Inter IIT 13.0 Midterm Report

Position and Attitude Control of Quadcopter with Single Motor Failure

Team Name: Team 62

November 10, 2024

Table of Contents

1 Approach Outline

1.1 Problem Understanding

Quadcopters, while offering stability and ease of control due to their symmetrical design, are inherently sensitive to motor failures. Due to the single-motor failure in a quadcopter, the drone becomes unstable due to an unopposed roll and pitch torque. To combat this issue effectively, it becomes vital to detect the position of the failed motor and power off the diagonally opposed motor to achieve some amount of stability using two-motor dynamics of the quadcopter. Once the quadcopter achieves a certain state, control algorithms can be triggered to complete certain tasks. The challenges in the problem statement lies in reducing the detection latency of the motor failure and developing a suitable control algorithm. The development of suitable control algorithm has certain challenges mainly due to the unequal thrust distribution to individual motors which accounts for the natural shift in center of gravity of drone and needs to be estimated accurately. Apart from this, reducing the computational cost of the algorithms poses a significant challenge as higher computation cost results in lower publishing frequency of commands to the motors which results in irregular control of drone pose.

1.2 Solution Overview

The project's solution focuses on two key areas:

1. **Motor Failure Detection**: Aims to detect motor failure with minimal latency through **adaptive thresholding**. This involves monitoring key odometry data (involving acceleration, pitch, and yaw) and **dynamically adjusting** detection thresholds to enhance reliability and **reduce detection time**.

2. **Control Strategy**: Once a failure is detected, the remaining three motors are controlled using LQR controller. This strategy aims to stabilize the quadcopter in flight, or if stabilization isn't feasible, to initiate a controlled and safe landing.

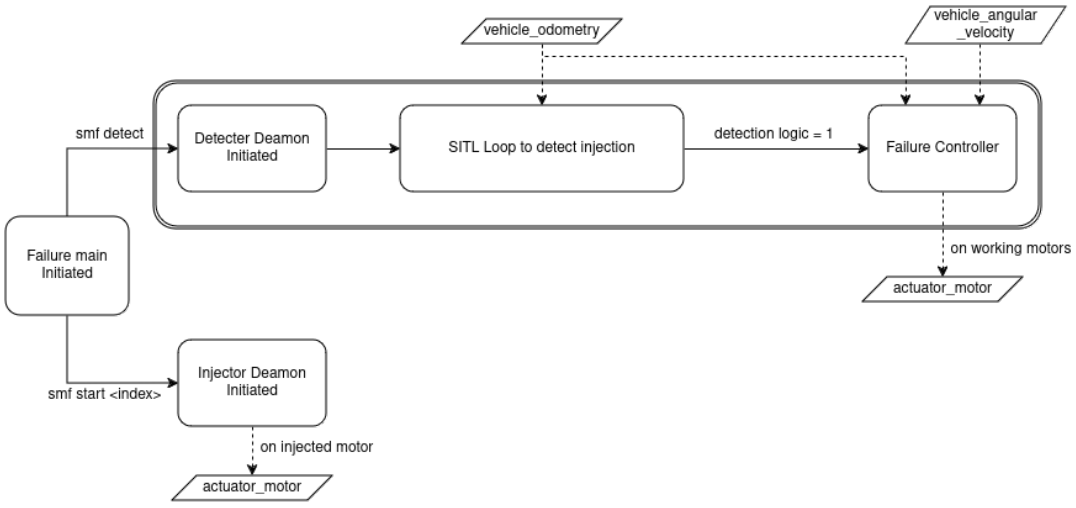

Figure 1: High-level Architecture for `single_motor_failure` module

The system operates as follows:

- Upon initiating the Software-in-the-Loop (SITL) simulation using `smf` command, `failure_main` component triggers.
- The `Detector Daemon` can be initiated using the command `smf detect` in the SITL shell. This daemon continuously monitors the `vehicle_odometry` topic to identify any anomalies indicative of motor failure. The detection logic is designed to recognize failures promptly.
- Upon executing the command `smf start <index>` in the SITL shell, the `Injector Daemon` is activated simultaneously. This daemon is responsible for publishing values to the specified motor index, thereby simulating a failure.
- Once a failure is detected, the `Failure Controller` takes precedence over the default PX4 control algorithm. This controller dynamically adjusts thrust to compensate for the malfunction and maintains stability through the control strategy.

1.3 Background Study

Relevant literature on UAV fault-tolerant control and motor failure response underpins our approach. Studies indicate that adaptive thresholding and feedback control strategies are effective in real-time UAV stabilization, guiding our implementation of these methods for enhanced reliability in single-motor failure scenarios.

To strengthen our understanding of quadcopter dynamics and control approaches, we have conducted a thorough literature survey of previous work understanding their methodology and feasibility in our scenario. According to the previous research work conducted in [1], [2] and [3], there can be two approaches in applying control algorithms on the drone, either we can employ a control algorithm on the **Thrust and Torque** as done in [2] or we can directly control the **actuator thrust** as done in [1] and [3]. The later approach reduces internal computation cost as no additional computation is done to translate thrust and torque to motor speeds and also has faster response. For our state space analysis, we modeled our system as provided in [3]. The state-space equation has been linearized using Jacobian about origin and aids in applying methods such as LQR to be able to find the gain matrix.

2 Analysis and Initial Experiments

2.1 Simulation Setup

Our simulation environment is configured using ROS2 Humble and Gazebo Harmonic as the simulation infrastructure. We employ the **PX4 x500 quadrotor** as the drone model, which serves as an effective platform for testing motor failure scenarios. To establish the necessary communication with the PX4 flight controller, we have integrated **QGroundControl** and **Micro DDS Agent**.

We developed a custom PX4 module named `single_motor_failure` (smf) to enhance our simulation capabilities. This module is designed to inject motor failures, detect these failures, and apply custom control strategies in response. To achieve this,

we utilize **uORB** topics within the PX4 ecosystem. Specifically, we publish to certain topics to inject failures and implement control measures, while subscribing to odometry topic to detect motor failures. This setup allows us to effectively simulate and manage motor failure scenarios in our drone model.

To achieve stable hover during a single motor failure, we **doubled** the **motor constant value** in the x500 quadrotor's SDF (Simulation Description Format) file. This modification enhances the thrust output of the remaining motors, enabling the drone to maintain stability and hover effectively even with one motor out of operation.

2.2 Motor Injection Functionality

To emulate real-time motor failure, we designed an injection daemon integrated within the PX4 and Gazebo environment. This capability allows us to simulate various failure scenarios and observe the quadcopter's response. The steps for motor injection are as follows:

- **Injector Daemon Initiation**: Upon initiation, the injector daemon takes a motor index as an argument. It targets the `actuator_motor` uORB topic and continuously injects a value of zero to the specified motor.
- **Motor Command Interception**: By injecting an NaN value, the selected motor is effectively prevented from receiving commands from the PX4 controller. Meanwhile, the other three motors continue to receive commands as usual, maintaining their normal operation.
- **Simulation of Single Motor Failure**: This setup simulates a single motor failure in the quadcopter, allowing for the testing of control algorithms and system responses under failure conditions.

2.3 Motor Detection Functionality

Following the motor injection functionality, the motor detection functionality is crucial for identifying failures in real-time. This process leverages an **adaptive thresholding** approach to analyze key UAV state parameters, ensuring rapid and accurate detection of motor failures.

- **Continuous Monitoring with Detection Daemon**: Upon initiation, the detection daemon enters a continuous loop, monitoring data from the `vehicle_odometry` uORB topic. This setup allows for real-time analysis of flight data to identify potential motor failures.
- **Feature Engineering and Analysis**: We conducted multiple data sessions to plot and analyze odometry parameters, focusing on features such as `average_acceleration_Z` and `avg_roll_rate`. These features are critical in determining motor failure conditions.
- **Dynamic Thresholding Approach**: Instead of relying on fixed thresholds, which may be inadequate due to the dynamic nature of flight, we employ dynamic thresholding. This method adjusts thresholds based on the slopes of `avg_roll_rate`

and avg_acc_z, along with their respective sensitivity factors. This adaptive approach significantly **reduces** detection latency from **80-90ms** (constant thresholding) to **25-35ms**.

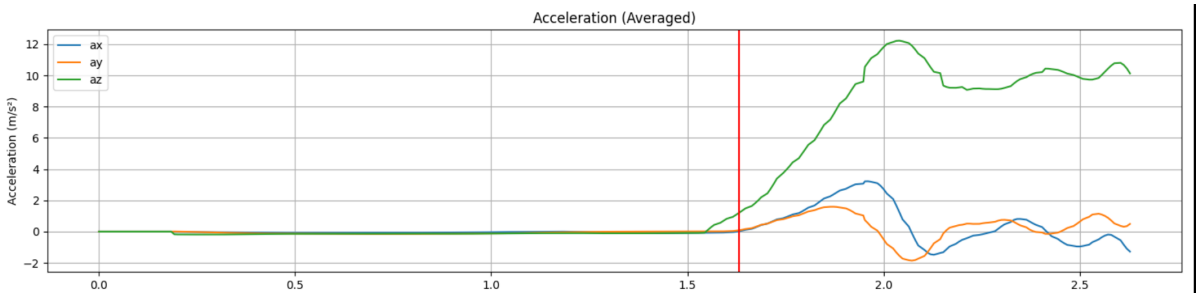

Figure 2: Odometric plot for average_acceleration_Z

- **Precise Motor Index Identification**: After detecting a failure, the module further analyzes avg_yaw_rate and avg_pitch_rate to accurately identify the failed motor. These parameters create a combination of four cases based on their concavity and convexity post-failure detection, allowing for precise motor index estimation.

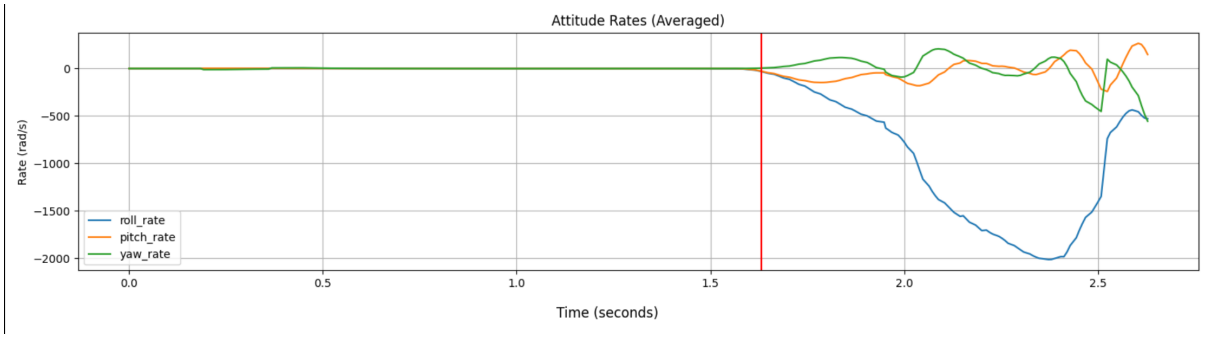

Figure 3: Odometric plot for avg_roll_rate, avg_yaw_rate and avg_pitch_rate

  - **Red Vertical Line:** Represents the detection time, marking the moment when the failure was identified based on dynamic thresholding criteria.
  - **Injection/Detection Time:** Indicates the number of microseconds elapsed since boot time.
  - **Latency:** Calculated as the difference between detection time and injection time.

***Note:** Motor indices are defined using 1-based indexing.

2.4 Quadrotor Dynamics

In this section, we present the equations as derived in [3]. The equations are as follows:

Before presenting the equations, we define the terms used:

- I_{xx}^B, I_{zz}^B: Moments of inertia of the drone along the x and z axes, respectively, in the body frame.
- I_{zz}^P: Moment of inertia of the propeller about its axis of rotation, i.e., the z axis.
- $\omega_1, \omega_2, \omega_3, \omega_4$: Angular velocities of the rotors.
- l: Distance of each rotor from the center of the drone.
- p, q, r: Roll, pitch, and yaw rates, respectively.
- κ_f: Thrust coefficient, representing the relationship between rotor speed and thrust.
- κ_τ: Torque coefficient, representing the relationship between rotor speed and torque.
- γ: Yaw damping coefficient, which accounts for the damping effect in the yaw motion.
- $\bar{f}_1, \bar{f}_2, \bar{f}_3, \bar{f}_4$: Forces generated by the motors.
- $\bar{n}$: Normal vector of the drone in the ground frame.
- $\bar{\omega}^B$: Vector of roll, pitch, and yaw rates, denoted as (p, q, r).
- ρ: Ratio of the force produced by the opposite working motor of a failed motor to the force produced by the adjacent motor of the failed motor.
- Further, we will consider that I_{zz}^T and I_{xx}^T equal I_{zz}^B and I_{xx}^B respectively because the propeller's moment of inertia is very small compared to the remaining drone's body.

$$I_{xx}^B \dot{p} = \kappa_f \left(\omega_2^2 - \omega_4^2\right) l - \left(I_{zz}^T - I_{xx}^T\right) qr - I_{zz}^P q \left(\omega_1 + \omega_2 + \omega_3 + \omega_4\right) \tag{1}$$

$$I_{xx}^B \dot{q} = \kappa_f \left(\omega_3^2 - \omega_1^2\right) l + \left(I_{zz}^T - I_{xx}^T\right) pr + I_{zz}^P p \left(\omega_1 + \omega_2 + \omega_3 + \omega_4\right) \tag{2}$$

$$I_{zz}^B \dot{r} = -\gamma r + \kappa_\tau \kappa_f \left(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2\right) \tag{3}$$

$$\bar{f}_\Sigma = \bar{f}_1 + \bar{f}_2 + \bar{f}_3 + \bar{f}_4 \tag{4}$$

$$\bar{n} = \epsilon \, \bar{\omega}^B \tag{5}$$

$$\|\bar{n}\| = \|\epsilon \, \bar{\omega}^B\| = 1 \tag{6}$$

$$\bar{f}_\Sigma \, \bar{n}_z = m\|\mathbf{g}\| \tag{7}$$

$$\rho = \frac{\bar{f}_2}{\bar{f}_1} \tag{8}$$

The values of the inertias (I_{xx}^B, I_{yy}^B, I_{zz}^P) and the distance l are taken from the SDF file as $I_{xx}^B = 0.02\,\text{kg} \cdot \text{m}^2$, $I_{yy}^B = 0.04\,\text{kg} \cdot \text{m}^2$, $I_{zz}^B = 2.0 \times 10^{-5}\,\text{kg} \cdot \text{m}^2$ and $l = 0.25m$.

To calculate the thrust coefficient κ_f, we set the rotor velocity to a maximum of 1000 rad/s, as specified in the SDF file. We then measured the vertical acceleration of the system. The total thrust force $\bar{f}_\Sigma$ is given by the sum of the forces produced by each rotor:

$$\bar{f}_\Sigma = \bar{f}_1 + \bar{f}_2 + \bar{f}_3 + \bar{f}_4 \tag{9}$$

Each individual rotor force is calculated using the equation:

$$f_i = \kappa_f \omega_i^2 \tag{10}$$

where ω_i is the angular velocity of the i-th rotor. The total thrust force can also be expressed in terms of the mass m of the drone, gravitational acceleration g, and the measured vertical acceleration:

$$\bar{f}_\Sigma = m(g + \text{vertical\_acceleration}) \tag{11}$$

By equating the two expressions for $\bar{f}_\Sigma$, we can solve for the thrust coefficient κ_f:

$$\kappa_f = \frac{m(g + \text{vertical\_acceleration})}{\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2} \tag{12}$$

This method allows us to determine κ_f based on the maximum rotor speed and the observed vertical acceleration.

To calculate the yaw damping coefficient γ, we set the drone to rotate with a constant angular velocity (yaw rate). The torque produced by the four motors is given by:

$$\tau_{\text{motors}} = \kappa_\tau \kappa_f (\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \tag{13}$$

where κ_τ is the torque coefficient, κ_f is the thrust coefficient, and ω_i are the angular velocities of the rotors.

The torque due to drag, which opposes the rotation, is expressed as:

$$\tau_{\text{drag}} = \gamma \cdot \text{yaw\_rate} \tag{14}$$

At a constant angular velocity, these torques are balanced, so we equate them:

$$\kappa_\tau \kappa_f (\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) = \gamma \cdot \text{yaw\_rate} \tag{15}$$

Solving for γ, we get:

$$\gamma = \frac{\kappa_\tau \kappa_f (\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)}{\text{yaw\_rate}} \tag{16}$$

This method allows us to determine γ by measuring the yaw rate and using the known values of the torque and thrust coefficients.

By solving eqs.(12) to (16), we get the values: $\kappa_f = 1.456 \times 10^{-5}\,\text{N} \cdot \text{s}^2$, $\kappa_\tau = 0.01\,\text{N} \cdot \text{m} \cdot \text{s}^2$ and $\gamma = 0.007\,\text{N} \cdot \text{m} \cdot \text{s}$.

To get equilibrium values, we set the angular accelerations to zero, as there is no change in angular velocities at equilibrium. That means $\dot{p}$, $\dot{q}$, $\dot{r}$ are all zero at equilibrium. This simplifies the system, allowing us to solve for the equilibrium conditions where the forces and moments balance. By doing so, we can derive the steady-state equations that describe the system's behavior when it is in equilibrium. So from solving, we get finalized equations as follows (when motor 4 is failed):

$$r = \left(\frac{\kappa_T \cdot \kappa_f}{\gamma}\right)\left(\frac{2}{\rho} - 1\right)\omega_2^2 \tag{17}$$

$$q = \frac{\kappa_f \, l \, \omega_2^2}{\left((I_{zz}^T - I_{xx}^T) \left(\frac{\kappa_T \, \kappa_f}{\gamma} \right) \left(\frac{2}{\rho} - 1 \right) \omega_2^2 + I_{zz}^P \left(\frac{2}{\sqrt{\rho}} + 1 \right) \omega_2 \right)} \tag{18}$$

$$p = 0 \tag{19}$$

$$n_Z = \frac{mg}{\kappa_f \left(\frac{2}{\rho} + 1 \right) \omega_2^2} \tag{20}$$

Now we find ω_4 from the below equation using $\rho = 0.5$:

$$\left(\frac{\kappa_f \, l \, \omega_2^2}{\left((I_{zz}^T - I_{xx}^T) \left(\frac{\kappa_T \, \kappa_f}{\gamma} \right) \left(\frac{2}{\rho} - 1 \right) \omega_2^2 + I_{zz}^P \left(\frac{2}{\sqrt{\rho}} + 1 \right) \omega_2 \right)} \right)^2 = \tag{21}$$

$$\left(\left(\frac{\kappa_T \cdot \kappa_f}{\gamma} \right) \left(\frac{2}{\rho} - 1 \right) \omega_2^2 \right)^2 \left(\left(\frac{\kappa_f \left(\frac{2}{\rho} + 1 \right) \omega_2^2}{mg} \right)^2 - 1 \right)$$

By substituting the previously determined constants, we obtain the following equilibrium values: $p = 0$ rad/s, $q = 2.532$ rad/s, $r = 8.50$ rad/s, $f_1 = 8.83$ N, $f_2 = 4.415$ N, $f_3 = 8.83$ N, $n_X = 0$, $n_Y = -0.285$, and $n_Z = 0.989$.

2.5 Quadrotor Control

We have tried multiple approaches to provide suitable control under motor failure: (a) Linear–quadratic regulator (LQR), (b) Sliding Mode Control, (c) Proportional Controller.

2.5.1 LQR Control System

The equilibrium values of p, q, n_x, and n_y obtained in section 2.4 are used to provide suitable control using a Linear–quadratic regulator (LQR) control system. LQR provides an algorithm used to regulate the state of a dynamic system by minimizing a cost function. This cost function(J) typically balances the inputs against the accuracy of tracking desired states. The cost function is given by:

$$J = x^T(t_1) F(t_1) x(t_1) + \int_{t_0}^{t_1} \left(x^T Q x + u^T R u + 2 x^T N u \right) \, dt \tag{22}$$

$$u = -K \cdot x \tag{23}$$

The control gain K is given by:

$$K = R^{-1} \left(B^T P(t) + N^T \right) \tag{24}$$

where P is found by solving the continuous-time Riccati differential equation:

$$A^T P(t) + P(t) A - (P(t) B + N) \, R^{-1} \left(B^T P(t) + N^T \right) + Q = -\dot{P}(t) \tag{25}$$

with the boundary condition:

$$P(t_1) = F(t_1). \tag{26}$$

where

- Q: The state weighting matrix, which penalizes deviations in the state x.
- R: The control weighting matrix, which penalizes the control effort u.
- N: The cross-weighting matrix that penalizes the interaction between state x and control u.
- t_0: The initial time of the control interval.
- t_1: The final time of the control interval.

2.5.2 Proportional Controller

For our initial testing, we have employed a proportional controller to do controlled landing of the drone. After the detection of the motor index for the failed motor, the diagonally opposite motor is turned on periodically to generate the required roll and pitch to maintain a stable attitude and x, y pose during descent. The proportional control law is applied to the remaining motors and equal in value to avoid changes in roll and pitch, the value is proportionally brought to zero as altitude decreases and the max value of the controller is clipped to avoid generating a net upward thrust at any point.

$$f_4 = 0 \tag{27}$$

$$f_1 = f_3 = min(alt, f_{max}) \tag{28}$$

$$f_2 = \begin{cases} min(alt, 0.83 f_{max}), & (yaw_{rad} - yaw_{init}) \geq 0.2 \\ 0, & otherwise \end{cases} \tag{29}$$

2.6 Preliminary Results

To evaluate our motor detection module, we created an automation script to run and log detection results across multiple scenarios. A dataset of 248 sessions was compiled, covering different operational states:

$$K = R^{-1} \left(B^T P(t) + N^T \right) \tag{30}$$

- **Hovering**: 128 sessions, varying heights and motor positions.
- **Takeoff**: 60 sessions simulating motor failure at various points in ascent.
- **Landing**: 60 sessions simulating motor failure during descent.

The table below summarizes the results of all the sessions conducted during several sessions, where motor failures were introduced at different heights and for each motor individually.

Metric	Hovering	Takeoff	Landing
Total Sessions	128	60	60
Accuracy (%)	93.75	90.0	91.67
Correct Detection Motor 1	31	13	14
Correct Detection Motor 2	29	15	12
Correct Detection Motor 3	31	13	15
Correct Detection Motor 4	29	13	14
Average Latency (s)	0.03556	0.02907	0.03427

Table 1: Combined Detection Performance Metrics for Hovering, Takeoff, and Landing

From the implementation the LQR controller, the controller gain matrix came out to be:

$$K = \begin{bmatrix} -1.564865 & 0.086021 & 0 & 0 \\ 0.086021 & 1.564865 & 0 & 0 \end{bmatrix}. \tag{31}$$

2.7 Observations and Adjustments

Observations from initial trials led to several adjustments:

- Enhanced the adaptive thresholding module by fine-tuning sensitivity factors, achieving faster detection times.
- Set the LQR parameters Q and R. In the state weighing matrix Q, we set those elements relatively larger which were pertaining to change in N_x and N_y so as to achieve faster convergence in these states. All of these were made such that Q and R satisfy the necessary conditions for LQR controller.

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 1 & 0 & 0 & 20 \end{bmatrix} \tag{32}$$

$$R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{33}$$

- Modified(doubled) the motor constant value in the x500 quadrotor's SDF file to enhance the thrust output of the motors, so that the vehicle maintains stability and hover effectively with the loss of one motor.

2.8 Challenges

- The code for controller in the file `Team_62/px4_modules/single_motor_failure/failure_controller.cpp` implements LQR controller(as discussed in 2.5.1) with which we are yet to achieve a stable hover. However, we could achieve a controlled landing using a proportional controller(as discussed in 2.5.2) which we demonstrated in the video file `Team_62/simulation-videos/controlled_landing_demo.mp4`.

3 Future Scope

In the ongoing development of our drone's motor failure detection and control system, we will be employing Proportional-Derivative (PD) and Linear Quadratic Regulator (LQR) methods. These methods have been chosen for their effectiveness in linear control scenarios and their ability to provide a stable response to motor failures.

1. **Optimization of Control Algorithms**: Our primary focus will be on optimizing the control algorithms by improving the PID response parameters and LQR parameters. Additionally, we will explore additional control strategies to enhance stability and performance.
2. **Enhanced Detection Module**: We aim to refine the motor detection algorithm to further reduce latency and improve failure localization, ensuring quicker and more accurate responses to motor failures.
3. **Advanced Navigation**: We plan to extend the system's capabilities to include advanced navigation features, such as implementing safe return-to-home capabilities in cases where hover stability is maintained post-failure.
4. **Transition to Nonlinear Approaches**: Should the current methods not yield satisfactory results, we plan to explore more advanced control strategies, such as Incremental Nonlinear Fault-Tolerant Control [1]. This approach offers the potential for improved adaptability and robustness in handling complex failure scenarios.

4 Conclusion

Our solution effectively handles the problem of failure and failed motor detection using adaptive thresholding. We considered various control schemes and were able to achieve a controlled landing using a proportional controller. Based on our observations and current results, We plan to leverage upon our LQR controller and implement fault tolerant controllers for achieving a stable hover and navigation, testing non-linear controllers as well. We would be working on improving our module by improving it's scalability and carry out testing on hardware.

References

[1] Sun, Sihao & Wang, Xuerui & Chu, Qiping & De Visser, Coen. (2020). Incremental Nonlinear Fault-Tolerant Control of a Quadrotor With Complete Loss of Two Opposing Rotors. IEEE Transactions on Robotics.

[2] Sun, Sihao, Cioffi, Giovanni, de Visser, Coen, and Scaramuzza, Davide. "Autonomous Quadrotor Flight Despite Rotor Failure With Onboard Vision Sensors: Frames vs. Events." IEEE Robotics and Automation Letters.

[3] Mueller, Mark W., and D'Andrea, Raffaello. "Stability and Control of a Quadrocopter Despite the Complete Loss of One, Two, or Three Propellers." Proceedings of the 2014 IEEE International Conference on Robotics & Automation (ICRA), 2014.