ideaForge™
Create. Inspire

ideaForge

Position and Altitude Control of Quadrotor with Single Motor Failure

Team 27

Introduction

Problem Understanding

Quadrotors are extensively used for applications in aerial surveillance, mapping, surveying, aerial photography, search and rescue, agriculture, logistics delivery, and other domains due to their structural simplicity, easy deployability, and capability to hover and manoeuvre in confined spaces. Unlike fixed-wing drones, quadrotors can perform vertical take-offs and landings, making them ideal for operations in cluttered or inaccessible environments.

Their ability to remain stationary mid-air and navigate through tight areas has made them indispensable for critical missions such as disaster relief, military reconnaissance, and infrastructure inspection. Furthermore, advances in lightweight materials, battery technology, and compact sensors have significantly enhanced their versatility and operational range.

However, the increasing reliance on quadrotors for high-stakes applications exposes them to risks inherent in their operating environments. Propeller damage or motor failure is a common issue due to collisions with obstacles, wear and tear, or adverse weather conditions. Such failures pose serious challenges, as the loss of even one motor disrupts the vehicle's thrust symmetry and stability, making it difficult to maintain normal operation.

In disaster relief, for example, a failed quadrotor could lead to delays in delivering medical supplies or assessing damage, potentially endangering lives. In military applications, the failure of a quadrotor during a reconnaissance mission in hostile territory could result in technology exfiltration or loss of sensitive intelligence. To mitigate these risks, there is a pressing need for robust fault-tolerant control (FTC) systems that can detect, isolate, and respond to failures in real-time, ensuring the quadrotor can continue its mission or safely return home.

- After motor failure, the remaining three working rotors may insufficiently compensate for the drone's weight and consequently result in non-zero moments.
- The **yaw** & the **torque** about one of the arms of the drone, would initially be large and **uncontrolled**.The designed control algorithm should be equipped to handle scenarios.
- Lack of feedback sensors limits fault detection speed and accuracy due to **baud rates** and noise levels, which may cause false positives.
- Even after accurate fault detection and isolation, the modified control algorithm would not have the luxury of accessing all maneuvers present previously with **no yaw** control.

For the mid-evaluation, we explored previous approaches to address these issues, including Fault Detection and Isolation using accelerometer data, a PD-based controller for emergency landing after motor failure, and a LQR controller for stable hovering. These methods provided initial solutions but revealed limitations in handling more complex failure scenarios, particularly in terms of stability and maneuverability. As a result, a unified controller was developed to efficiently handle both tasks, offering better fault tolerance, adaptability, and stability across various failure conditions, ensuring more robust performance in real-world situations.

For the current report, we have explored better solution as well as tried to improve upon the existing solutions. Our final solution is able to navigate and land with a single motor failure. On top of that we have tried to make our controller uniform as well as demonstrate its effectiveness for more than 1 rotor failure.

The following section contains a brief review of previously used approaches mentioned in our mid-evaluation report, which provide useful insights into our direction of thought towards the solution of the problem statement.

Fault Detection and Isolation (FDI) using Accelerometer

This fault detection approach identifies motor failures by analyzing linear acceleration changes (ΔA_x and ΔA_y) over a **three-sample** window. A sudden change exceeding a **threshold** marks the fault's timestamp. This method proved to be effective in simple scenarios of hover but failed during extreme maneuvers which caused false positives.

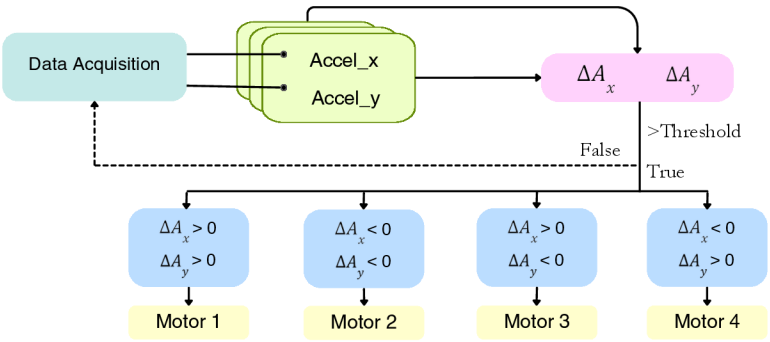


Fig.1 Diagram for motor failure detector

Landing Using PD Controller

A simple PD-based controller was used for emergency landing after motor failure, shutting off the opposite propeller to simplify dynamics and focus on thrust and pitch for stable attitude control. However, the controller was too simple and struggled with highly nonlinear scenarios[3].

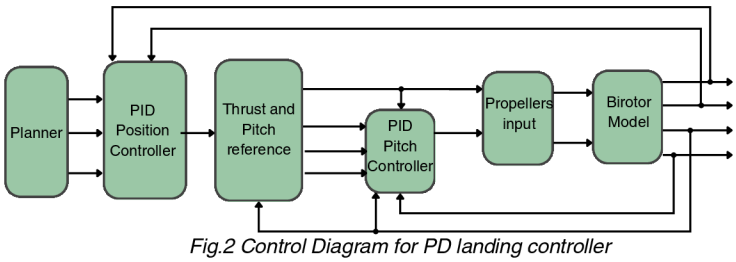


Fig.2 Control Diagram for PD landing controller

Hover using Linear Quadratic Regulator (LQR)

An LQR-based controller was used for hovering: an outer loop for desired attitudes and an inner loop ensuring attitude tracking for throttle inputs. This approach was **tuning-intensive** and heavily dependent on system parameters [4].

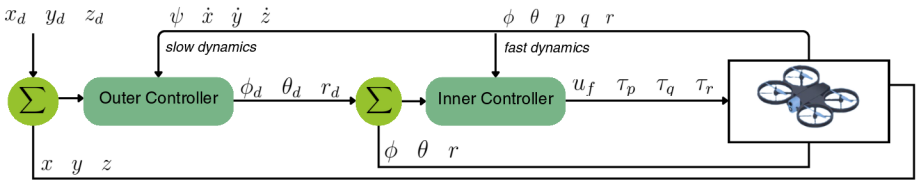


Fig.3 Control Diagram for Hover using LQR

Challenges during Mid Eval

Ultimately, the approaches that were presented during mid-evaluation were replaced with a better fault detection algorithm and a unified controller handling both tasks due to the following reasons:

- The approaches struggled with nonlinear scenarios & extreme maneuvers, leading to compromised system stability under dynamic conditions.
- The simplistic designs lacked the robustness needed for unpredictable environments, limiting their ability to perform reliably in real-world applications.
- Excessive reliance on precise system parameters and complex tuning revealed inefficiencies, underscoring the need for a more unified and practical solution.
- High false-positive rates in fault detection diminished system reliability, particularly during complex and critical operations.

Building Up

Feedback Linearization

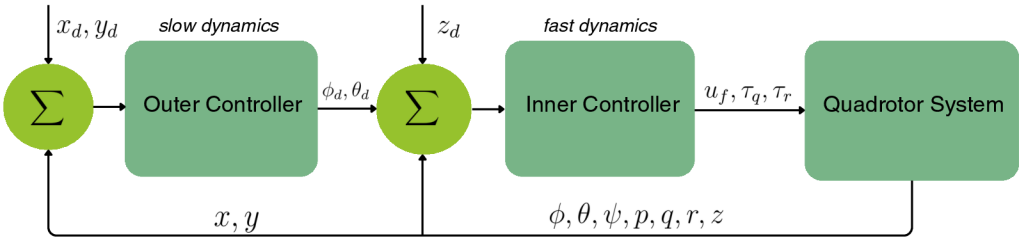


Fig.4 Control Diagram for Feedback Linearization

Feedback linearization linearizes nonlinear dynamics by using control inputs to cancel out nonlinearities, enabling linear control laws for complex systems. A **reduced state space** excluding yaw and yaw rates simplifies the dynamics. The inner loop handles high-frequency rotational dynamics (roll, pitch) via motor inputs, while the outer loop manages slower positional adjustments to control the quadrotor's trajectory.

Desired **waypoints** for landing and hovering are provided directly. For distant targets, waypoints are specified in 10-meter increments along the x and y axes, with the z-axis handled separately. The **decoupled** approach allows the quadrotor to first reach the x and y coordinates before adjusting altitude, simplifying control and enhancing stability during trajectory tracking [2].

Inner

$$\begin{bmatrix} u_f^* \\ \tau_q^* \\ \tau_r^* \end{bmatrix} \doteq -\left(J(\bar{x})h(\bar{x})\right)^{-1} J(\bar{x})f(\bar{x}) - \left(J(\bar{x})h(\bar{x})\right)^{-1} \left(2\xi_{in}c_{in} \begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_9 \end{bmatrix} + c_{in}^2 \begin{bmatrix} e_1 \\ e_2 \\ e_9 \end{bmatrix} - \begin{bmatrix} \ddot{x}_{1d} \\ \ddot{x}_{2d} \\ \ddot{x}_{9d} \end{bmatrix} \right)$$

$$\begin{bmatrix} x_{1d} \\ x_{2d} \end{bmatrix} = -\frac{m}{u_f} \begin{bmatrix} S_{x_3} & -C_{x_3} \\ C_{x_3} & S_{x_3} \end{bmatrix} \begin{bmatrix} -\frac{k_t}{m}x_{10} + 2\xi_o c_o e_{10} + c_o^2 e_7 - \dot{x}_{10d} \\ -\frac{k_t}{m}x_{11} + 2\xi_o c_o e_{11} + c_o^2 e_8 - \dot{x}_{11d} \end{bmatrix}$$

Outer

INDI

Incremental Nonlinear Dynamic Inversion (INDI) approximates quadrotor attitude control post-failure by **linearizing** dynamics. The controller consists of four modules which are as follows: [7]

Dynamics

$$\dot{\Omega}_b = I_b^{-1}\left[Q_b - \Omega_b \times (I_b \Omega_b)\right]$$

$$\dot{V}_{Kb} = \frac{1}{m} R_b + M_{bg}\left[0\ 0\ g\right]^T - \Omega_b \times V_{Kb}$$

- **Position Controller :** Computes virtual acceleration ν_s from position, velocity, and acceleration errors to determine desired thrust direction n_d for trajectory correction.
- **Outer Loop :** Computes total thrust ν_{aT} based on the desired thrust direction, which is low-pass filtered.
- **Inner Loop :** Aligns n_d using dynamics and a pseudo-control input combining NDI and feedback gains.
- **Control Allocator :** Maps virtual inputs to rotor throttles via the **Weighted Least Squares** (WLS) method.

Inner

$$v_{\dot{y}} \approx \ddot{y}_f + (\tilde{G}_1 + \tilde{G}_2)\Delta\mathbf{u} - \tilde{G}_2 z^{-1}\Delta\mathbf{u}$$

Inner

$$n_{d,\text{ndi}} := \frac{(v_s - g)}{\|v_s - g\|} \qquad v_{a_T} := \|a_T n_f + (v_s - \ddot{s}_f)\| \quad \text{with} \quad a_T = n_f \cdot (\ddot{s}_f - g)$$

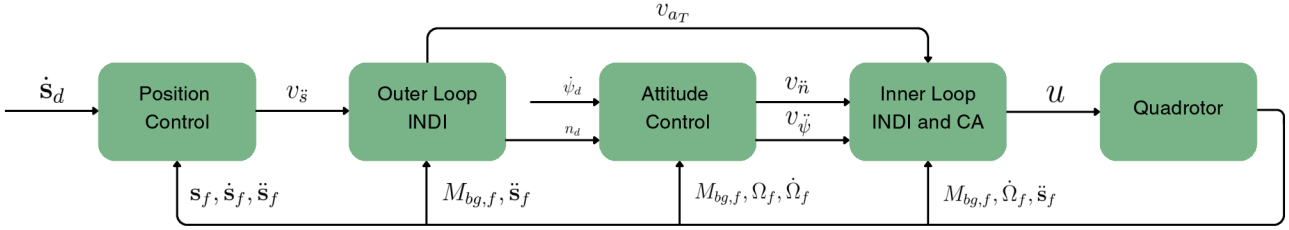


Fig.5 Control Diagram for INDI

Final Solution

Apparent Force Divergence Fault Detection

Irrespective of the states of the drone prior to the fault, the drastic reduction in thrust of the faulty motor, can be equivalently represented by an **external force** applied on the faulty motor, equal and opposite to its thrust. To **oppose** this apparent force on the motor, the controller would try to increase the thrust of the faulty motor and decrease the thrust of the other three motors. This behavior can be exploited for fault detection & identification.

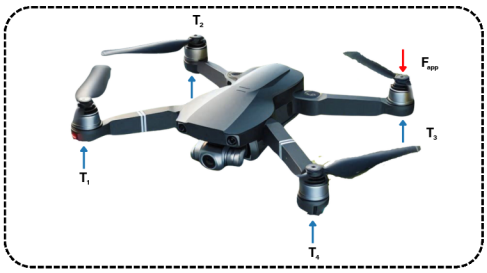
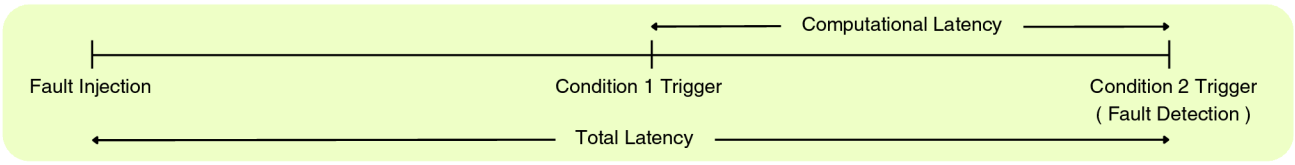


Fig. 6 Fault in Motor 3 seen by the Controller



Implementation

To summarize, two behaviours of the drone system at the time of motor failure has been exploited here:

- Control signals to the faulty motor rise sharply, while signals to the others drop significantly
- The anomaly in the control signals progressively compounds and intensifies over time

To observe the first behaviour, we consider the control signals given to each actuator relative to another. Consider ui to be the control signal given to the i^{th} motor (i = 1,2,3,4), then the rowsum of the i^{th} row of the matrix mentioned on the right, R_i, will give us the sum of the differences in the control signals of all other motors with respect to the i^{th} motor.

The quantity R_i is representative of the variability of the other motors with respect to the i^{th} motor. This quantity increases rapidly for the faulty motor and decreases to negative values for the remaining motors. This gives us an estimate of the motor fault.

Thus, condition A is checked to detect the motor fault.

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix}$$

$$\text{where } m_{ij} = u_i - u_j$$

$$R_i = \sum_{j=0}^{4} (u_i - u_j)$$

$$\Delta R_i = R_i(t) - R_i(t-1)$$

However, considering and comparing only this quantity poses a few challenges. **False positives** may arise in case of a slight imbalance in the centre of mass of the drone for motor fault detection on hardware. However sensor noise and extreme maneuvers may also produce false positives.

To solve this issue, we must consider the change in R_i with time (between consecutive timestamps) R_i In case of a true motor failure event, it is observed that R_i spikes just before condition A is satisfied.

$$\text{Condition A: } R_i \geq R_{\text{Threshold}} \quad \text{and} \quad R_j < 0, \forall j \neq i$$

$$\text{Condition B: } \Delta R_i \geq \Delta R_{\text{Threshold}} \quad \text{and} \quad \Delta R_j < 0, \forall j \neq i$$

$$\langle R_i \rangle(t) = \sum_{j=0}^{k-1} (R_i(t-j)) \quad \text{where } k \text{ - moving average window}$$

$$\Delta \langle R_i \rangle(t) = \langle R_i \rangle(t) - \langle R_i \rangle(t-1)$$

$$\textbf{Condition 1: } \Delta \langle R_i \rangle \geq \Delta \langle R \rangle_{\text{Threshold}}$$
$$\Delta \langle R_j \rangle < 0, \forall j \neq i$$

$$\textbf{Condition 2: } \langle R_i \rangle \geq \langle R \rangle_{\text{Threshold}}$$
$$\langle R_j \rangle < 0, \forall j \neq i$$

Hence, condition B can be checked at first, giving us an indication of a potential fault, and condition A can be checked within a few timestamps after condition B is met, confirming the initial suspicion and eliminating false positives to a substantial extent.

Noise can still interfere with this algorithm and produce undesired results. To eliminate noise, the moving averages of R_i and R_i are considered instead of the instantaneous values.

Multi Rotor Fault Tolerant Controller

Core Principles and Logic

Building on insights from previous approaches, the control system for the final approach comprises three parts: **Disturbance estimator**, desired virtual control generation, and commanded virtual **control feedback** loop. In the FTC case, the system is modelled with loss of control over r, and the passive controller treats motor failure(s) as a lumped disturbance.[6]

The fault-free system model is as described below, with control inputs consisting of the **net force** and **torques** about the body-frame x, y, and z axes. The primary axis, n_3, is defined as the direction of the body-frame z-axis projected in the inertial frame, expressed as R_{eb} [0,0,1]. Here, J represents the inertia matrix of the body about its center of mass (COM).

$$\begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\mathbf{v}} \\ \dot{\mathbf{n}}_3 \\ \dot{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{g} \\ -\mathbf{n}_3 \times \mathbf{R}_{eb}\boldsymbol{\omega} \\ \mathbf{J}^{-1}\left(-\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}\right) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{-1}{m}\mathbf{n}_3 & 0 \\ 0 & 0 \\ 0 & \mathbf{J}^{-1} \end{bmatrix} \begin{bmatrix} f \\ \boldsymbol{\tau} \end{bmatrix}$$

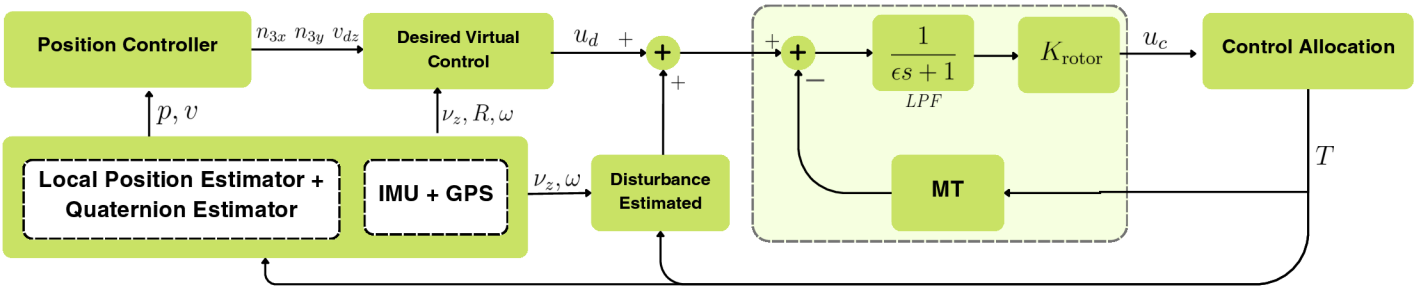


Fig.7 Control Diagram for MRFTC

The desired **waypoints** for landing and hovering are given directly. For distant targets, waypoints are specified in 10-meter increments along the x and y directions, with the z-axis considered separately. X-Y and Z traversals are decoupled: the quadrotor first reaches the target x and y coordinates, then adjusts altitude to the desired z position. This **decoupling** simplifies control and enhances stability during trajectory tracking.

Mathematical Model

The above defines the relation between the virtual control u and the thrusts T. The angles ψ_i are defined as the angles between the body x axis and each rotor. Λ = diag($\lambda_1, \lambda_2, \lambda_3, \lambda_4$) is the rotor health matrix; **efficiency coefficient** λ_i [0, 1]; λ_i = 0 indicates that the i^{th} rotor is completely faulty, λ_i = 1 represents that the i^{th} rotor is fault-free. We then write the virtual control in the following form.

Virtual Control and Thrust

$$\mathbf{u} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ -l_1 \sin\Psi_1 & -l_2 \sin\Psi_3 & -l_1 \sin\Psi_4 & -l_2 \sin\Psi_2 \\ l_1 \cos\Psi_1 & l_2 \cos\Psi_3 & l_1 \cos\Psi_4 & l_2 \cos\Psi_2 \\ c & -c & c & -c \end{bmatrix}}_{\mathbf{M}} \Lambda\mathbf{T}$$

$$\mathbf{d} = \mathbf{M}\left(\mathbf{I}_4 - \boldsymbol{\Lambda}\right)\mathbf{T} \qquad \mathbf{u}_C = \frac{K_{rotor}}{\epsilon s + 1}\left[\mathbf{u}_d - \mathbf{M}\mathbf{T} + \mathbf{d}\right]$$

We calculate the control command u_c in a feedback loop with the desired thrust T, followed by defining the control loops. The **outer position** controller uses a **PD-approach** to determine the desired primary axis and velocity in the z-direction.

Then based on the fault-tolerant model, we define the (estimated) disturbance as well as the control:

The disturbance is modelled as followed, from the above dynamics, and its **estimate** is taken by adding a low pass filter to it. The purpose of the low pass filter is to reduce the effect of noise in the calculated outputs.

Outer Loop

$$\mathbf{n}_{3,d} \triangleq \begin{bmatrix} \mathbf{x}_{1,d}^\top \\ n_{3,z,d} \end{bmatrix}^\top = -\frac{\mathbf{a}_d - \mathbf{g}}{\|\mathbf{a}_d - \mathbf{g}\|}$$

$$\text{sat}(\mathbf{u}, a) = \begin{cases} \mathbf{u}, & \|\mathbf{u}\| \le a \\ \frac{a}{\|\mathbf{u}\|}\mathbf{u}, & \|\mathbf{u}\| > a \end{cases}$$

$$\mathbf{a}_d = \text{sat}(\mathbf{K}_v(\mathbf{v}_d - \mathbf{v}), a)$$

$$\mathbf{v}_d = \begin{bmatrix} v_{x,d} & v_{y,d} & v_{z,d} \end{bmatrix}^\top = k_p(\mathbf{p}_d - \mathbf{p})$$

Inner Loop

Dynamics

$$\begin{cases} \dot{x}_1 = \mathbf{A}_1 x_2 \\ \dot{x}_2 = \mathbf{A}_2 x_2 + \mathbf{D}_2 + \mathbf{B}_2 u_1 \\ u_1 = \mathbf{E}_1 \mathbf{M} T - \mathbf{d}_1 \end{cases}$$

$$\tilde{\mathbf{x}}_1 = \mathbf{x}_1 - \mathbf{x}_{1,d}, \quad \tilde{\mathbf{x}}_2 = \mathbf{x}_2 - \mathbf{x}_{2,d},$$

$$\mathbf{u}_{d,1} = \mathbf{B}_2^{-1}\left(-\mathbf{K}_2\tilde{\mathbf{x}}_2 - \mathbf{A}_2\mathbf{x}_2 - \mathbf{D}_2 - \left(\mathbf{A}_1^T - \dot{\mathbf{K}}\right)\tilde{\mathbf{x}}_1 + \mathbf{G}_3\mathbf{x}_2\right)$$

$$\mathbf{d}_1 = \mathbf{E}_1\mathbf{M}\mathbf{T} - \mathbf{B}_2^{-1}\left(\dot{\mathbf{x}}_2 - \mathbf{A}_2\mathbf{x}_2 - \mathbf{D}_2\right)$$

Finally the command control value, which is in feedback with the desired/current thrusts is defined by the following relation.

The final equation uses a **Moore-Penrose inverse**, since $E_1 M$ is not a square matrix, and it is the only available control we have, as $u_{c,1}$ does not have calculated $\mathcal{T}_r$.

Uniform Passive Control

This method **passively** generates control commands for a drone, eliminating the need for a dedicated motor failure detection system. Since the disturbance is estimated directly within the rotor health matrix, and the fault-tolerant controller accounts for the full rotor dynamics, there is **no need** for an **explicit switch** to a separate controller during a fault. Instead, the controller operates seamlessly as a unified system under all conditions.

Multi rotor failure

This method is designed to generate **passive** control commands for a drone with a total mass of 1.3 kg. It ensures **stability** and operational control even in the event of up to two simultaneous motor failures. These failures can occur on opposite motors, highlighting the system's robust design.

Metrics

$$\mathbf{R}_{\text{eb}} \triangleq \begin{bmatrix} A & C & n_{3,x} \\ B & D & n_{3,y} \\ E & F & n_{3,z} \end{bmatrix} \triangleq \begin{bmatrix} \mathbf{n}_1 & \mathbf{n}_2 & \mathbf{n}_3 \end{bmatrix} \qquad \mathbf{A}_1 = \begin{bmatrix} 0 & -\mathbf{C} & \mathbf{A} \\ 0 & -\mathbf{D} & \mathbf{B} \end{bmatrix} \qquad \mathbf{A}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\frac{J_z - J_x}{J_y} r \\ 0 & -\frac{J_x - J_z}{J_y} r & 0 \end{bmatrix}$$

$$\mathbf{B}_2 = \begin{bmatrix} -\frac{n_{3,z}}{m} & 0 & 0 \\ 0 & \frac{1}{J_x} & 0 \\ 0 & 0 & \frac{1}{J_y} \end{bmatrix} \qquad \mathbf{D}_2 = \begin{bmatrix} g \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{G}_3 = \begin{bmatrix} 0 & 0 \\ 0 & G \end{bmatrix}, \quad \mathbf{G} = -n_{3,z} k_{n_3} \mathbf{I}_2$$

$$\mathbf{K} = k_{n_3} \begin{bmatrix} 0 & 0 \\ -B & A \\ -D & C \end{bmatrix}, \quad \mathbf{K}_2 = \begin{bmatrix} k_{v,z} & 0 & 0 \\ 0 & k_\omega & 0 \\ 0 & 0 & k_\omega \end{bmatrix}, \quad \dot{\mathbf{K}} = k_{n_3} \begin{bmatrix} 0 & 0 \\ -\dot{B} & \dot{A} \\ -\dot{D} & \dot{C} \end{bmatrix} \qquad where \;\; d_1 = E_1 d, \mathbf{E}_1 = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3\times 1} \end{bmatrix}, and$$

System Setup

Hardware Setup

S500

The S500 drone was chosen for its alignment with IRIS drone specifications, affordability, availability, and easy access to components. Here's why it was ideal for our study:

- **Alignment with IRIS Specifications:** The S500 closely matches the IRIS drone in size and weight, ensuring simulation results apply effectively to real-world implementation.
- **Flight Controller Compatibility:** Equipped with the Pixhawk 6C, the S500 ensures industry-standard reliability for both simulation and deployment.
- **Cost Considerations:** The S500 offers a balanced, cost-effective option, combining affordability with robust features and capabilities.
- **Ease of Availability:** The S500 is readily available, with strong technical support and spare parts access, ensuring smooth operation during the competition.

Parameters	Challenge	Solution
Multiple Batteries	Battery shortages delayed test flights	Multiple batteries purchased
Proper Testing Rig	6 DOF Rig Testing	Setup developed using ropes for testing
Open Testing Space	A wide area required	Setup built outdoors
Crash Protection	Cushioning and Net Required	Mattresses and Net used

Outdoor Hardware Testing Setup

1. Setup for Yawing Motion

- To test yaw stability, a BLDC motor was used for free rotation around the attachment point between the rope and the drone. This setup allowed for controlled yaw motion during testing.

2. Setup for Fault Injection

- A relay switch was placed between the motor and the ESC, controlled via the transmitter. This allowed for fault injection at desired intervals, simulating real-world failures during flight.

3. Testing Rig Setup

- A 15x20 ft bamboo pole structure was constructed to create a sturdy testing frame. This setup provided a stable platform for testing the drone under controlled conditions with ample space to test fault control.

Software Setup

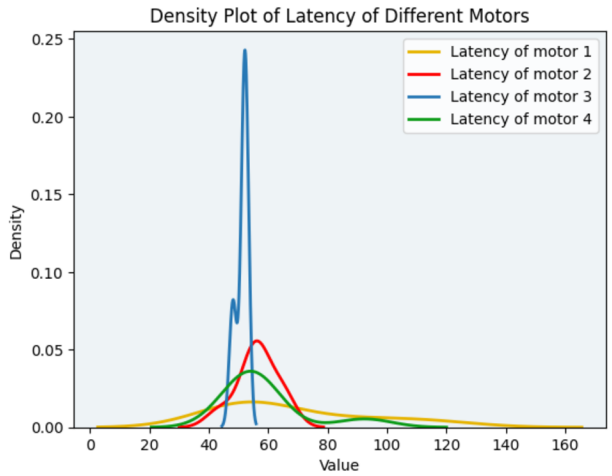
We work with PX4-Autopilot v1.15.1 on an Ubuntu 20.04 platform, customizing simulation & control modules:

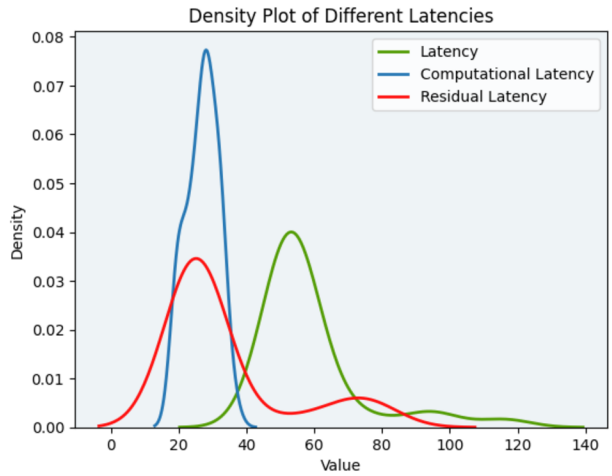
1. **Simulation Environment:**
- Built `sitl_gazebo-classic` for access to Gazebo plugins, models, and worlds.
- Modified motor failure and motor model plugins to enable failure injections.
2. **Custom Messaging:**
- Developed new message files for motor failure detection & waypoint transmission to a custom controller.
3. **Module and Configuration Enhancements:**
- Added new modules: `motor_failure_detector` and a modified `control_allocator.`
- Updated `flight_mode_manager` to work with the custom changes.
- Modified board configuration files and startup scripts ensure proper startup of `motor_failure_detector.`
4. **Web Interface:**
- Developed a real-time console for monitoring telemetry data and sending navigation commands to the running PX4 instance via the MAVLink protocol.

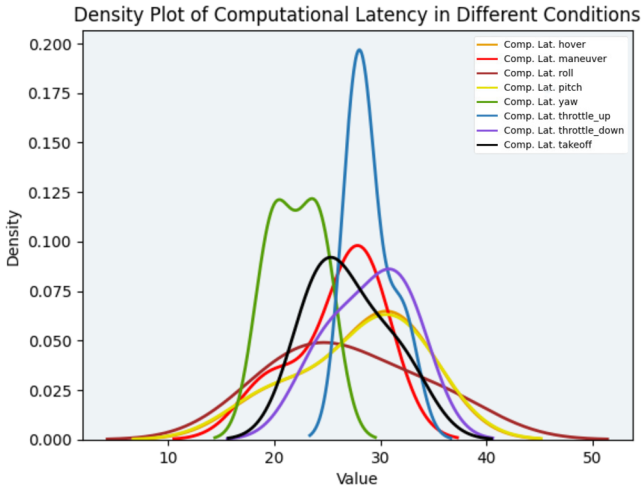
Results

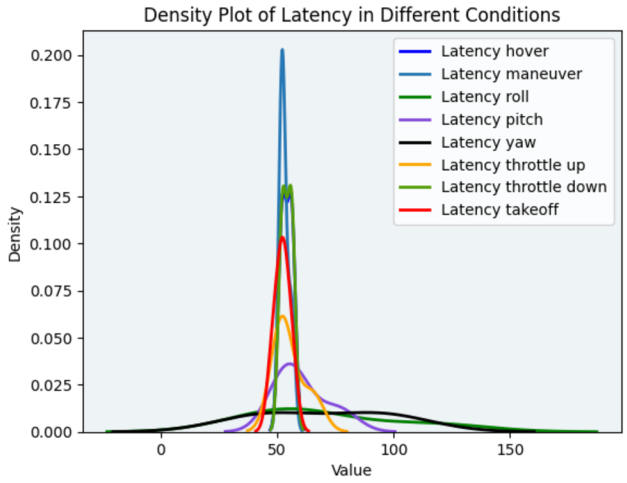
Fault Detection

Latency







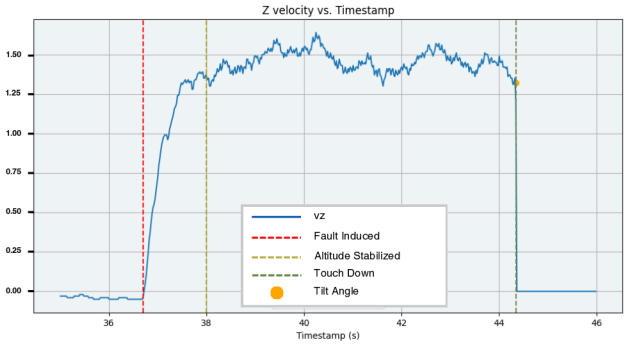


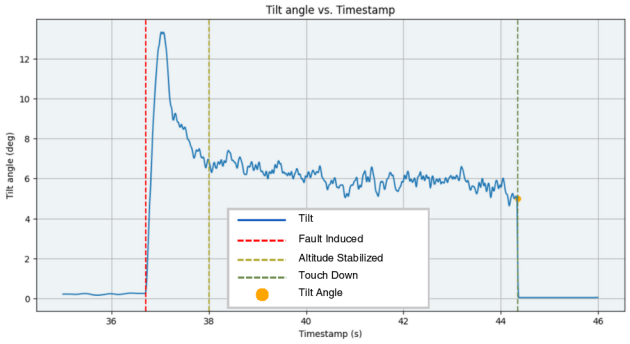
Condition	Latency	F-Score
Hover	54.204	10.895
Maneuver	53.196	14.345
Roll	70.211	0.056
Pitch	60.358	0.480
Yaw	70.111	0.069
Throttle Up	55.408	1.315
Throttle Down	54.256	11.406
Take Off	52.219	5.299

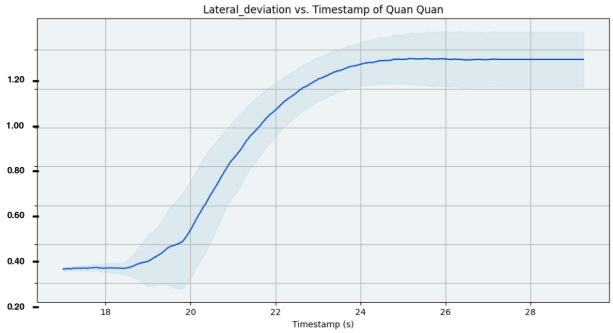
- Total 5 tests were performed for each motor for each conditions leading to total of 160 tests; plotted motor specific and condition specific latency plots
- In hover, maneuver, and throttle-down conditions, detection **latency** is **consistent** and relatively low because there is a gradual change in states
- In conditions like roll and yaw, motors experience significant **residual latencies** due to rapid angular movements
- Throttle up and takeoff operations represent transitional phases, introducing **dynamic variability** that is harder to model
- However, there is relatively less latency between flag1 and flag2 for yaw because the rapidly changing states help trigger the flag2 faster

Landing

MRFTC

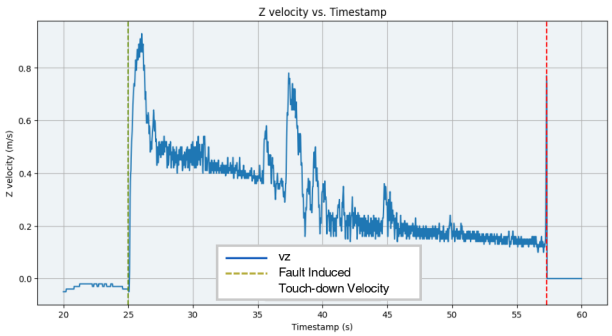


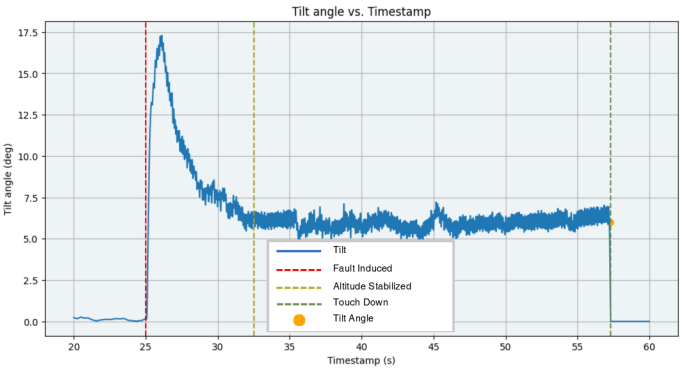


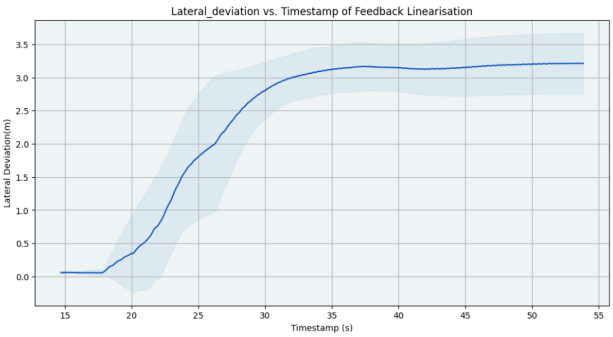


Touch down speed = 1.32 m/s (9 cm drop above ground)
Touch down tilt = 5 degrees
Attitude stabilization = 1.3 s
Lateral deviation = 1.5m

Feedback

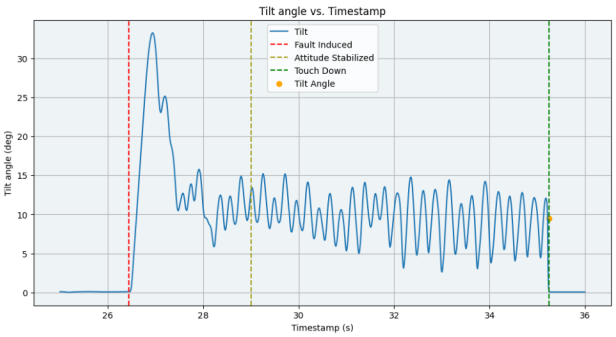


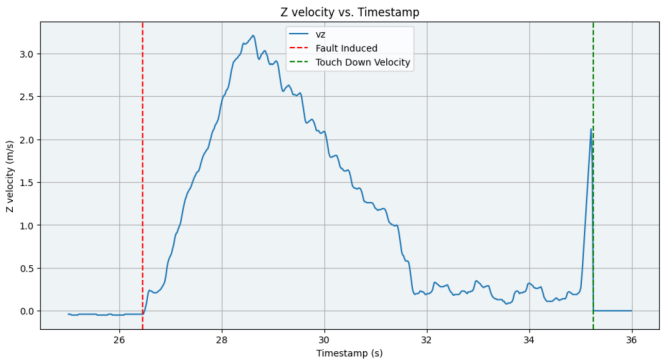


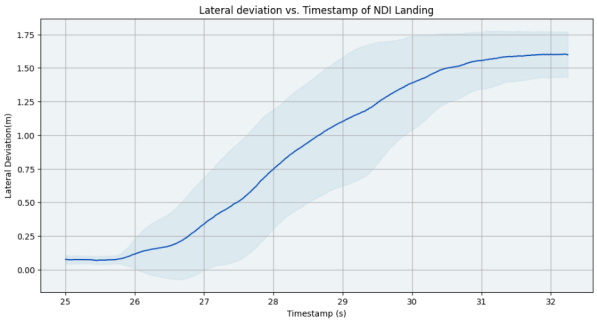


Touch down speed = 0.4 m/s (1 cm drop above ground)
Touch down tilt = 6 degrees
Attitude stabilization = 7 s
Lateral deviation = 4 m

INDI





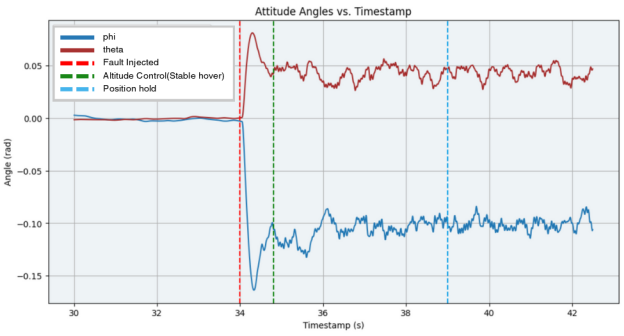


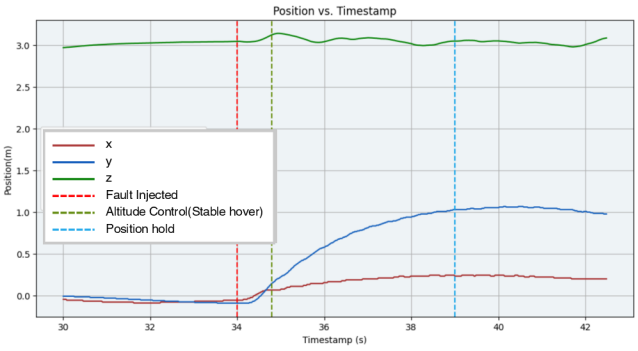
Touch down speed = 2.13 m/s (23 cm drop above ground)
Touch down tilt = 9 degrees
Attitude stabilization = 2.46 s
Lateral deviation = 1.6 m

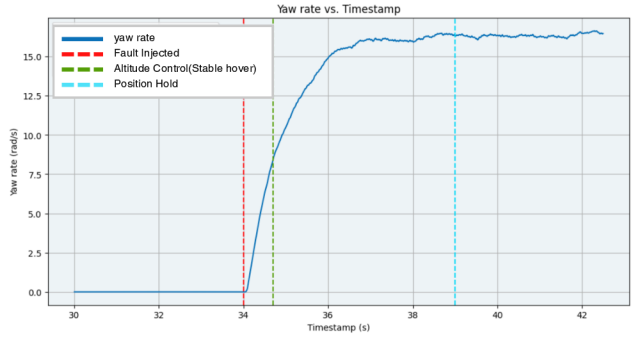
In terms of the touchdown speed, **Feedback Linearisation** is able to dominate the other controllers. However, **MRFTC** performs better than others in the other metrics for landing. However, with the given analogies of free-falling objects, none of the strategies compromises the safety of the drone.

Hover

MRFTC

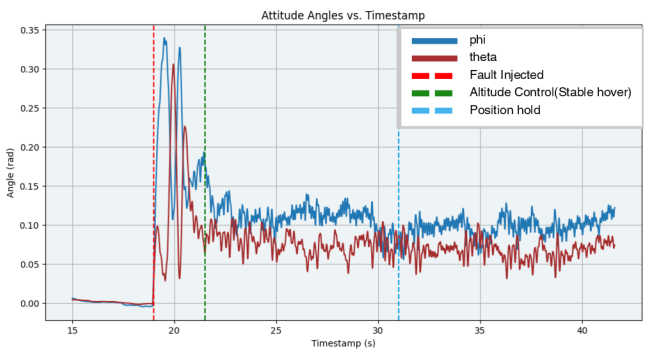


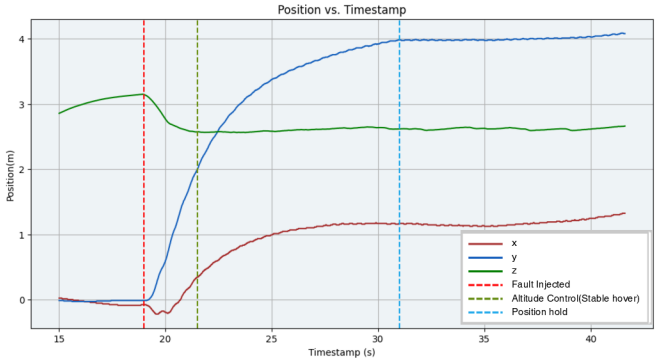


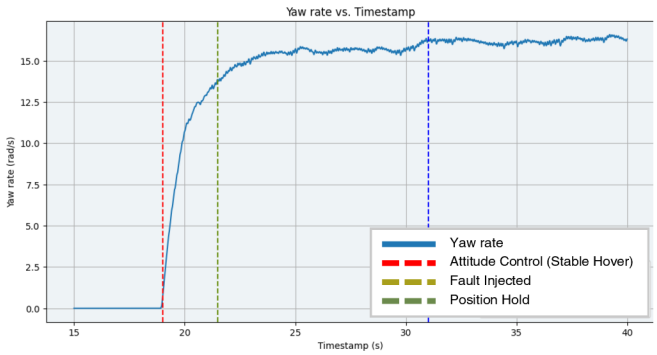


Attitude stabilization = 0.75 s
Z offset = 0.1 m (upwards)
Lateral offset = 0.13 m
Position hold time = 4 s
Position hold deviation = 1.3 m

Feedback

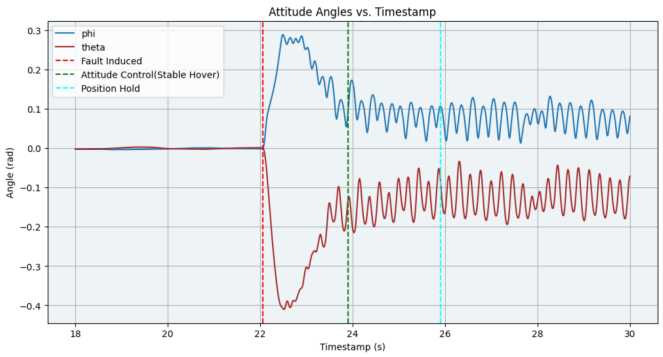


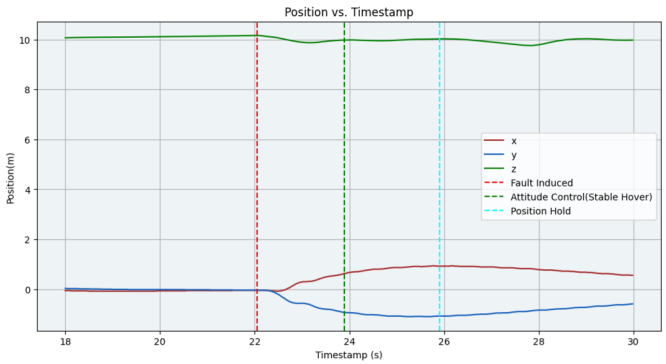


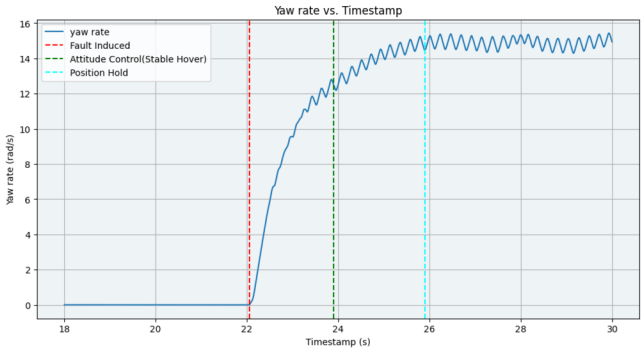


Attitude stabilization = 1.5 s
Z offset = 1 m (downwards)
Lateral offset = 1 m
Position hold time = 9 s
Position hold deviation = 4.2 m

INDI





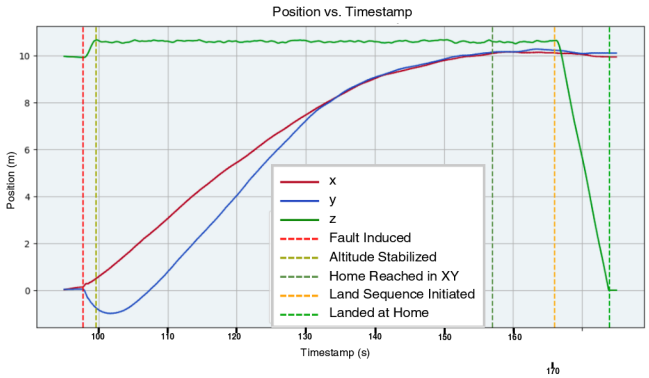


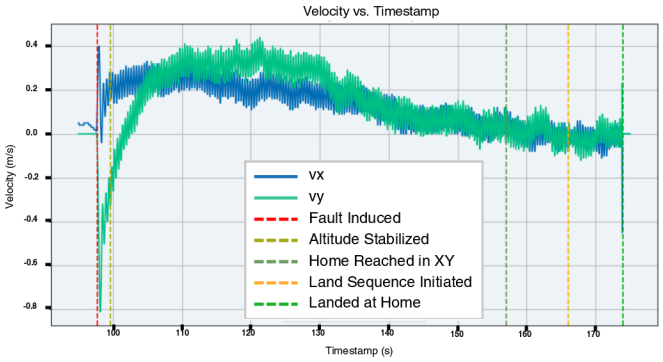
Attitude stabilization = 1.86 s
Z offset = 0.1 m (upwards)
Lateral offset = 0.35m
Position hold time = 1.98 s
Position hold deviation = 1.35 m

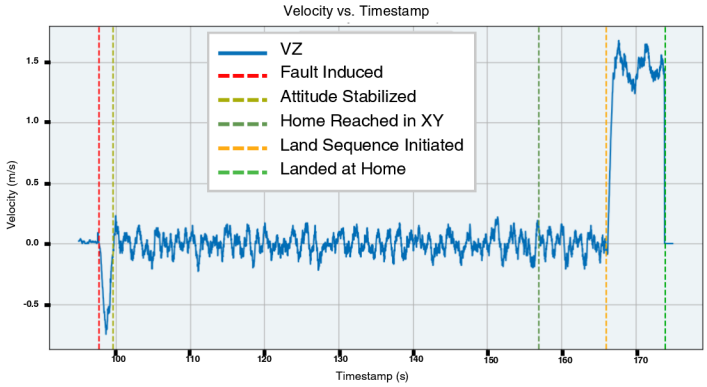
MRFTC has the best attitude stabilization response, however **INDI** is able to achieve position hold in a short time, with a slightly worse position hold distance. All of the strategies achieve hover with ease

Return to home (RTH)

MRFTC

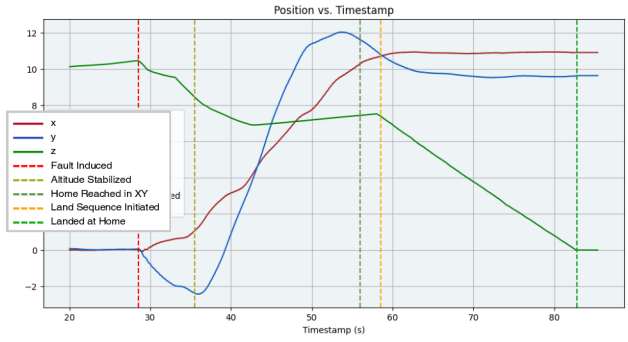


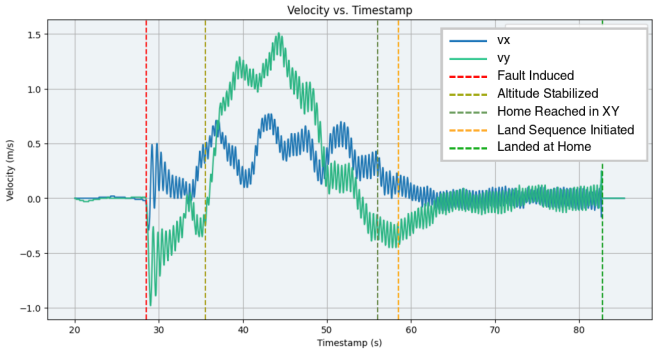


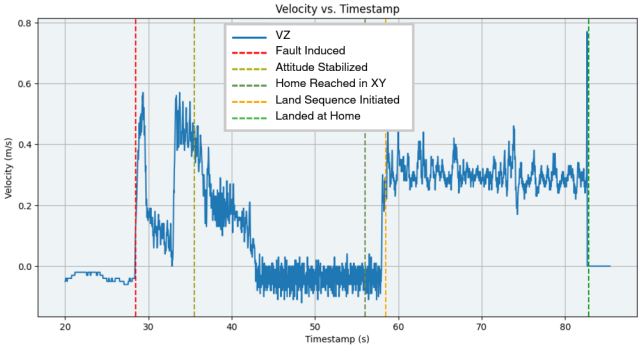


RTH time = 60 s
Stable height = 10.5 m (after 2 s)
Lateral speed = 0.25 m/s (stable)

Feedback

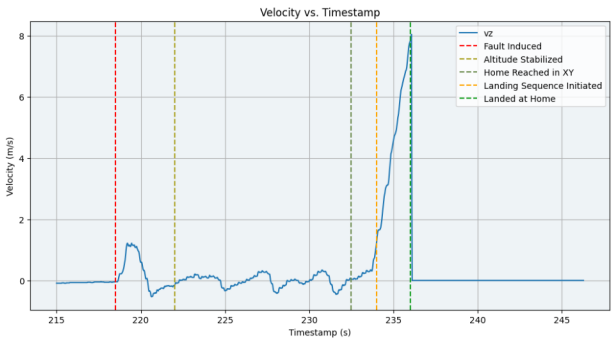


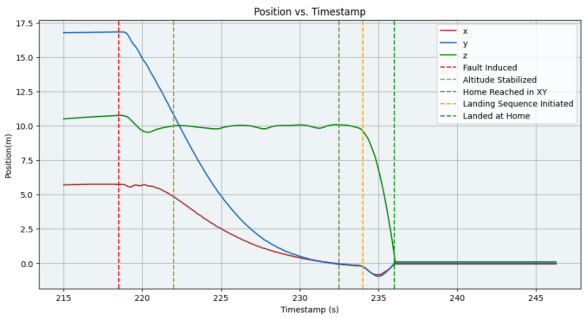


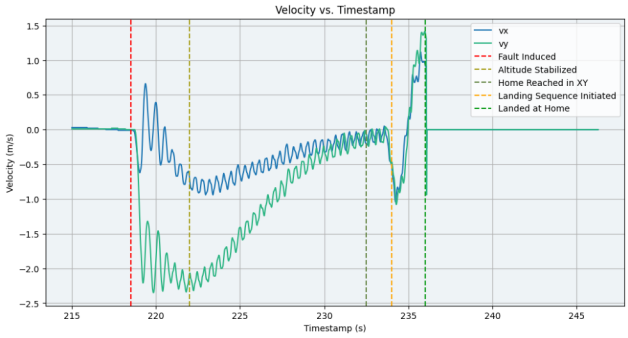


RTH time = 30 s
Stable height = 7 m (after 13 s)
Lateral speed = 1.1 m/s (varying)

INDI



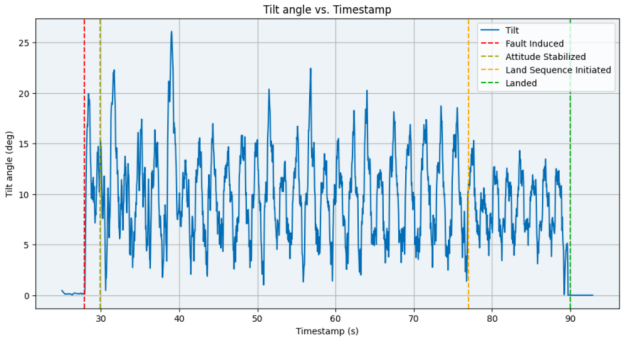


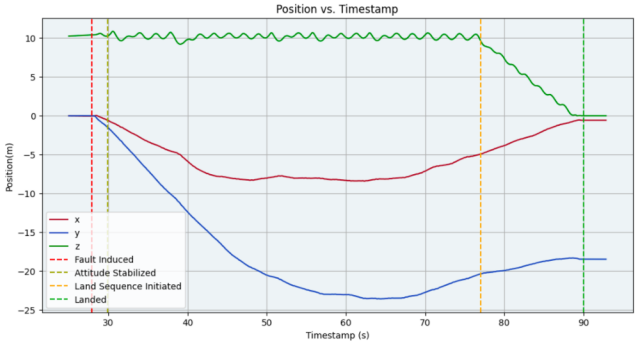


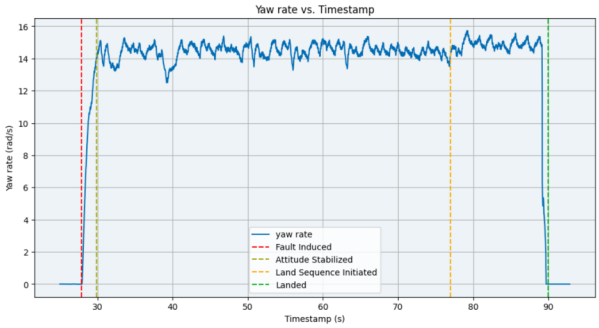
RTH time = 18 s
Stable height = 10 m (after 4s)
Lateral speed = 2.3 m/s (varying)

INDI reaches its goal the fastest, while **MRFTC** is able to achieve stabilization the fastest. It is also seen that throughout the path, **INDI** and **MRFTC** have minimal deviation in height and z velocity

RTH + Estimator





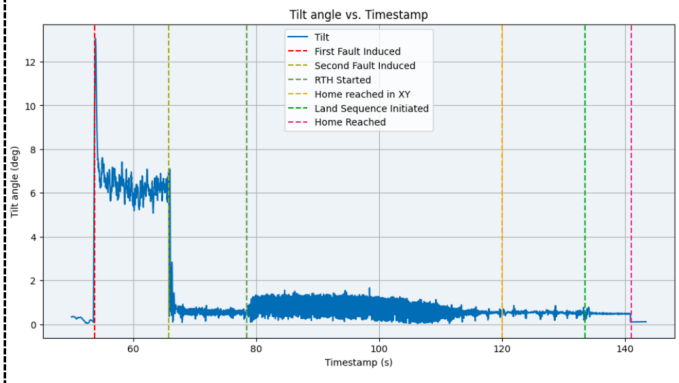


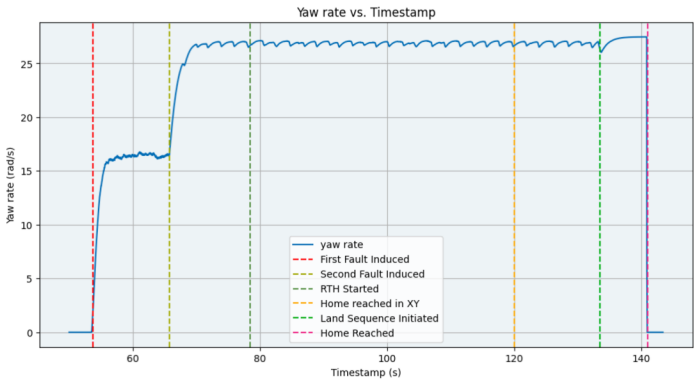
Attitude Stabiliization: 2 s
Land Time: 13 s
Yaw rate mean: 14.5 rad/s
Yaw rate deviation: 0.5 rad/s
Tilt angle mean: 10 degrees
Tilt angle deviation: 4.5 degrees

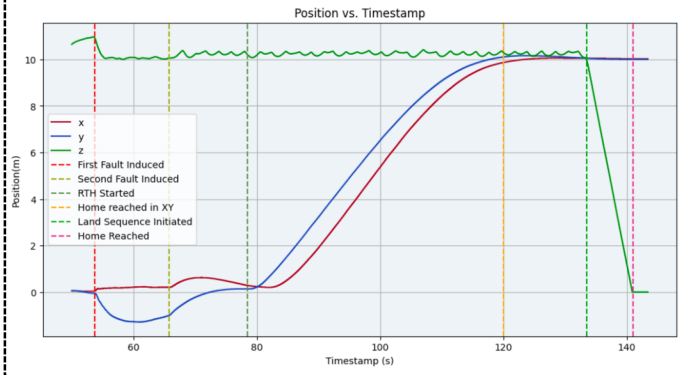
The controller stabilizes the yaw rate and height with minimal deviation using data from the estimator. However, an increase in tilt angle occurs due to a noisy attitude channel. The system demonstrates Lyapunov stability, as evidenced by the drone's lateral position being confined within a bound around its initial position, rather than exhibiting asymptotic stability.

Double Motor failure in MRFTC controller

MRFTC - RTH







Tilt after failure 1: 5 degrees
Tilt after failure 2: 0 degrees
Yaw rate after failure 1: 17 rad/s
Yaw rate after failure 2: 27 rad/s
Slight increase in height deviation

- Quan Quan can maintain full control and stability even with two opposing motor failures by directly incorporating the faulty motors into its dynamic model
- We induce a second motor failure after the drone stabilizes from an initial single-motor failure, demonstrating the controller's ability to maintain stability under double-motor failure conditions
- The drone's tilt angle reduces from an initial 5 degrees to 0 degrees during hover, indicating stability. While a single motor failure results in a non-zero tilt due to unbalanced torques, an opposite double-motor failure scenario can balance the torques, leading to a zero or near-zero tilt angle. We also note a sudden increase in the yaw rate for the same reason - more net torque in the same direction
- Quan Quan maintains a stable height with minimal vertical velocity, though the variance is higher compared to the single-motor failure case.
- The drone maintains stability even during lateral movement. A slight increase in tilt angle is observed, which is essential for aligning the drone's orientation with the desired direction of movement.

Conclusion

Challenges

Solutions

Challenges	Solutions
The **asymmetric** geometry of IRIS complicates modelling, frame transformations and control formulations	Accurate **mathematical** models developed for modelling thrusts and torques of the quadrotor
High yaw rate post failure causes **centrifugal** forces on onboard sensors leading to inaccurate state estimation	PX4's **default estimator** was tuned to reduce noise in the estimates. In addition, GPS and Q-attitude estimator was used to provide accurate altitude and attitude estimates
Accurate **estimation** of quadrotor parameters such as Moment of Inertia for actual hardware. The latency and noise coming from actual sensors need to be modelled	Model parameters estimated using approximate methods and experimentation on indigenous setup. Latency and noise modelled using post failure data from **log files** of actual hardware
Most fault detection methods mentioned in literature were computationally demanding and **slower response times** making them unsuitable for the task	Innovative low-cost and **low latency** fault detection methods were devised through meticulous data analysis, enabling accurate failure prediction even in extreme environments
Most control algorithms mentioned in literature developed separate algorithms for pre-failure and post-failure cases requiring explicit runtime switching of control algorithms based on failure detection	A uniform passive fault tolerant controller was developed with **augmented states** dependent on motor health allowing the control algorithm to switch passively to failure mode as soon as failure is detected

Future Scope and Recommendations

The **Extended Kalman Filter (EKF)** showed **high-frequency variations** in its state estimates that did not match the ground truth, likely caused by noise being amplified during the filter's updates. These variations can reduce the accuracy of state estimation, affecting tasks like control and navigation. To fix this, low-pass filters can be used to smooth the estimates by removing unnecessary high-frequency noise while keeping the important data. Careful tuning of these filters is needed to ensure they remove noise effectively without slowing down the system's response to changes.

Wind disturbances were not considered in the initial analysis, limiting the understanding of their effects on the quadrotor. Simulations showed that the quadrotor remained stable with **slight drifting** under low-magnitude winds. However, as wind strength increased, the control algorithm became unstable. This instability occurs because stronger winds create forces that the control system struggles to compensate for. These results highlight the need for more robust control strategies to handle wind disturbances and ensure stable flight in challenging conditions. Future development should focus on improving wind disturbance handling in both simulations and real-world tests.

A **user interface** has also been developed to enable users to input all control commands seamlessly, providing an intuitive and efficient way to interact with the system. This interface functions as an abstraction layer, allowing for **easy integration** with existing quadrotor systems. As a result, the solution can be deployed as a software update, making it possible to enhance the functionality of current systems without the need for significant hardware modifications.

Appendix

Hardware Testing

A 15x20 ft bamboo pole structure was constructed to serve as a stable testing frame for the drone. This rig provided a secure platform for conducting tests under controlled conditions, particularly useful in windy or unstable environments. The setup allowed us to test various controllers and flight systems while keeping the drone tethered and protected. To prevent the drone from falling directly to the ground during testing, a rope was used to secure it to the rig. This rope not only ensured that the drone remained attached but also allowed us to manually adjust the tension during takeoff to avoid slack and maintain stability. Additionally, the rig enabled us to fine-tune the drone's performance, calibrate control inputs, and optimize flight behavior, all without the risk of crashing. To further ensure safety, mattresses were placed around the testing area to cushion any hard landings and protect the drone from potential damage. The combination of the stable rig, manual rope tensioning, and crash protection allowed for safe and efficient testing, ensuring that adjustments could be made and reliable data gathered without compromising the integrity of the drone.

Fig. 8 Setup for outdoor testing

Hardware Setup

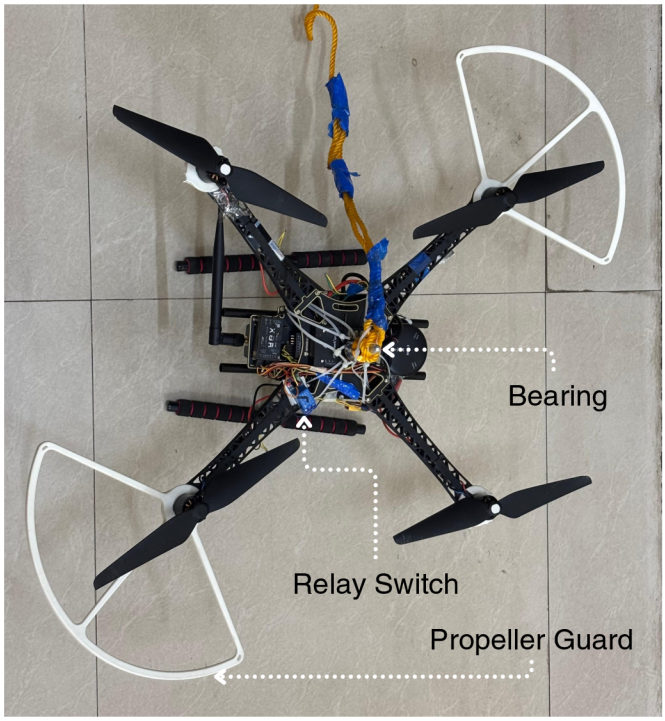


Fig. 9 Holybro S500 drone

To allow yawing motion, a BLDC motor was used as a bearing for free rotation around the attachment point between the rope and the drone.

A relay switch was placed between the motor and the ESC, controlled via the transmitter. The relay acted as an interrupter in the PWM signal transmission, enabling us to simulate a failure by cutting the signal and effectively disabling the motor. Powered directly through the receiver, the relay allowed for precise control over when the fault was triggered. This setup provided the flexibility to inject faults at specific intervals during flight, giving us an effective way to test the drone's response to motor failure and recovery mechanisms.

Additionally, propeller guards were installed to prevent damage to the propellers and enhance the safety of both the drone and the people around it. These guards helped protect the propellers from chipping during tests, ensuring that the drone remained in optimal working condition and reducing the risk of injury or damage during flight operations.

Appendix

IRIS Parameters

Parameters	Value
Mass	1.5 kg (1.535 considering total)
Ixx	0.029125 kg-m^2
Iyy	0.029125 kg-m^2
Izz	0.055225 kg-m^2
L1	0.2 m
L2	0.22 m
Motor Constant	$5.84*10^{-6}$ kg-m
Moment Constant	0.06
Max Rotor Angular Velocity	1100 rpm

S500 Parameters

Parameters	Value
Mass	1.5 kg
Ixx	0.04 kg-m^2
Iyy	0.04 kg-m^2
Izz	0.024 kg-m^2
Arm Length	0.29 m
Motor Constant	$6.76*10^{-6}$ kg-m
Moment Constant	0.07
Maximum Angular Velocity	2500 rpm

Appendix

Motor Fault Detection

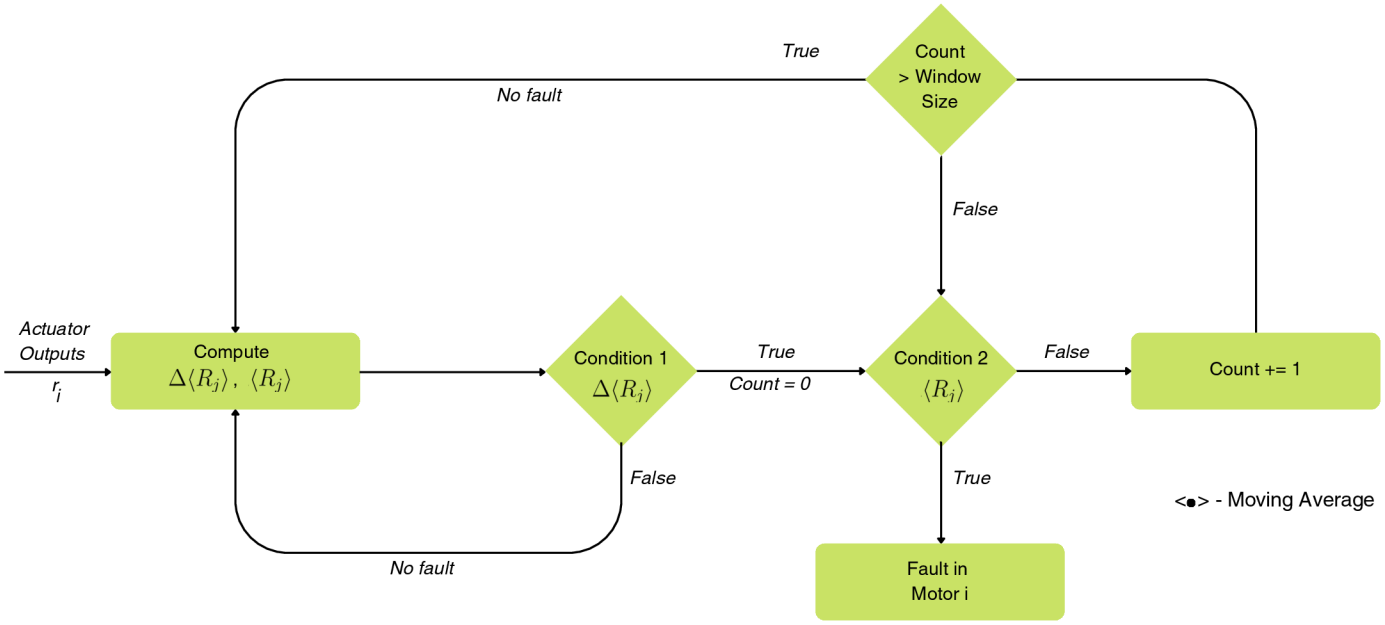
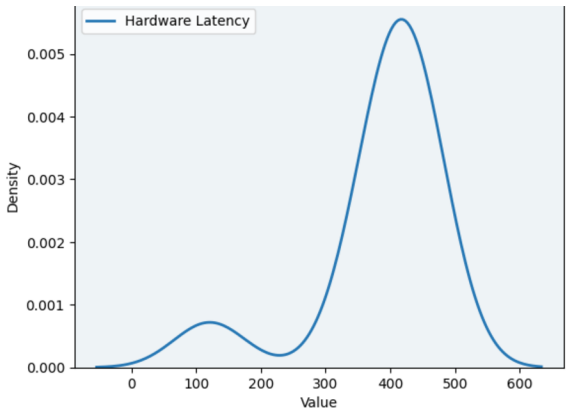


Fig.10 Flowchart for Motor Fault Detection Algorithm (AFDFD)

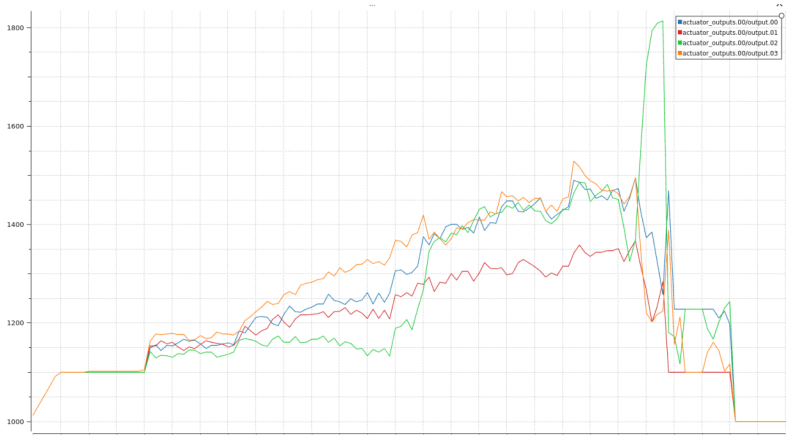
Motor with Fault	Latency (sec)	Angular Error during Latency (deg)		
		Roll	Pitch	Yaw
1	0.41863	7.1560709	6.4346025	5.1371968
2	0.370917	8.5753447	9.9259208	6.855554
3	0.135748	1.2527149	0.9487608	1.2527149
4	0.445711	1.7053515	1.9341336	1.5033839

Appendix

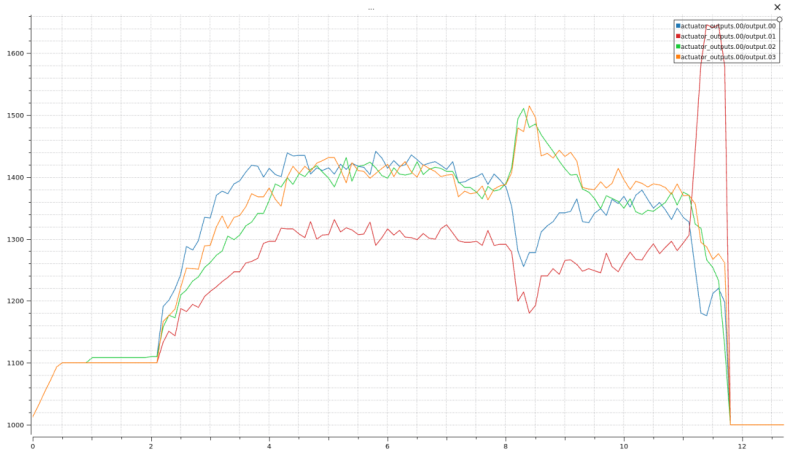
Fault Detection Results in Hardware



Distribution of Motor Fault Detection Latency on hardware



Motor number is detected correctly in case of throttle up which is evident by increasing motor outputs.



Motor number is detected correctly in case of yaw movement which is evident by increasing output of two rotors and decreasing for other two

System Setup

Moment of Inertia Approximation

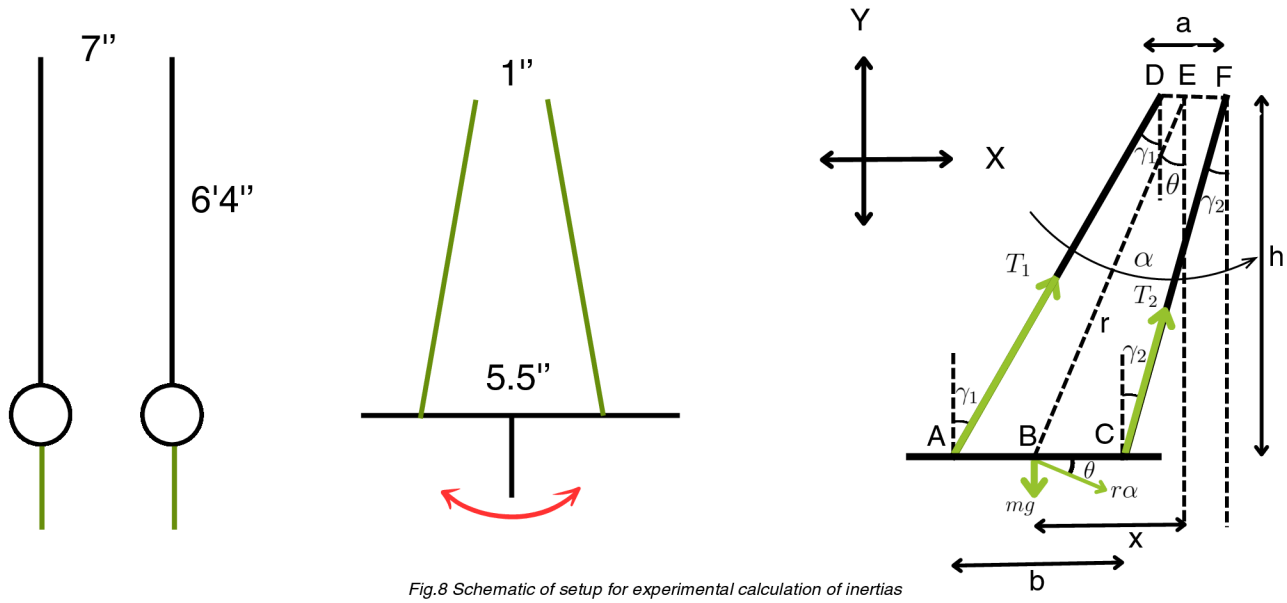


Fig.8 Schematic of setup for experimental calculation of inertias

For hardware testing, we needed to determine the moment of inertia of the S500 V2 drone as these parameters were necessary for controller tuning in SITL before applying them in hardware. Given above is the experimental setup that was used. The drone was suspended by 4 strings attached to its stands. It was then allowed to execute small oscillations about its mean position and the time period of oscillations were noted.

In order to calculate of moment of inertia from time period, a mathematical model was developed.

Force Equation along X - direction : $T_1 \sin\gamma_1 + T_2 \sin\gamma_2 = m(r\alpha)\cos\Theta$

Force Equation along Y - direction : $mg - T_1\cos\gamma_1 - T_2\cos\gamma_2 = m(r\alpha)\sin\Theta$

Force Equation about F : $(T_1\sin\gamma_1)h + mg\left(x+\frac{a}{2}\right) - (T_1\cos\gamma_1)\left(\frac{a}{2}+x+\frac{b}{2}\right) = I\alpha$

For I_{xx} if we have : $a = 1'',\ b = 5.5'',\ h = 72'',\ m = 1.5kg,\ g = 9.81\text{ms}^{-1}$

Substituting these values into the equations of the model and solving for α and θ we get,

$$\alpha = \frac{-7.5141}{0.2286I + 0.694}\quad \Theta$$

This is similar to the equation of SHM

Angular rate of oscillation :

$$\omega^2 = \frac{7.5141}{0.2286I + 0.694}$$

$$\implies I = \frac{32.8701}{\omega^2} - 3.0359$$

here, $I = I_{\text{CM}} + mr^2$

Using the above expression we calculated the values of moment of Inertia experimentally.

Small angle approximation :

$$\sin\gamma_1 = \tan\gamma_1 = \gamma_1 = \frac{x - \frac{a}{2} + \frac{b}{2}}{h}\ ,\quad \sin\gamma_2 = \tan\gamma_2 = \gamma_2 = \frac{x + \frac{a}{2} - \frac{b}{2}}{h}$$

$$\sin\Theta = \tan\Theta = \Theta = \frac{x}{h} \qquad and \qquad r = \frac{h}{\cos\Theta} = h$$

References

[1] Adaptive Three-Stage Kalman Filter for Robust Fault Estimation

Zhong, Y., Zhang, Y., Zhang, W., Zuo, J. and Zhan, H., 2018. Robust actuator fault detection and diagnosis for a quadrotor UAV with external disturbances. IEEE Access, 6, pp.48169-48180.

[2] Feedback linearization

Lanzon, A., Freddi, A. and Longhi, S., 2014. Flight control of a quadrotor vehicle subsequent to a rotor failure. Journal of Guidance, Control, and Dynamics, 37(2), pp.580-591.

[3] PD Landing

Lippiello, V., Ruggiero, F. and Serra, D., 2014, October. Emergency landing for a quadrotor in case of a propeller failure: A PID based approach. In 2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014) (pp. 1-7). IEEE.

[4] LQR Hover

Mueller, M.W. and D'Andrea, R., 2014, May. Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. In 2014 IEEE international conference on robotics and automation (ICRA) (pp. 45-52). IEEE.

[5] Non-Linear Dynamic Inversion

Ahmadi, K., Asadi, D., Nabavi-Chashmi, S.Y. and Tutsoy, O., 2023. Modified adaptive discrete-time incremental nonlinear dynamic inversion control for quad-rotors in the presence of motor faults. Mechanical Systems and Signal Processing, 188, p.109989.

[6] Multi Rotor Fault Tolerant Controller

Ke, C., Cai, K.Y. and Quan, Q., 2023. Uniform passive fault-tolerant control of a quadcopter with one, two, or three rotor failure. IEEE Transactions on Robotics.

[7] Incremental Non-Linear Dynamic Inversion

Beyer, Y., Steen, M. and Hecker, P., 2023. Incremental passive fault-tolerant control for quadrotors subjected to complete rotor failures. *Journal of Guidance, Control, and Dynamics*, *46*(10), pp.2033-2042.