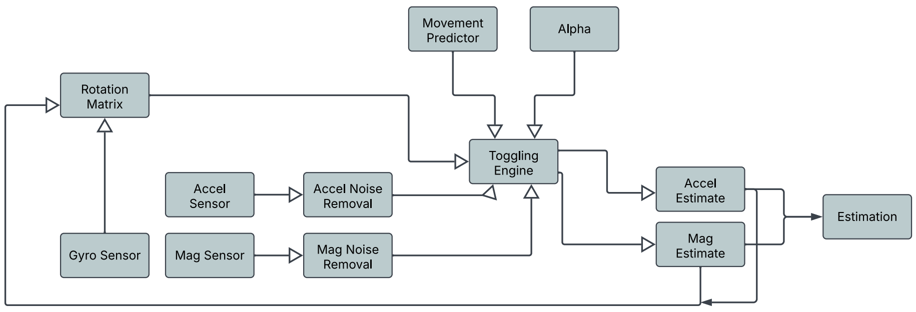


Figure 3.2: Filter Diagram

3.3 Modified Complementary Filter Design

The complementary filter algorithm we incorporated in our thesis is described in this section. The CF filter featured here incorporates a gain parameter α which would be generated by a *Toggling Engine* mentioned in section 3.5. The algorithm also supports noise removal if a trained model is provided [9].The algorithm is a built on the works of [42] where we have added the magnetometer readings, modified the α prediction and introduced noise removal. Some information about the algorithm:

- The algorithm uses the a_{pred} and m_{pred} (predicted gravity and north vectors respectively) to describe the orientation. This representation can be interchanged to quaternion or a rotation matrix[7][10].
- The sensor readings are formatted as $(3, 1)$ column vectors:
  - Gravitational Vector from the accelerometer: $a_i = (a_{ix}, a_{iy}, a_{iz})$
  - Magnetic North Vector from the magnetometer: $m_i = (m_{ix}, m_{iy}, m_{iz})$
  - Gyro angular rates on 3 axis: $g_i = (g_{ix}, g_{iy}, g_{iz})$
- The rotation matrix generated in the iteration i from the angular rates [48] is:

$$R_g = e^{(\Omega_\times(g_i)\delta t)} \tag{3.1}$$

  Where $\Omega_\times$ is the skew-symmetric operator and δt is the sampling rate.

The algorithm is described in algorithm 1. The algorithm is represented through the flow diagram in Figure 3.2

3.4 Dataset Description

This section describes the dataset we are currently utilizing to train our models. The dataset we are using is the Berlin Robust Orientation Estimation Assessment Dataset(BROAD)[25].

The BROAD dataset contains 39 recordings or trial. Each of the trial contains a time series of MARG sensor data readings along with the boolean parameter `movement` and the position(`opt_pos`) and the orientation(`opt_quat`) information of each timestamp gathered from the Optitrack OMC system at 120Hz.

The MARG sensor readings come from a custom 3D printed device on which an Inertial Measurement Unit(IMU) is mounted. The IMU was a commercially available nine-axis inertial sensor (myon aktos-t, myon AG) recording 9 sensor reading at a sampling rate of 286Hz. The hardware device is shown in Figure 3.3(b). The trials in the dataset are generated through the motion of the 3d printed device through different pre-defined motion parameters which is elaborated in the paper for BROAD dataset. A breif categorization of the types of recording is shown in Table 3.1. This categorization is based on the type of physical motion the 3D printed hardware goes through while undergoing a pre-defined motion trajectory while recording the trial.

Table 3.1: Categorization of BROAD Dataset Trials

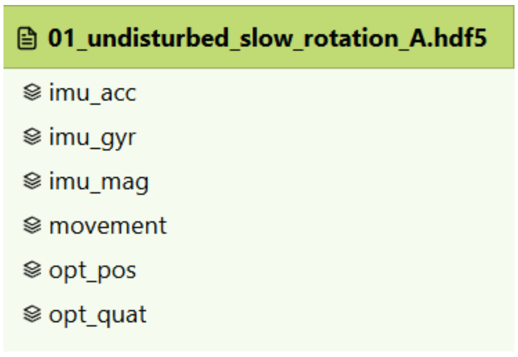
Motion Type	**Speed**	**Trial/recording Indexes**
Undisturbed		
Rotation	Slow	01, 02, 03, 04, 05
Rotation	Fast	06, 07, 08, 09
Translation	Slow	10, 11, 12, 13, 14
Translation	Fast	15, 16, 17, 18
Combined	Slow	19, 20
Combined	Fast	21, 22, 23
Disturbed (Medium Speed)		
Tapping	–	24, 25
Vibrating Smartphone	–	26, 27
Stationary Magnet	–	28, 29, 30, 31
Attached Magnet (1–5)	–	32, 33, 34, 35, 36
Office Environment	–	37, 38
Mixed (Disturbed and Undisturbed)	–	39

3.5 Gain(α) Estimation through Movement Prediction

Here we describe the working principle of the *Toggling Engine*. The α parameter is initially set to be a constant value. The toggling engine is responsible for toggling the alpha in presence of movement. Detailed steps:

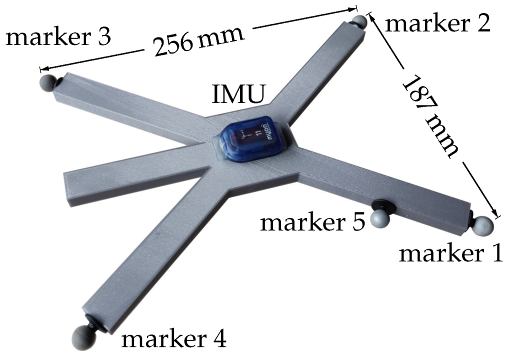
Algorithm 1: Modified CF algorithm

```
Input: MARG sensor readings = $\{(g_1, a_1, m_1), (g_2, a_2, m_2), ....\}$ , Toggling Engine ,
       Noise Model , Alpha: $\alpha$, Duty Cycle Params: $d_l$ and $d_h$
Output: Predicted gravity and north vectors $a_{pred}$ and $m_{pred}$

(1)  Calibration sample count: $N = 500$
(2)  Gyro bias: $g_{bias} = \sum_{i=0}^{N} g_i$
(3)  Initial Gravity Vector Prediction: $a_{pred} = \sum_{i=0}^{N} a_i$
(4)  Initial North Vector Prediction: $m_{pred} = \sum_{i=0}^{N} m_i$

(5)  foreach $(g_i, a_i, m_i)$ in MARG sensor Readings, $i > N$
(6)  do
(7)      Remove gyro bias and normalize accel-mag readings:
             $g_i = g_i - g_{bias}$
             $a_i = normalize(a_i)$
             $m_i = normalize(m_i)$
(8)      Form $R_g$ from $g_i$.
(9)      Form the predictions from the gyro sensor:
             $a_{gyr} = R_g \cdot a_{pred}$
             $m_{gyr} = R_g \cdot m_{pred}$
(10)     Predict noise from noise model:
             $(a_{noise}, m_{noise}) = NoiseModel(g_i, a_i, m_i)$
(11)     Predict alpha from the Toggling Engine:
             $\alpha = TogglingEngine(i, g_i, a_i, m_i)$
(12)     Compute the dynamic scaling values:
             $e_{ga} = ||\mathbf{a} - \mathbf{a}_g||, \quad e_{gm} = ||\mathbf{m} - \mathbf{m}_g||, \quad e_{na} = ||\mathbf{a}_{noise}||, \quad e_{nm} = ||\mathbf{m}_{noise}||$
             $\alpha_{ga} = \frac{\alpha \cdot e_{ga}}{e_{na} + e_{ga}}, \quad \alpha_{gm} = \frac{\alpha \cdot e_{gm}}{e_{nm} + e_{gm}}, \quad \alpha_{na} = \frac{e_{na}}{e_{na} + e_{ga}}, \quad \alpha_{nm} = \frac{e_{nm}}{e_{nm} + e_{gm}}$
(13)     Update the predictions:
             $a_{pred} = a_{gyr} - \alpha_{ga}(a_i - \alpha_{na} a_{noise} - a_{gyr})$
             $m_{pred} = m_{gyr} - \alpha_{gm}(m_i - \alpha_{nm} m_{noise} - m_{gyr})$
(14)     Normalize the predictions:
             $a_{pred} = normalize(a_{gyr})$
             $m_{pred} = normalize(m_{gyr})$
(15)     Append $a_{pred}$ and $m_{pred}$ to $orientationEstimates$
(16) end
(17) return $orientationEstimates$
```



01_undisturbed_slow_rotation_A.hdf5
- imu_acc
- imu_gyr
- imu_mag
- movement
- opt_pos
- opt_quat

(a) Columns in the BROAD dataset



(b) 3D printed hardware device with strapped on IMU.Optical markers placed on four corners for tracking

- Train an XGBoost Classifier model[11] on the input features X where X contains the raw sensor readings of the MARG sensors and output class label y where y contains the boolean variable movement.

- Use the trained Model in the *Toggling Engine* according to the algorithm 2.

Algorithm 2: Toggling Engine

```
Input: MARG Sensor Readings: $(g_i, a_i, m_i)$, Iteration Index $i$
Data: Alpha: $\alpha$=0.01, Duty Cycle High: $d_h$, Duty Cycle Low: $d_l$,XGB Model $\mathcal{M}$
Output: Estimated $\alpha$
Function TogglingEngine($i$,$g_i$,$a_i$,$m_i$):
    Compute total period: $T \leftarrow d_h + d_l$
    Predict movement: $\hat{\delta} \leftarrow \mathcal{M}(g_i, a_i, m_i)$
    if $\hat{\delta}$ == 0 then
        return $\alpha$
    if ($i$ mod $T$) < $d_h$ then
        $\alpha \leftarrow \alpha$
    else
        $\alpha \leftarrow 0$
    end
    return $\alpha$
```

3.6 Noise Removal

Here we describe the training procedure of the model that removes noise. The ground truth noise values are established from the difference of actual sensor measurements and the ground truth gravitational and magnetic north vectors. The ground truth quaternion q_{gt} generated from the quaternion components in `opt_quat` in the dataset is used to rotate the a_{avg} and m_avg vectors to remove the noises. The a_{avg} and m_avg vectors are calculated by averaging initial sensor readings for $N = 500$ iterations. The value of N could be varied.

- Ground truth gravity vector is calculated using the ground truth q_{gt} obtained for each record in a recording dataset `opt_quat`: More information on this rotation formula in subsection A.1.2

$$a_{gt} = q_{gt}^{-1} \cdot a_{avg} \cdot q_{gt} \tag{3.2}$$

- Ground truth magnetometer north vector is calculated using the ground truth q_{gt} obtained for each record in a recording dataset `opt_quat`:

$$m_{gt} = q_{gt}^{-1} \cdot m_{avg} \cdot q_{gt} \tag{3.3}$$

- An XGBoost Regression model[11] is to be trained with input features X being the raw MARG sensor readings and the output feature y being the six components of the a_{gt} and m_{gt} vectors. The training dataset contains may contain all the MARG sensor readings from all the recordings, each sensor reading corresponds to a ground truth quaternion q_{gt} represented as `opt_quat` in the datasets. Each of the q_{gt} is used to generate a_{gt} and m_{gt} vectors used for the regression model training. The loss function used is the generic *Mean Squared Error Loss function* [6]. The trained model is to be used to do noise prediction in algorithm 1 where the noise is subtracted from the sensor readings.

Chapter 4

Experimental Setup

4.1 Mechanical Platform (Turntable)

4.1.1 Turntable Setup

The experimental turntable was designed in Fusion360 as shown in Figure 4.1(a) and constructed using 3D-printed parts and integrated sensors to provide both actuation and ground-truth angle measurement. The constructed device is shown in Figure 4.1(b) and Figure 4.1(c). The system consists of a circular rotating disk mounted on a bearing, a box-shaped support structure, and a base platform with a magnet for the AS5600 magnetic encoder. The disk carries the electronics, including a Raspberry Pi 5, an MPU9250 MARG sensor, and a 20,000 mAh power bank, which provides standalone operation. The detailed specifications of the turntable are summarized below.

- **Turntable geometry:** Diameter [XX cm], 3D printed PLA/ABS material, mass [YY g].
- **Actuation:** There is no electrical actuation built into the turntable, rather movement is executed by manually rotating the turntable.
- **Mechanical construction:**
  - Upper circular disk connected to the inner ring of the 6010 bearing.
  - Box-shaped 3D printed housing supporting the bearing.
  - Platform with embedded magnet for AS5600 encoder.
- **Ground-truth sensing: Ground-truth sensing:** AS5600 absolute magnetic encoder with the following characteristics:

  - Resolution: 12-bit (4096 positions per revolution, $\approx 0.087°$ per step).
  - Angular accuracy: typically $\pm 0.5°$ to $\pm 1°$, depending on magnet alignment and air gap.
  - Maximum angular deviation: up to $\pm 1.4°$ worst-case under misalignment or offset conditions.
  - Latency: on the order of 1–2 ms (limited by I^2C polling rate or PWM read-out frequency).
  - Output interfaces: I^2C, PWM, or analog voltage.

- **On-disk payload:** 20,000 mAh powerbank, Raspberry Pi 5 (8 GB), MPU9250 MARG sensor, AS5600 encoder.

4.1.2 Electronics & Sensors

The inertial sensing subsystem is based on the InvenSense/TDK MPU-9250, which integrates a tri-axial accelerometer, gyroscope, and the AK8963 tri-axial magnetometer. Each sensing axis employs 16-bit ADCs, ensuring adequate dynamic range for motion tracking. The IMU was mounted on a 3D-printed cube positioned at the edge of the turntable disk. Directly beneath the ring connecting the disk to the bearing, the AS5600 magnetic angle encoder was fixed, enabling precise alignment between inertial measurements and ground-truth angular position.

- **IMU:** InvenSense/TDK MPU-9250, tri-axial accelerometer and gyroscope with integrated AK8963 magnetometer, 16-bit ADCs.
- **Gyroscope ranges:** ± 250, ± 500, ± 1000, and ± 2000 dps (default range used).
- **Accelerometer ranges:** ± 2, ± 4, ± 8, and ± 16 g (default range used).
- **Sample-rate and DLPF settings:** Default sample rate and default digital low-pass filter configuration were used.
- **Processor/DAQ:** Raspberry Pi 5 (8 GB) running the default Raspberry Pi OS. Communication with both the MPU-9250 and the AS5600 magnetic encoder was established over the I^2C bus. Supply voltage and logic level were 3.3 V.
- **Power:** A 20,000 mAh USB power bank mounted on the disk provided power to the Raspberry Pi, IMU, and encoder.

4.1.3 Motion Protocols

The turntable experiments were conducted under three categories of motion: static, quasi-static, and dynamic. The coordinate frames tested were determined by the feasible orientations of the setup: rotation was applied around the z-axis and y-axis with the positive direction pointing upward, and around the x-axis with the positive direction pointing downward. Rotation around the x-axis upward was not performed due to mechanical constraints imposed by wiring. The three types of motions are described below:

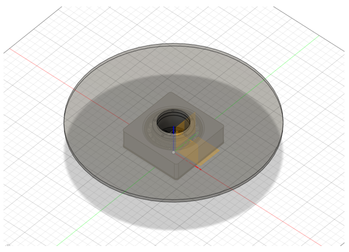
- **Static:** The system experiences no movement. This condition is used to establish baseline sensor stability and drift characteristics.
- **Quasi-static:** Slow and controlled movements are applied, typically at speeds low enough that inertial effects are negligible. This allows the evaluation of sensor accuracy under gradual angular changes.
- **Dynamic:** Continuous and rapid motions are applied, where inertial effects are significant. This tests the system response under sustained rotation and higher angular velocities.

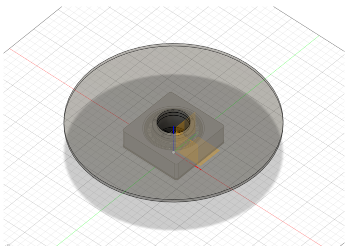
4.1.4 Software Setup and Result Generation

The software infrastructure for the experimental turntable was implemented on a Raspberry Pi 5 (8 GB). The system executes four independent Python scripts corresponding to different orientation estimation algorithms: the proposed filter, the AQUA filter, the Extended Kalman Filter (EKF), and the Madgwick filter. These scripts are coordinated by a parent `run` script, which launches each of the four filter scripts in separate threads to allow concurrent execution.

Each filter script reads sensor measurements from the MPU-9250 MARG sensor via the I^2C protocol, capturing tri-axial accelerometer, gyroscope, and magnetometer data. In parallel, a dedicated script reads angle measurements from the AS5600 magnetic encoder and converts the encoder readings into a ground-truth quaternion representation of the turntable orientation.

During runtime, the estimated quaternions from the four filters, along with the ground-truth quaternion, are continuously recorded into CSV files. Post-processing involves computing the quaternion absolute distance metric as in section A.2, between the ground-truth quaternion and each filter's estimate. These distance metrics are then plotted to visualize and compare the performance of the different orientation filters under various motion conditions. The summarized version of the results are shown in section 5.4. The graphs of the results can be found in the appendix.


(a) Fusion360 model of the turntable


(b) Actual Constructed Turntable

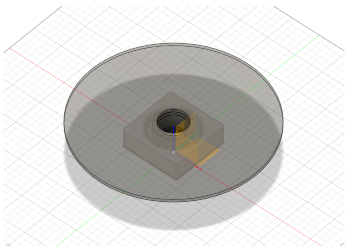

(c) With the disk Removed

Figure 4.1: Overview of the turntable setup with key components highlighted.

Table 4.1: Summary of motion protocols applied to the turntable

Motion Type	**Description**
Static	No movement; turntable held stationary.
Quasi-static	Rotation from 0° to 180° clockwise, then 0° to 180° counterclockwise; break; repeated once (total of two cycles).
Dynamic	Continuous full rotations: 3 turns, break, 4 turns, break, 5 turns.

4.2 Datasets

The dataset was already presented in section 3.4. Based on this the experimental pipeline was generated. The experiment is carried out in 3 parts, in the initial two parts the models for the movement and noise prediction are trained on the BROAD[25] dataset. Then the trained model is used to implement the Modified CF as of algorithm 1 and the results are reported as absolute distance between the Ground truth and the predicted rotation quaternions[22].

4.2.1 Training to Predict Movement

In this part we discuss the training process of the XGBoost Classifier model to predict movement. The detailed steps:

- BROAD dataset has 39 recordings. All the 39 recordings are concatenated one after another and a concatenated dataset containing all records from 39 recordings are formed. The setup for input features `X` and the output feature `y` goes as: After forming X and y the datasets were split to train and test sets with train set having 20% of the split.

Table 4.2: Dataset Generation to Predict Movement

X	y
All records/samples of MARG sensor readings from the concatenated dataset	Corresponding movement column of the concatenated dataset

- After the formation of `X` and `y`, we train an XGBoost Classifier model optimized through the `Oputna Study` library[2].The optuna study was constructed to maximize the `accuracy_score`[24] among the `y_test` and the predicted `y_pred`. The suggested parameters for the *study* were chosen as in Table 4.3:

Table 4.3: Choice of Ranges of Hyperparams for *Optuna Study(For Movement)*

Hyperparameter	**Description**	**Range**
`colsample_bytree`	Fraction of features to sample per tree.	[0.0, 1.0]
`learning_rate`	Step size shrinkage to prevent overfitting.	[0.0, 1.0]
`max_depth`	Maximum depth of a tree.	[2, 15]
`n_estimators`	Number of boosting rounds.	[1, 15]
`reg_alpha`	L1 regularization term on weights.	[0, 10]
`reg_lambda`	L2 regularization term on weights.	[0, 10]
`subsample`	Fraction of samples to use per tree.	[0.6, 1.0]

- After running the Optuna Study for `N=100` iterations, the best hyperparameter was chosen and the model was trained choosing the best hyperparams.

4.2.2 Training to Predict Noise

This subsection focuses on the training of model to predict the noises among the reported sensor readings. Detailed steps:

- There are 39 recordings in the dataset of BROAD. We choose the recordings `concat_list=[0,1,2,3,4,5,6,7,9,10,11,12,13,15,16,18,19,20,21,23, 24,25,26,27,28,29,31,32,33,34,35,36,37,38]`.

- For each recording in `concat_list` we follow the procedure mentioned in section 3.6. We average the first *N* entries then generate `a_noise` according to Equation 3.2 and `m_noise` according to Equation 3.3 for each record `i`. Then generate a new .csv dataset for each recording whose columns contain 9 MARG sensor raw values and 6 noise values (3 component from `a_noise` and 3 component from `m_noise`. Doing it for each recording in `concat_list` generates 34 new csv datasets. We concatenate all of them to form a large dataset.

- Generate `X` and `y` from the large dataset according to Table 4.4

Table 4.4: Dataset Generation to Predict Noise

X	y
All records/samples of MARG sensor readings from the concatenated dataset	Corresponding 6 values of a_noise and m_noise

- Choice of model was **XGB regressor** but coupled with the **Random Sample Consensus(RANSAC)**[16] algorithm. The ransac algorithm made training the base XGB regressor more robust to outliers and it improved the bias variance tradeoff.
- Split the dataset into train and test sets. Generate an optuna study[2] where the model is trained within the study objective on the train set and the objective works to maximize the r2_score on the test set. The choice of hyperparameters is the same as in Table 4.5. The study runs for 200 iterations.

Table 4.5: Hyperparameter Search Space for XGBoost Regressor with RANSAC

Hyperparameter	Type	Range	Description
`colsample_bytree`	Float	[0.0, 1.0]	Subsample ratio of columns for each tree
`learning_rate`	Float	[0.0, 1.0]	Step size shrinkage used in update to prevent overfitting
`max_depth`	Integer	[2, 15]	Maximum depth of a tree
`n_estimators`	Integer	[1, 25]	Number of boosting rounds
`reg_alpha`	Float	[0.0, 10.0]	L1 regularization term on weights
`reg_lambda`	Float	[0.0, 10.0]	L2 regularization term on weights
`subsample`	Float	[0.6, 1.0]	Subsample ratio of the training instances
`min_samples`	Float	[0.1, 0.9]	Minimum number of samples for a model to be considered valid in RANSAC
`residual_threshold`	Float	[1e-3, 10.0]	Maximum residual for a data point to be classified as an inlier in RANSAC
`max_trials`	Integer	[100, 500]	Maximum number of iterations for RANSAC algorithm

- The final model is trained with the best hyper parameter returned from the `study` object.

4.2.3 Implementation and Testing of the Modified Complementary Filter

The complementary filter with all its modifications proposed in algorithm 1 were implemented using `python`. Separate classes were constructed for the *Toggling Engine* and *Complementary Filter* implementation. Code of the actual implementaion could be found here.

After implementation, the filter was fed with sensor readings from `cv_set=[8, 14, 17, 22, 30]` where each indexes correspond to the index of the recording file not in the `train_set`.The `cv_set` is being used for cross-validation. Each of the recording file fed through the filter gave us a quaternion estimation for each iteration, i.e. the file `01_undisturbed_slow_rotation_A` has approximately 56,000 sensor readings. Each sensor reading will correspond to a quaternion estimation(q_{esti}) and a ground truth(q_{gti}) quaternion. The error metric per iteration stands to be:

$$e_i = Quat_{abs\_dist}(q_{gti}, q_{esti}) \tag{4.1}$$

More information about the metric is provided in section A.2. We compare the mean of all e_i found over a particular recording for various standard filters such as the Madgwick and the Complementary filter to our proposed filter.

Chapter 5

Results

5.1 Results of noise Prediction

The XGB regression model run according to subsection 4.2.2 showed promising result in being able to predict the amount of noise in test set. The test set contained the recordings with indexes `cv_set=[8, 14, 17, 22, 30]`. The summary of the result is showed in Table 5.1. The r2 scores are reported for the `cv_set` and the test set generated after `train_test_split`.

Table 5.1: Final Hyperparameters for XGB RANSAC and Model Evaluation Results

Hyperparameters		**Test Results**	
`degree`	2	R^2 `Score Test Set`	0.8893
`colsample_bytree`	0.9274		
`learning_rate`	0.3028		
`max_depth`	15		
`n_estimators`	20		
`reg_alpha`	0.1242	R^2 `Score CV Set`	0.620
`reg_lambda`	0.7352		
`subsample`	0.6777		
`min_samples`	0.2392		
`residual_threshold`	5.0912		
`max_trials`	136		

5.2 Results of Movement Prediction

The XGB Classifier model run according to section 3.5 showed promising result in being accurate about prediction the movement. The training set contained all the recordings from 1 to 39. The summary of the result is showed in Table 5.2. The accuracy score is reported for the maximum accuracy acheived in the trial conducted by `Optuna Study`. The metric used for this is the `accuracy_score`[24].

Table 5.2: Final Hyperparameters for Movement Prediction XGB Classfier

Hyperparameters		**Test Results**	
`degree`	1		
`colsample_bytree`	0.9382		
`learning_rate`	0.6043		
`max_depth`	15		
`n_estimators`	22		
`reg_alpha`	2.02025	`Movement Prediction accuracy over the test split`	0.996
`reg_lambda`	2.8650		
`subsample`	0.9198		
`min_samples`	0.7556		
`residual_threshold`	2.3906		
`max_trials`	136		

5.3 Performance of Our Filter on BROAD Dataset Vs Standard Implementations

The experimental setup was implemented following the steps mentioned in subsection 4.2.3. The `cv_set` contains `[8, 14, 17, 22, 30]` file indexes. Thus results were generated based on these 5 files where noise removal was done. Results for simply performing toggling was evaluated over all the 39 recordings.

5.3.1 Results with only Toggling Engine, no noise Removal

In this subsection the charts of Mean errors and Error variance have been shown in Figure 5.1 and Figure 5.2 respectively. These charts were generated from the implementation of algorithm 1 but without the noise prediction model removing the noise from the sensor readings. The duty cycle chosen was 50% and α was chosen a low value of `0.001`.

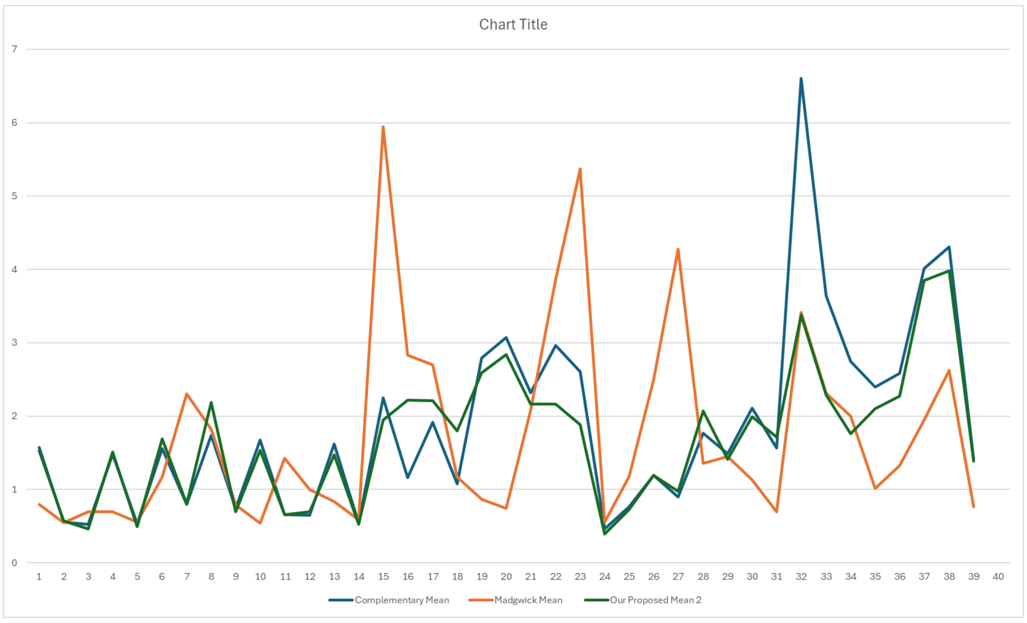


Figure 5.1: Mean of errors on all 39 recordings, for the Madgwick, Complementary , and our Complementary filtering with movement prediction and toggling with 50% duty cycle

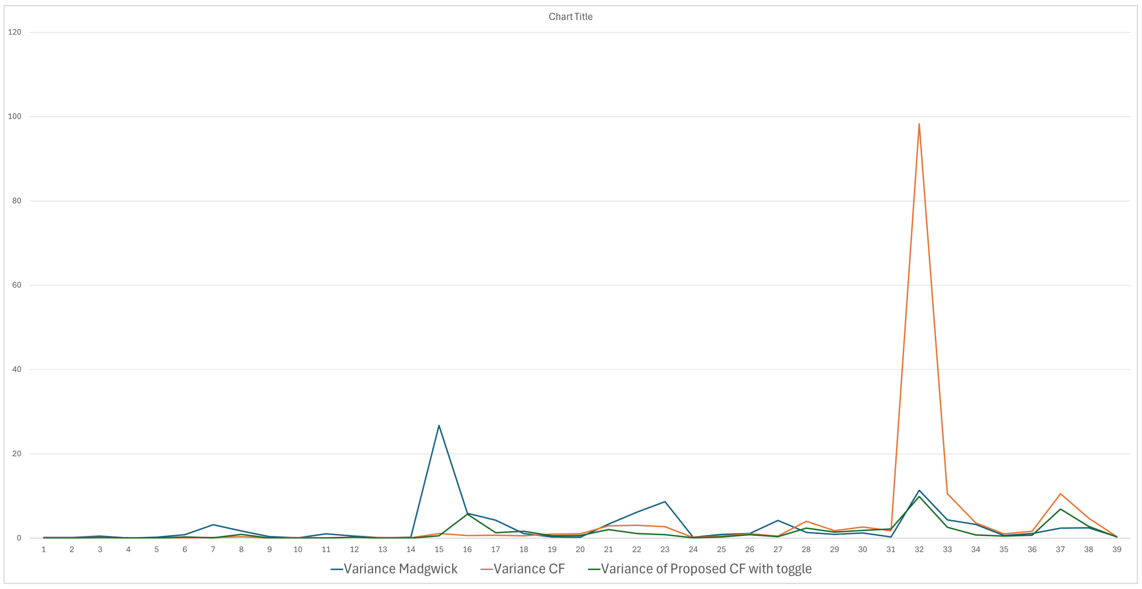


Figure 5.2: Variance of errors on all 39 recordings, for the Madgwick, Complementary , and our Complementary filtering with movement prediction and toggling with 50% duty cycle

5.3.2 Results of CF with α-toggling and Noise removal

This subsection displays the result of α-toggling noise removing Complementary filter which is the complete realization of algorithm 1. The charts only compare the results on 5 files of the `cv_set` containing `[8, 14, 17, 22, 30]`, as the other files were used to train the noise removal model. The mean and variance results are shown in Table 5.3 and Table 5.4 respectively. The chart in Figure 5.3 shows a comparision between the noise removal approach with other filters.

Table 5.3: Mean Error Statistics across the `cv_set`

Filename	**Basic CF**	**Proposed CF**	**MadgWick Filter**
09 undisturbed fast rotation with breaks B.hdf5	0.8134	0.7408	0.7899
15 undisturbed fast translation A.hdf5	3.1007	1.6212	5.9503
18 undisturbed fast translation with breaks B.hdf5	5.5009	1.2757	1.1728
23 undisturbed fast combined 360s.hdf5	3.6085	2.7388	5.3682
31 disturbed stationary magnet D.hdf5	2.4708	1.5798	0.6953

Table 5.4: Variance of Error Statistics across the `cv_set`

Filename	**Basic CF**	**Proposed CF**	**MadgWick Filter**
09 undisturbed fast rotation with breaks B.hdf5	0.1110	0.2603	0.3510
15 undisturbed fast translation A.hdf5	3.8544	0.9997	26.7320
18 undisturbed fast translation with breaks B.hdf5	31.1229	1.1998	1.1499
23 undisturbed fast combined 360s.hdf5	2.8928	1.1746	8.7074
31 disturbed stationary magnet D.hdf5	4.9321	1.5576	0.3195

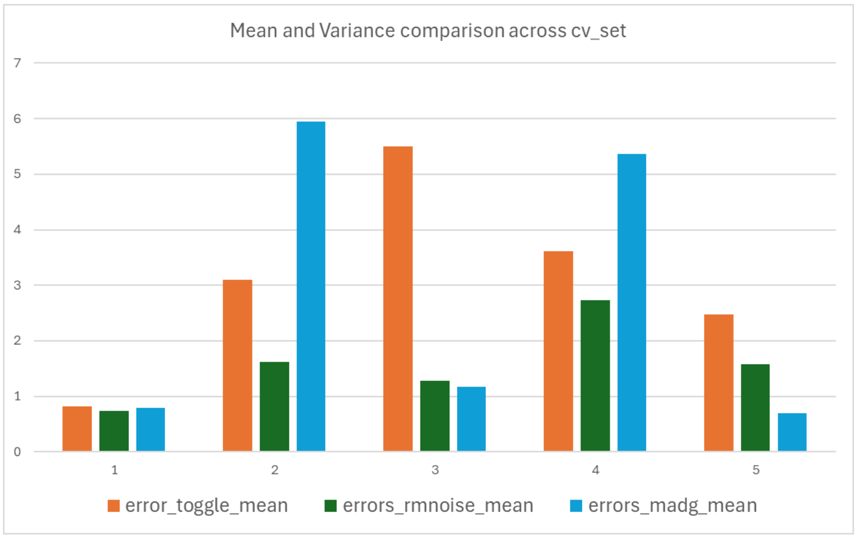


Figure 5.3: Mean and Variance comparison across cv_set. Filenames are sequentially arranged according to Table 5.3

5.3.3 Results of CF with α-Toggling and Noise Removal on Individual Files

This subsection illustrates the performance of the filter on a particular recording of the BROAD dataset over the iterations. Before demonstration of the graphs, the process how the error graphs on individual files have been generated should be stated.

- The individual record files have 50,000 to 300,000 sensor readings. Each reading correspond to a Quaternion Estimate q_{esti}.
- Each reading also corresponds to a ground truth quaternion q_{gti}.
- Their distance is calculated according to the process mentioned in subsection 4.2.3.
- This distance/error is plotted vs the iteration count in figures 5.4,5.5,5.6,5.7 and 5.8.
- **Legend:**
  - ***Red:*** Madgwick Filter
  - ***Green:*** Complementary Filter
  - ***Blue:*** Our Proposed Filter

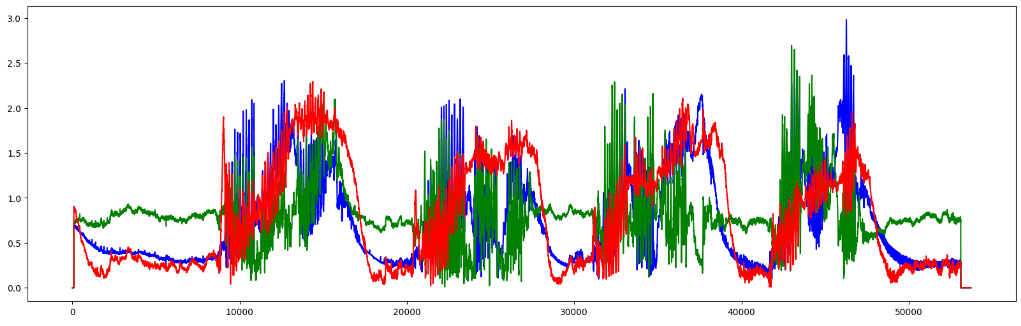


Figure 5.4: 9_undisturbed_fast_rotation_with_breaks_B

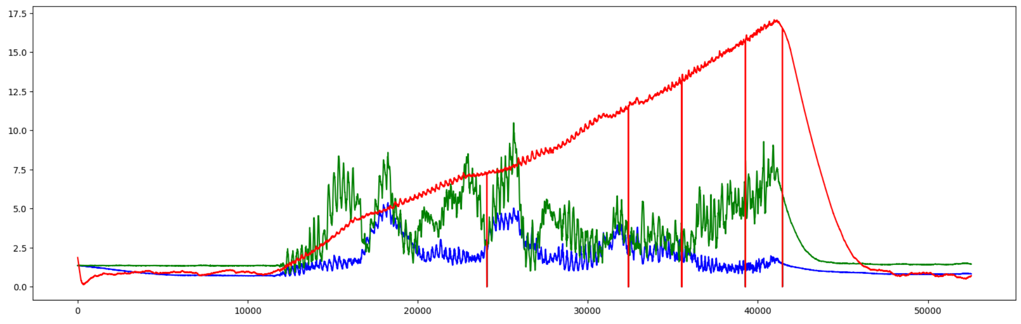


Figure 5.5: 15_undisturbed_fast_translation_A

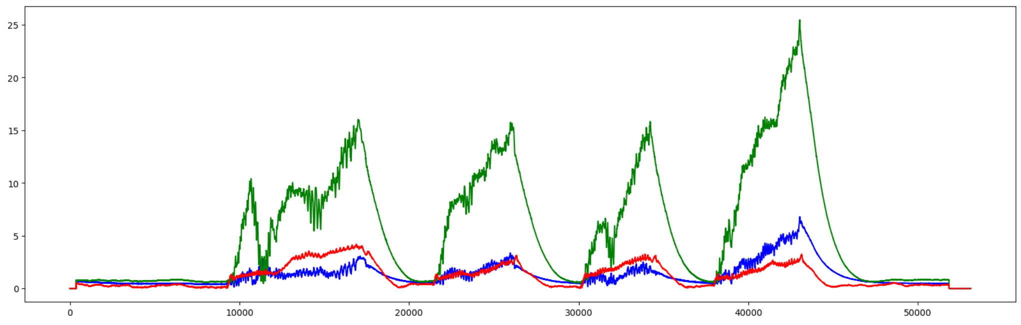


Figure 5.6: 18_undisturbed_fast_translation_with_breaks_B

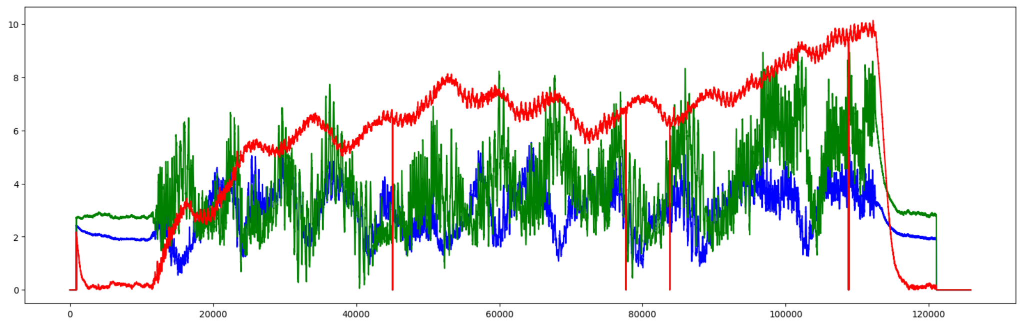


Figure 5.7: 23_undisturbed_fast_combined_360s

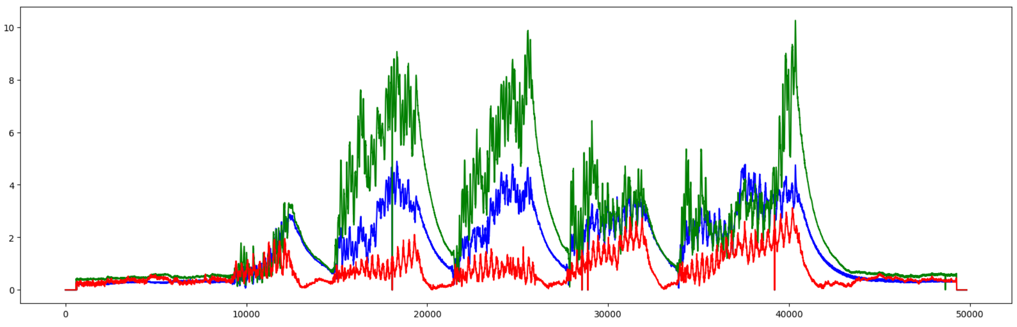


Figure 5.8: 31_disturbed_stationary_magnet_D

5.4 Performance of Our Filter against other filters on the Mechanical Platform

Here the results of running our proposed filter is tested along with the performaces of other standard existing filters - *AQUA* [4], *EKF* [32] and the MadgWick Gradient Descent filter [28].

Table 5.5: Static error metrics (°) per axis when ML denoising is active.

Metric	Proposed (ML)			Madgwick			EKF			Aqua		
	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
count	6869.000	6869.000	6869.000	...	...	...	...	...	...	...	...	...
mean	17.249	18.139	18.976	...	...	...	...	...	...	...	...	...
std	13.995	14.498	19.820	...	...	...	...	...	...	...	...	...
min	0.000457	0.000624	4.670	...	...	...	...	...	...	...	...	...
25%	0.044354	0.006244	4.670	...	...	...	...	...	...	...	...	...
50%	19.770	14.941	7.157	...	...	...	...	...	...	...	...	...
75%	30.153	32.723	30.253	...	...	...	...	...	...	...	...	...
max	42.025	52.562	84.391	...	...	...	...	...	...	...	...	...

Table 5.6: Static error metrics (°) per axis when ML denoising is not active.

Metric	Proposed (-ML)			Madgwick			EKF			Aqua		
	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
count	6869.000	6869.000	6869.000	...	...	...	...	...	...	...	...	...
mean	17.249	18.139	18.976	...	...	...	...	...	...	...	...	...
std	13.995	14.498	19.820	...	...	...	...	...	...	...	...	...
min	0.000457	0.000624	4.670	...	...	...	...	...	...	...	...	...
25%	0.044354	0.006244	4.670	...	...	...	...	...	...	...	...	...
50%	19.770	14.941	7.157	...	...	...	...	...	...	...	...	...
75%	30.153	32.723	30.253	...	...	...	...	...	...	...	...	...
max	42.025	52.562	84.391	...	...	...	...	...	...	...	...	...

Table 5.7: Quasi Static error metrics (°) per axis when ML denoising is active.

Metric	Proposed (ML)			Madgwick			EKF			Aqua		
	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
count	6869.000	6869.000	6869.000	...	...	...	...	...	...	...	...	...
mean	17.249	18.139	18.976	...	...	...	...	...	...	...	...	...
std	13.995	14.498	19.820	...	...	...	...	...	...	...	...	...
min	0.000457	0.000624	4.670	...	...	...	...	...	...	...	...	...
25%	0.044354	0.006244	4.670	...	...	...	...	...	...	...	...	...
50%	19.770	14.941	7.157	...	...	...	...	...	...	...	...	...
75%	30.153	32.723	30.253	...	...	...	...	...	...	...	...	...
max	42.025	52.562	84.391	...	...	...	...	...	...	...	...	...

Table 5.8: Quasi Static error metrics (°) per axis when ML denoising is not active.

Metric	Proposed (-ML)			Madgwick			EKF			Aqua		
	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
count	6869.000	6869.000	6869.000	...	...	...	...	...	...	...	...	...
mean	17.249	18.139	18.976	...	...	...	...	...	...	...	...	...
std	13.995	14.498	19.820	...	...	...	...	...	...	...	...	...
min	0.000457	0.000624	4.670	...	...	...	...	...	...	...	...	...
25%	0.044354	0.006244	4.670	...	...	...	...	...	...	...	...	...
50%	19.770	14.941	7.157	...	...	...	...	...	...	...	...	...
75%	30.153	32.723	30.253	...	...	...	...	...	...	...	...	...
max	42.025	52.562	84.391	...	...	...	...	...	...	...	...	...

Table 5.9: Dynamic error metrics (°) per axis when ML denoising is active.

Metric	Proposed (ML)			Madgwick			EKF			Aqua		
	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
count	6869.000	6869.000	6869.000	...	...	...	...	...	...	...	...	...
mean	17.249	18.139	18.976	...	...	...	...	...	...	...	...	...
std	13.995	14.498	19.820	...	...	...	...	...	...	...	...	...
min	0.000457	0.000624	4.670	...	...	...	...	...	...	...	...	...
25%	0.044354	0.006244	4.670	...	...	...	...	...	...	...	...	...
50%	19.770	14.941	7.157	...	...	...	...	...	...	...	...	...
75%	30.153	32.723	30.253	...	...	...	...	...	...	...	...	...
max	42.025	52.562	84.391	...	...	...	...	...	...	...	...	...

Table 5.10: Dynamic error metrics (°) per axis when ML denoising is not active.

Metric	Proposed (-ML)			Madgwick			EKF			Aqua		
	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
count	6869.000	6869.000	6869.000	...	...	...	...	...	...	...	...	...
mean	17.249	18.139	18.976	...	...	...	...	...	...	...	...	...
std	13.995	14.498	19.820	...	...	...	...	...	...	...	...	...
min	0.000457	0.000624	4.670	...	...	...	...	...	...	...	...	...
25%	0.044354	0.006244	4.670	...	...	...	...	...	...	...	...	...
50%	19.770	14.941	7.157	...	...	...	...	...	...	...	...	...
75%	30.153	32.723	30.253	...	...	...	...	...	...	...	...	...
max	42.025	52.562	84.391	...	...	...	...	...	...	...	...	...

Table 5.11: Resource consumption per filter on RPI: average CPU (%) and memory (MB).

Filter	CPU Usage (%)	Memory Usage (MB)
Proposed (ML)	[·]	[·]
Proposed (-ML)	[·]	[·]
Madgwick	[·]	[·]
EKF	[·]	[·]
Aqua	[·]	[·]

Table 5.12: Long-term drift over T hours (°/h).

Filter	Drift (°/hr)
Proposed (ML)	[·]
Proposed (-ML)	[·]
Madgwick	[·]
EKF	[·]
Aqua	[·]

Chapter 6

Interpretation Of Results

In this chapter, the results achieved in chapter 5 will be interpreted in detail. The results we have acheived so far is promising, and it is able to outperform standard Madgwick filter implementation in fast motion scenarios which was one of our goals initially.

6.1 Results Without Denoising

In Figure 5.1 and Figure 5.2 we can observe spikes in the orange line graph which represent the Madgwick Filter. Our proposed filter (in Green) is able to avoid high spikes in files `15_undisturbed_fast_translation_A`,`23_undisturbed_fast_combined_360s` and in `27_disturbed_phone_vibration_B` in comparisions of mean error in Figure 5.1. This indicates that in fast motion scenarios the toggling of α has a positive effect on the error in our proposed filter avoiding higher mean error. In case of variance comparison in Figure 5.2 we see that our proposed(Green line) has the least variance among all three.

6.1.1 Intuitive Reasoning Backed by Emperical Evidence

Here we suggest why α-toggling might result in a better performance.

- Toggling the α is done in scenarios when there is movement. When the model predicts there is some movement instead of varying the alpha, the alpha is toggled according to defined duty cycle and frequency, this gives just enough information for the gyroscope error to correct itself.
- When there is no movement(predicted by the model), then the α is set to its pre-

defined value. Here the assumption is that the system is much less noisy when there is less movement, and thus can employ more trust to the accelerometer and magnetometer by keeping the alpha to a constant value.

6.2 Results With Denoising

The results shown after denoising is much more promising than the results without denoising. In Table 5.3 and Table 5.4 we can observe that our proposed filter performs better than atleast one of the filters in all five trials of the `cv_set`. In case of variance of errors in Table 5.4, our filter consistently is able to maintain error variance around 1.0 where the other compared filters often exceed and reach a variance 10x or 20x of our proposed filter.

From the tables, it is observed that in Table 5.3, the best performance of our filter is visible in entries 2 and 4 which correspond to the files `15_undisturbed_fast_translation_A` and `23_undisturbed_fast_combined_360s` respectively. Thus our filter is able to outperform the standard Madgwick filter implementation in fast motion scenarios.

If we observe the error graphed vs iterations for `15_undisturbed_fast_translation_A` in Figure 5.5 in subsection 5.3.3, we observe that the Madgwick Filter's error steadily rises where our proposed complementary filter stays close to zero. For the file Figure 5.6, we observe that Madgwick filter performs marginally better.

6.2.1 Intuitive Reasoning Based on Emperical Evidence

- As noise is subtracted from the accelerometer and magnetometer readings, the extraneous linear acceleration is subtracted, as a result the orientation estimation is far superior in case of fast motions.
- It is observed from Figure 5.6 that the MadgWick filter stabilizes and converges to a good orientation when there is no motion, thus in the file `18_undisturbed_fast_translation_with_breaks_B` the Madgwick filter performs better than our filter as it has breaks to recover from its error. In case of Figure 5.5 for the file `15_undisturbed_fast_translation_A` the Madgwick filter is not able to stabilize itself,as there are no breaks and hence our filter performs better.

Chapter 7

Shortcomings in our Approach

Our approach is data driven, thus it would require us more time and memory complexity to perform inference from our models, which will nullify the mathematical simplicity of the complementary filter.

- **Size of the model:** The inference time of a tree based model is $O(log(n))$ where n is the number of nodes in the tree. The inference time is thus bound by the depth of the tree which was limited to 15. Thus for 6 variables and 15 estimators we would need $15 * 6 * 6$ decisions. This is not a large number , but the cause of concern would be the memory complexity as the trees of the XGBoost models are observed to be very wide.
- **Cost of Matrix Exponentiation:** Matrix exponentiation is used to rotate the vectors.
- **Training Time:** Training with RANSAC requires considerable amount of time.
- **Generalizability:** The models perform well in test set compared to the cv set.

Chapter 8

Future Works

This thesis has explored and implemented data-driven orientation estimation methods by combining machine learning models with complementary filtering techniques. Although the results have shown clear improvements compared to traditional approaches, there remain several limitations. This chapter discusses the challenges found and suggests possible directions for future research to improve the system's performance, ease of use, and real-world relevance.

8.1 Areas for Improvement

Algorithm and Computational Efficiency: The tree-based models we use, like XGBoost, require significant memory and processing power. This can be a problem, especially on small devices with limited resources. Also, matrix exponentiation used for orientation calculations needs a lot of computation, making real-time operation difficult on embedded platforms.

Training Time and Scalability: Using RANSAC during training helps handle outliers but also increases training time. This slows down scaling to larger datasets or thorough hyperparameter tuning. Additionally, because the models are trained on specific datasets, they might not generalize well to new, unseen data.

Mismatch Between Training and Live Data: The data used to train the models, like the BROAD dataset, differs in noise characteristics and motion from the live data collected during experiments. Such differences can reduce the model's accuracy when used in real-life conditions.

Evaluation Metrics: The quaternion absolute distance used to measure performance might be too strict, exaggerating small orientation errors. More suitable metrics that consider practical application needs should be explored.

8.2 Proposed Future Directions

8.2.1 Improve Computational Efficiency

To better suit embedded and low-power devices, work should focus on reducing memory use and speeding up inference. This can be done by pruning large models, using quantization to lower numerical precision, or applying knowledge distillation to create smaller models that perform similarly. Exploring simpler models like decision trees or linear models, possibly enhanced by smart feature design, could also help. Designing specialized hardware to accelerate machine learning computations could make real-time operation feasible on tiny devices.

8.2.2 Online and Adaptive Learning

Currently, models are trained once offline and not updated during use. Future research might explore online learning, where models continually update themselves based on new sensor data. This would help the system adapt to sensor aging, changing environments, or new types of motion, making it more robust and reducing the need for manual retraining. Adaptive algorithms that adjust parameters in real time based on sensor readings may further improve long-term performance.

8.2.3 Multi-Sensor Fusion

Adding other sensors to the system could increase accuracy and robustness. For example, GPS or ultra-wideband sensors can provide absolute position and heading to help correct drift. Cameras or depth sensors can offer environmental context useful for navigation and avoiding occlusions. Barometric pressure sensors might improve altitude estimation, indirectly helping orientation estimation. Future work should investigate fusion methods that weigh sensor data intelligently depending on context and sensor reliability.

8.2.4 Better Noise and Disturbance Models

Although noise has been reduced using regression models, there's still room for improvement, especially for time-dependent noise patterns.

Techniques like recurrent neural networks or probabilistic models may capture temporal correlations better. Domain-specific noise, such as vibrations in drones or magnetic interference in factories, should also be studied. Combining robust statistics and outlier detection with sensor fusion algorithms might reduce unwanted noise even more.

8.2.5 Deep Learning End-to-End Models

Instead of hybrid systems, future research could look at deep learning models that directly map raw sensor inputs to orientation estimates. Models such as convolutional neural networks, recurrent neural networks, or transformers are promising, but they need large labeled datasets and heavy computing power. Hybrid models that combine deep learning with classical filtering might deliver a good balance between accuracy and computational cost while maintaining interpretability.

8.2.6 Comprehensive Benchmarking and Evaluation

More testing of orientation algorithms on different datasets, IMU hardware, and sensor setups is needed to fully understand their strengths and weaknesses.

Developing a standard set of performance metrics that cover accuracy, computational load, latency, robustness, and resource use would allow fair comparison between methods and promote best practices.

8.2.7 Real-World Deployment and Testing

The current work mainly tests the method on controlled datasets and setups. Future research should implement these algorithms on real-world devices like smartphones, drones, or wearables. Testing in various real environments will better show how the system handles real-life disturbances, sensor noise, and motion complexity. Long-term trials will reveal drift behavior and system stability over time.

8.2.8 Standardization of Tools and Validation

The AHRS Python library used in this thesis has not been fully validated with respect to reference implementations. Future work should include thorough validation of this and similar libraries. Establishing consistent testing protocols and evaluation standards will enable reliable comparison of orientation estimation methods and advance the research field.

References

[1] "A robust complementary filter approach for attitude estimation of unmanned aerial vehicles using AHRS," in *Proceedings of the 2019 CEAS EuroGNC conference*, CEAS-GNC-2019-036, Milan, Italy, Apr. 2019.

[2] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2623–2631. DOI: `10.1145/3292500.3330701`

[3] A. Asgharpoor Golroudbari and M. Sabour, *End-to-end deep learning framework for real-time inertial attitude estimation using 6dof imu*, Feb. 2023. DOI: `10.22541/au.167628608.89589390/v1`

[4] A. Author1 and B. Author2, "Aqua: An efficient orientation filter for inertial and magnetic sensors," *Journal of Sensor Fusion*, vol. 5, pp. 123–134, 2010.

[5] I. Y. Bar-Itzhack, "Request - a recursive quest algorithm for sequential attitude determination," *Journal of Guidance, Control, and Dynamics*, vol. 19, no. 5, pp. 1034–1038, 1996. DOI: `10.2514/3.21742` eprint: `https://doi.org/10.2514/3.21742`. [Online]. Available: `https://doi.org/10.2514/3.21742`

[6] P. J. Bickel and K. A. Doksum, *Mathematical statistics: basic ideas and selected topics, volumes I-II package*. Chapman and Hall/CRC, 2015.

[7] H. D. Black, "A passive system for determining the attitude of a satellite," *AIAA journal*, vol. 2, no. 7, pp. 1350–1351, 1964.

[8] M. Bramer, "Avoiding overfitting of decision trees," *Principles of data mining*, pp. 119–134, 2007.

[9] M. Brossard, S. Bonnabel, and A. Barrau, "Denoising imu gyroscopes with deep learning for open-loop attitude estimation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4796–4803, 2020.

[10] A. Cariow and G. Cariowa, "A hardware-efficient approach to computing the rotation matrix from a quaternion," *arXiv preprint arXiv:1609.01585*, 2016.

[11] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[12] 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, no. 24, 2023, ISSN: 1424-8220. DOI: 10.3390/s23249756 [Online]. Available: https://www.mdpi.com/1424-8220/23/24/9756

[13] D. Duong Quoc, J. Sun, V. Le, and L. Luo, “Complementary filter performance enhancement through filter gain,” *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 8, pp. 97–110, Jul. 2015. DOI: 10.14257/ijsip.2015.8.7.10

[14] D. Engelsman and I. Klein, “Data-driven denoising of stationary accelerometer signals,” *Measurement*, vol. 218, p. 113 218, 2023.

[15] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel, “A complementary filter for attitude estimation of a fixed-wing uav,” in *2008 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, 2008, pp. 340–345.

[16] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981. DOI: 10.1145/358669.358692

[17] J. R. Forbes, “Fundamentals of spacecraft attitude determination and control [bookshelf],” *IEEE Control Systems Magazine*, vol. 35, no. 4, pp. 56–58, 2015.

[18] *Self Localization Using a 9DOF IMU Sensor With a Directional Cosine Matrix*, vol. Volume 4: 18th Design for Manufacturing and the Life Cycle Conference; 2013 ASME/IEEE International Conference on Mechatronic and Embedded Systems and Applications, International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Aug. 2013, V004T08A051. DOI: 10.1115/DETC2013-12906 eprint: https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2013/55911/V004T08A051/4257997/v004t08a051-detc2013-12906.pdf. [Online]. Available: https://doi.org/10.1115/DETC2013-12906

[19] S. Guo, J. Wu, Z. Wang, and J. Qian, “Novel marg-sensor orientation estimation algorithm using fast kalman filter,” *Journal of Sensors*, vol. 2017, no. 1, p. 8 542 153, 2017.

[20] W. T. Higgins, “A comparison of complementary and kalman filtering,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-11, no. 3, pp. 321–325, 1975. DOI: 10.1109/TAES.1975.308081

[21] J.-S. Hu and K.-C. Sun, “A robust orientation estimation algorithm using marg sensors,” *IEEE transactions on instrumentation and measurement*, vol. 64, no. 3, pp. 815–822, 2014.

[22] D. Q. Huynh, “Metrics for 3d rotations: Comparison and analysis,” *Journal of Mathematical Imaging and Vision*, vol. 35, pp. 155–164, 2009.

[23] G. Ke et al., “Lightgbm: A highly efficient gradient boosting decision tree,” in *Advances in Neural Information Processing Systems*, 2017, pp. 3146–3154.

[24] V. Labatut and H. Cherifi, “Accuracy measures for the comparison of classifiers,” *arXiv preprint arXiv:1207.3790*, 2012.

[25] D. Laidig, M. Caruso, A. Cereatti, and T. Seel, “Broad—a benchmark for robust inertial orientation estimation,” *Data*, vol. 6, no. 7, p. 72, 2021.

[26] U. López, L. Trujillo, Y. Martinez, P. Legrand, E. Naredo, and S. Silva, “Ransac-gp: Dealing with outliers in symbolic regression with genetic programming,” in *Genetic Programming*, Springer, Cham, 2017, pp. 114–130. DOI: 10.1007/978-3-319-55696-3_8

[27] S. O. H. 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, no. 4, pp. 2054–2064, 2020. DOI: 10.1109/TMECH.2020.2992296

[28] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, “Estimation of imu and marg orientation using a gradient descent algorithm,” in *2011 IEEE international conference on rehabilitation robotics*, Ieee, 2011, pp. 1–7.

[29] R. Mahony, T. Hamel, and J.-M. Pflimlin, “Nonlinear complementary filters on the special orthogonal group,” *IEEE Transactions on automatic control*, vol. 53, no. 5, pp. 1203–1218, 2008.

[30] J. Marins, X. Yun, E. Bachmann, R. McGhee, and M. Zyda, “An extended kalman filter for quaternion-based orientation estimation using marg sensors,” in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, vol. 4, 2001, 2003–2011 vol.4. DOI: 10.1109/IROS.2001.976367

[31] D. Maton et al., “Indirect tuning of a complementary orientation filter using velocity data and a genetic algorithm,” *Systems Science & Control Engineering*, vol. 12, no. 1, p. 2 343 303, 2024. DOI: 10.1080/21642583.2024.2343303 eprint: https://doi.org/10.1080/21642583.2024.2343303. [Online]. Available: https://doi.org/10.1080/21642583.2024.2343303

[32] P. S. Maybeck, *Stochastic Models, Estimation, and Control: Volume 1*. Academic Press, 1982.

[33] A. Nawrat, K. Jedrasiak, K. Daniec, and R. Koteras, “Inertial navigation systems and its practical applications,” *New approach of indoor and outdoor localization systems*, 2012.

[34] A. Sabatini, “Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing,” *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 7, pp. 1346–1356, 2006. DOI: 10.1109/TBME.2006.875664

[35] W. Shao, J. Zang, J. Zhao, and K. Liu, “A variable gain complementary filtering fusion algorithm based on distributed inertial network and flush air data sensing,” *Applied Sciences*, vol. 13, no. 14, 2023, ISSN: 2076-3417. DOI: 10.3390/app13148090 [Online]. Available: https://www.mdpi.com/2076-3417/13/14/8090

[36] M. K. Al-Sharman, Y. Zweiri, M. A. K. Jaradat, R. Al-Husari, D. Gan, and L. D. Seneviratne, “Deep-learning-based neural network training for state estimation enhancement: Application to attitude estimation,” *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 1, pp. 24–34, 2020. DOI: 10.1109/TIM.2019.2895495

[37] N. El-Sheimy and A. Youssef, “Inertial sensors technologies for navigation applications: State of the art and future trends,” *Satellite navigation*, vol. 1, no. 1, p. 2, 2020.

[38] 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. DOI: 10.2514/3.19717 eprint: https://doi.org/10.2514/3.19717. [Online]. Available: https://doi.org/10.2514/3.19717

[39] M. D. Shuster et al., “A survey of attitude representations,” *Navigation*, vol. 8, no. 9, pp. 439–517, 1993.

[40] D. Tedaldi, A. Pretto, and E. Menegatti, “A robust and easy to implement method for imu calibration without external equipments,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3042–3049. DOI: 10.1109/ICRA.2014.6907297

[41] R. G. Valenti, I. Dryanovski, and J. Xiao, “Keeping a good attitude: A quaternion-based orientation filter for imus and margs,” *Sensors*, vol. 15, no. 8, pp. 19 302–19 330, 2015.

[42] E. Vertzberger and I. Klein, “Adaptive attitude estimation using a hybrid model-learning approach,” *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–9, 2022.

[43] M. Vezočnik, R. Kamnik, and M. B. Juric, “Inertial sensor-based step length estimation model by means of principal component analysis,” *Sensors*, vol. 21, no. 10, 2021, ISSN: 1424-8220. DOI: `10.3390/s21103527` [Online]. Available: `https://www.mdpi.com/1424-8220/21/10/3527`

[44] G. Wahba, “A least squares estimate of satellite attitude,” *SIAM Review*, vol. 7, no. 3, pp. 409–409, 1965. DOI: `10.1137/1007077` eprint: `https://doi.org/10.1137/1007077`. [Online]. Available: `https://doi.org/10.1137/1007077`

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

[46] D. Weber, C. Gühmann, and T. Seel, “Neural networks versus conventional filters for inertial-sensor-based attitude estimation,” in *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, 2020, pp. 1–8. DOI: `10.23919/FUSION45008.2020.9190634`

[47] S. Wilson et al., “Formulation of a new gradient descent marg orientation algorithm: Case study on robot teleoperation,” *Mechanical Systems and Signal Processing*, vol. 130, pp. 183–200, 2019, ISSN: 0888-3270. DOI: `https://doi.org/10.1016/j.ymssp.2019.04.064` [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0888327019303012`

[48] S. Zhao, “Time derivative of rotation matrices: A tutorial,” *arXiv preprint arXiv:1609.06088*, 2016.

[49] Z. Zhe, W. Jian-bin, S. Bo, and T. Guo-feng, “Adaptive complementary filtering algorithm for imu based on mems,” in *2020 Chinese Control And Decision Conference (CCDC)*, 2020, pp. 5409–5416. DOI: `10.1109/CCDC49329.2020.9164809`

Appendices

Appendix A

Quaternion

A.1 Quaternion Fundamentals

A.1.1 Basic Operations with Quaternions

A quaternion q is a four-dimensional number composed of a scalar part and a three-dimensional vector part:

$$q = w + xi + yj + zk$$

where $w, x, y, z \in \mathbb{R}$, and i, j, k are imaginary units satisfying:

$$i^2 = j^2 = k^2 = ijk = -1$$

The basic operations on quaternions are:

- **Addition:** Given two quaternions $q_1 = (w_1, \mathbf{v}_1)$ and $q_2 = (w_2, \mathbf{v}_2)$, their sum is

$$q_1 + q_2 = (w_1 + w_2, \mathbf{v}_1 + \mathbf{v}_2)$$

- **Multiplication:** The product $q = q_1 \otimes q_2$ is defined as

$$q = (w_1 w_2 - \mathbf{v}_1 \cdot \mathbf{v}_2,\ w_1 \mathbf{v}_2 + w_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2)$$

  where $\cdot$ denotes the dot product and $\times$ denotes the cross product.

- **Conjugate:** The conjugate of $q = (w, \mathbf{v})$ is

$$q^* = (w, -\mathbf{v})$$

- **Norm:** The norm of q is given by

$$||q|| = \sqrt{w^2 + x^2 + y^2 + z^2}$$

- **Inverse:** For a nonzero quaternion,

$$q^{-1} = \frac{q^*}{||q||^2}$$

A.1.2 Rotating a Vector Using a Quaternion

To rotate a 3D vector $\mathbf{v} \in \mathbb{R}^3$ using a unit quaternion q, the following procedure is used:

1. Represent $\mathbf{v}$ as a pure quaternion:

$$v_q = (0, \mathbf{v})$$

2. Apply the rotation:

$$v_{\text{rotated}} = q \otimes v_q \otimes q^*$$

   where q^* is the conjugate of q, and $\otimes$ denotes quaternion multiplication.

3. Extract the vector part of v_{rotated} to obtain the rotated vector.

The quaternion rotation formula ensures a smooth and gimbal-lock-free rotation, which is particularly advantageous in 3D applications such as orientation tracking and robotics.

A.2 Absolute Distance

In this section, we describe the computation of the *Quaternion Absolute Distance* (QAD), a common metric used to evaluate differences between two orientation quaternions.

Given two unit quaternions:

- $\mathbf{q}_1 = [w_1, x_1, y_1, z_1]$
- $\mathbf{q}_2 = [w_2, x_2, y_2, z_2]$

The absolute distance $Quat_{abs\_dist}$ between them is defined as:

$$Quat_{abs\_dist}(\mathbf{q}_1, \mathbf{q}_2) = 1 - |\langle \mathbf{q}_1, \mathbf{q}_2 \rangle| = 1 - |w_1 w_2 + x_1 x_2 + y_1 y_2 + z_1 z_2| \tag{A.1}$$

This distance is derived from the dot product between the two quaternions and captures the angular difference in orientation. Since quaternions $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation, taking the absolute value of the dot product ensures the distance is invariant to this ambiguity.

A.2.1 Properties

- $Quat_{abs\_dist}(\mathbf{q}_1, \mathbf{q}_2) = 0$ if $\mathbf{q}_1 = \mathbf{q}_2$ or $\mathbf{q}_1 = -\mathbf{q}_2$
- $Quat_{abs\_dist} \in [0, 1]$ for normalized quaternions
- The closer the value is to 0, the more similar the orientations

A.2.2 Use Cases

Quaternion Absolute Distance is frequently used in:

- Orientation estimation
- Motion capture and sensor fusion
- Evaluating the accuracy of inertial tracking algorithms