Online Rotor Fault Detection and Isolation for Vertical Takeoff and Landing Vehicles

Jiaqi Lian, Neeraj Gandhi, Yifan Wang, and Linh Thi Xuan Phan

***Abstract*— Vertical take-off and landing (VTOL) vehicles are becoming increasingly popular for real-world transport; but, as with any vehicle, guaranteeing safety is both extremely critical and highly challenging due to issues like rotor faults. Existing fault detection and isolation (FDI) techniques usually focus on multirotor systems or fixed wing systems, rather than the hybrid VTOLs. Since VTOLs have *both* rotors and ailerons, a fault in a rotor may be masked by the (correctly working) ailerons, making it much more difficult to detect faults. However, this masking *only* works when ailersons are used (e.g., during cruising), leaving the takeoff and landing vulnerable to crashes.**

This paper presents an online rotor fault detection and isolation (FDI) method for VTOLs. The approach uses pose analysis and aileron command data to quickly and accurately identify the faulty rotor and to compute the severity of the fault. Our method works for hard-to-detect fault scenarios, such as small-severity faults that are masked during cruise flight but not during vertical motion. We evaluated our technique in a SITL PX4 simulation of a modified Deltaquad QuadPlane. The results show that our FDI technique can quickly detect and isolate faults in real time (within 1s-2.5s) and achieve high isolation success rate (91.67%) across six rotors, and that it can estimate the severity of faults to within 2%. When applying a simple recovery process post-isolation, the system consistently achieved safe landing.

I. Introduction

Aerospace ventures have traditionally focused on either fixed wing aircraft or multirotor systems; however, recent efforts by companies such as Joby, Archer, and Lilium are exploring vertical takeoff and landing vehicles (VTOLs) that use both fixed wings and multiple rotors. Such hybrid systems offer two key advantages: (i) unlike pure-fixed wing aircraft, which need large runways or slingshots to become airborn, VTOLs can use their multiple rotors to move vertically; and (ii) systems that rely solely on rotors to fly suffer from low range (because of the high energy cost inherent in relying entirely on propellers for lift), whereas VTOLs can leverage their fixed wings to operate for long periods of time (while still being battery-powered). For example, using a VTOL, one can cut down a fifty-minute drive in the Munich Metropolitan Region or San Francisco Bay Area by 3%-13% in 'normal' traffic and by *more than half* in heavy traffic [1].

As in pure multi-rotor systems, VTOL rotors are susceptible to faults, which reduce trajectory tracking ability and may cause the vehicle to crash. In fact, the impact of faulty actuators have been shown in practice. For example,

This work was supported in part by NSF grants CNS-1750158, CNS-1955670, CNS-2111688 and CCF-2326606.

Jiaqi Lian, Neeraj Gandhi, Yifan Wang, and Linh Thi Xuan Phan are with the University of Pennsylvania, Philadelphia, PA, USA. {lianj2, ngandhi3, yyifan, linhphan}@seas.upenn.edu

Fig. 1. (Left) MV22 VTOL system crashing [2]. (Right) F-35B VTOL crashing [3].

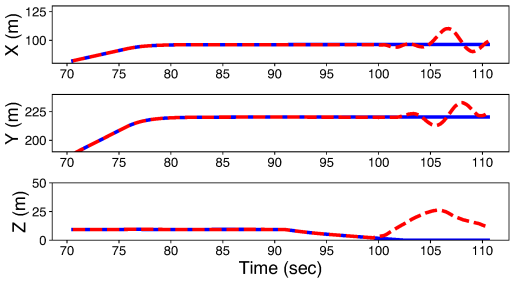

Fig. 2. A fault that is masked during cruise flight can crash the system when landing. In each subfigure, the blue solid line shows the trajectory without faults; the red dashed line shows the trajectory with 10% fault injected to Rotor 1 at 60.5 sec after takeoff.

Fig. 1 shows two real-world VTOLs that crashed because of a single faulty actuator. Thus, if we could quickly identify and mitigate the impact of the faulty actuators, we could improve the ability of these systems to avoid expensive and dangerous crashes, thereby ensuring safety for the nearby people, infrastructure, and environment.

Goal and challenges. Our goal is to develop techniques that leverage unique aspects of VTOLs to quickly detect and isolate *rotor* faults. We consider a rotor *faulty* if its output thrust is a scaled version of the desired output thrust.[1] One challenge towards this goal is to detect and isolate very *low-severity* faults. Such faults are especially hard to detect in VTOLs because they are masked by the fixed wings during cruising but can cause the system to crash as soon as the vehicle switches to landing. As an example, Fig. 2 shows the desired and achieved trajectories for a VTOL system that does not deploy any FDI method. When a 10% rotor fault occurs while cruising (using fixed wings), the system travels along its desired trajectory. However, upon switching to a 'vertical' landing mode, in which the ailerons and fixed wings play little role, the rotor fault causes a critical failure and crash shortly afterwards. This can happen in seemingly less serious cases, in which as little as 4% of thrust capacity is lost in one rotor. Due to the short time window during

[1] Developing recovery strategies for handling complete rotor failure is an interesting future work.

which the fault impact can be observed, one needs a real-time FDI method that can isolate such low-severity faults within a bounded amount of time while also being lightweight.

Another challenge is to accurately estimate the severity of the fault, which is important because different fault severity levels have different effects on the actual thrust output. In principle, if we know in advance the severity of a rotor fault – for instance, the rotor operates at only 70% of the commanded thrust – then, we could design recovery actions (e.g., by adapting the controller) for that particular fault mode. In practice, however, the degree of a fault can vary substantially: a faulty rotor may produce a thrust anywhere between 0% and near 100% of the desired thrust. Therefore, we need to be able to accurately determine the fault severity *at run time* to perform the right recovery actions. Recovery actions based on incorrect estimation of the severity level will not only fail to compensate for the impact of the fault on motion but could also further destabilize the controller.

Related work. The dynamic model of VTOLs has been well studied in prior studies [4], [5], but to our best knowledge, FDI techniques that consider the hybrid dynamics (i.e., partly fixed wing aircraft and partly multirotor aerial vehicle) of VTOLs are much less explored. The authors of [6] designed a controller capable of leveraging tilting rotors to recover from the effects of (a) a rotor that completely fails, or (b) a rotor-tilting mechanism that locks a rotor at a particular tilt angle. Unlike our work, however, it focuses on complete rotor faults and thus is orthogonal to our method.

FDI has been more extensively studied in both pure fixed-wing aircraft and pure multirotor aerial vehicles (MAVs). For instance, one approach for fixed-wing aircraft uses current sensors to perform fault detection; if more current than that of the fault-free condition is drawn, then a fault is deemed detected [7]. Conceptually, it should be possible to port such a technique to MAVs. However, additional hardware components would need to be added to each motor in the system, and we would need to know the fault-free current drawn despite natural changes such as the direction of the wind. Other techniques, such as [8], [9], propose using set-valued observers for FDI in fixed-wing UAVs; however, these have not been studied in the context of hybrid VTOLs.

Similarly, detecting rotor faults in multirotor systems is a topic of considerable research over the past several years. One approach is to use vibration data to find faulty propellers [10]–[12]. However, this approach does not scale well, even if the number of rotors does not grow significantly; for example, it can take over 100 seconds to detect a rotor fault in a quadrotor [11]. Another approach for finding faults is to use state observers [13]–[15], but such techniques would need to be adapted significantly to work for VTOLs because VTOLs have additional forms of actuation and a rigid body with different dynamics than MAVs. Analyzing the three angular velocities can also be used to quickly identify faults in systems with eight rotors [13], but this was only examined in the context of pure multirotor systems. The work in [16] extends the concept of [13] to first detect a sector of rotors that contain the faulty rotor, and then uses a finer-grain search within the sector to isolate the fault. This approach can scale to systems with dozens of rotors; however, it relies on the analysis of the steady state error magnitude in roll and pitch, which does not work for low-severity faults in our setting since (i) during cruising, the error is masked (by the fixed wings) and (ii) the fault can cause the system to quickly crash upon switching, making it impossible to obtain the steady state behavior that is required for fault detection.

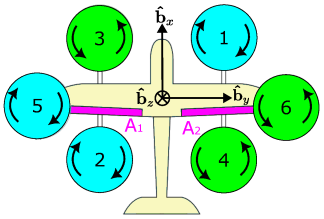

Fig. 3. The actuators in our modeled VTOL.

Finally, data-driven approaches for fault detection are becoming increasingly common, such as those of [17]–[20]. For instance, [20] proposes a supervised neural network to detect faults in a robotic system. More recent work has even considered how to run techniques based on neural networks in microcontrollers [21], although this involves adding an additional sensor *per* rotor. Furthermore, the trained model is specific to the sensor setup that is used and thus needs to be retrained to work with a different vehicle.

Contributions. We present a method for the FDI of rotor faults in VTOL systems, and make three key contributions. First, we adapt the 'fault-indexing' technique used in [13], [16] to identify a 'half-space' set of three rotors that contain the faulty rotor. Second, we propose a technique that uses the ratio of the error peaks that occur post-fault-detection to quickly isolate the faulty rotor. We leverage impulse analysis to efficiently isolate faults (in $\leq$ 2.5s in our simulations); then, by analyzing aileron control data, we can accurately estimate the severity level of a fault with a precision of within $\pm 2\%$. Third, we provide experimental validation using software-in-the-loop simulations of a DeltaQuad Quadplane in a PX4 environment. Online fault isolation enables the system to deploy a recovery technique, such as a different controller, that is better able to hew to the desired trajectory given that a rotor is faulty. *Our VTOL model and FDI technique are made available as an open source project.* [22]

II. Model

For concreteness, our technique is developed using the DeltaQuad Quadplane dynamics model provided by PX4 autopilot software [23], which models the system's force and torque dynamics. We made modifications to the system to make it conform better to passenger VTOLs. In particular, we added two rotors having rotation axis parallel to the body-frame z-axis, and we disabled the rotor with a rotation axis along the body-frame x-axis to better approximate the VTOL system that Joby is developing [24]. Fig. 3 shows the placement of the six rotors and two ailerons in our VTOL frame.

Attitude Representation. Let the inertial frame be represented as $\{\mathcal{F}^I\} = \{O_I; \hat{\mathbf{b}}_n, \hat{\mathbf{b}}_e, \hat{\mathbf{b}}_d\}$, where $\hat{\mathbf{b}}_d$ points downward, consistent with North-East-Down (NED) orientation as seen in related work [4], [6]. The body frame is denoted by $\{\mathcal{F}^B\} = \{O_B; \hat{\mathbf{b}}_x, \hat{\mathbf{b}}_y, \hat{\mathbf{b}}_z\}$, centered at the vehicle's center of mass (CoM). The aircraft's orientation relative to the inertial frame is captured by the rotation matrix R_I^B.

VTOL dynamics consist of three components: gravity dynamics, propulsion dynamics, and aerodynamics. These components constitute the total force and moments that act on the VTOL, as shown in (1) and (2).

$$F^B = F_g^B + F_r^B + F_a^B \tag{1}$$

$$M^B = M_m^B + M_r^B + M_a^B \tag{2}$$

Here, g represents the component attributable to gravitational influences, r denotes the contribution from propulsion thrust, and a signifies the aerodynamic forces and moments. The moment component M_m^B refers to resisting moments caused by the spinning of rotors.

Gravity dynamics. The effect of gravity on the VTOL system can be modeled with the equation

$$F_g^B = R_I^B \begin{bmatrix} 0 \\ 0 \\ mg\hat{\mathbf{b}}_d \end{bmatrix},$$

where m is the mass of the VTOL system.

Propulsion dynamics. Assuming negligible induced drag, the total thrust force is the collective force produced by the propellers:

$$F_r^B = \sum_{i=1}^{6} T_i^B = \sum_{i=1}^{6} T_i \hat{\mathbf{u}}_i^B,$$

where T_i is the thrust generated by the i^{th} rotor and $\hat{\mathbf{u}}_i^B$ defines the direction of the thrust force produced by the i^{th} rotor's rotation relative to the body frame. Then,

$$T_i = c_F \omega_i^2 \text{ and } F_r^B = c_F \sum_{i=1}^{6} \omega_i^2 \begin{bmatrix} \sin\chi_i \\ 0 \\ -\cos\chi_i \end{bmatrix},$$

where $c_F > 0$ is the thrust constant coefficient of the rotors, ω_i is the rotational speed of the i^{th} rotor, and χ_i is the angle between $\mathbf{b}_z$ and $-\mathbf{b}_y$, which defines the tilt of rotor's thrust direction relative to the vertical axis of the body frame.

Thrust moments M_r^B occur when the rotor-generated force is applied at a distance from the vehicle's CoM, creating a rotational effect due to the offset between the thrust's line of action and the CoM. These moments are calculated by considering the thrust magnitude and the radial distance from the CoM to the point of force application:

$$M_r^B = \sum_{i=1}^{6} (\mathbf{r}_i^B \times (T_i \hat{\mathbf{u}}_i))$$

Resisting moments M_m^B occur due to the torques generated by the rotor rotations, which produce a rotational effect that counteracts the vehicle's motion. These moments are calculated by considering the torque magnitude and the rotational direction of the i^{th} rotor relative to the body frame. Formally,

$$M_m^B = \sum_{i=1}^{6} \tau_i^B,$$

where τ_i^B is the torque generated by rotation of the i^{th} rotor relative to the body frame. This torque can be expanded into the form

$$\tau_i = (-1)^{d_i} c_K \omega_i^2, \;\; M_m^B = c_K \sum_{i=1}^{6} (-1)^{d_i} \omega_i^2 \begin{bmatrix} \sin\chi_i \\ 0 \\ -\cos\chi_i \end{bmatrix},$$

where $c_K > 0$ is the torque constant coefficient of the rotors, and $d_i \in \{0, 1\}$ is the rotation direction of the i^{th} rotor around its axis (i.e., clockwise or counter-clockwise).

Aerodynamics. The aerodynamic force F_a^B arises from airflow interaction with the airfoil, and plays a critical role in the aircraft stability. Three forces constitute the aerodynamic forces acting on the VTOL: drag (X^W), lateral forces (Y^W), and lift (Z^W). The total aerodynamic force vector is thus:

$$F_a^B = R_W^B \begin{bmatrix} X^W \\ Y^W \\ Z^W \end{bmatrix},$$

where $X^W = \bar{q} S C_X(\alpha, \beta)$, $Y^W = \bar{q} S C_Y(\beta)$, and $Z^W = \bar{q} S C_Z(\alpha)$. Here, $\bar{q} = \frac{\rho V_\alpha^2}{2}$ represents the dynamic pressure, where ρ is the air density, V_α is the airspeed, and S denotes the aerodynamic surface area of the model. The angle of attack is given by α, and β represents the angle of the velocity vector relative to the projection of x_b onto the wind plane. Coefficients C_X, C_Y, and C_Z correspond to drag, lateral, and lift forces, respectively. For simplicity, we assume lateral forces are negligible, and thus $C_Y = 0$. For lift and drag coefficients, approximations are made for two cases: the pre-stall region ($\alpha > 0$), and the post-stall region ($\alpha < 0$).

$$\begin{aligned} \text{Pre-stall region:} \quad & C_x(\alpha, \beta) \approx C_{D,0} + C_{D,\alpha}\alpha^2 \\ & C_z(\alpha) \approx C_{Z,0} + C_{Z,\alpha}\alpha \\ \text{Post-stall region:} \quad & C_x(\alpha, \beta) \approx c_1 \sin(2\alpha) \\ & C_z(\alpha) \approx c_0 + 2c_1 \sin^2(\alpha), \end{aligned}$$

where coefficients c_0, c_1, $C_{D,0}$,$C_{D,\alpha}$, $C_{Z,0}$, and $C_{Z,\alpha}$ are sourced from the PX4 documentation [23].

While cruising, the aerodynamic moments M_a^B play a critical role in stabilization. They are controlled by the deflection of ailerons, elevators, and rudder, which are denoted by δ_a, δ_e, and δ_r, respectively. (Our specific aircraft uses only ailerons.) The aerodynamic moments are approximated with the combination of roll torque $\Phi^B \approx qSbC_{La}\delta_a$, pitch torque $\Theta^B \approx qS\bar{c}C_{Me}\delta_e$, and yaw torque $\Psi^B \approx qSbC_{Nr}\delta_r$. The vector then becomes

$$M_a^B = \begin{bmatrix} \Phi^B \\ \Theta^B \\ \Psi^B \end{bmatrix},$$

TABLE I
PARAMETERS OF VTOL DYNAMICS

Signal	Description
c_F	Thrust constant coefficient of the rotors
c_K	Torque constant coefficient of the rotors
$C_{Z,0}$	Lift coefficient at zero angle of attack
$C_{Z,\alpha}$	Lift coefficient slope
$C_{D,0}$	Drag coefficeint at zero lift
$C_{D,\alpha}$	Incremental drag coefficient
C_{La}	Aileron coefficient
C_{Me}	Elevator coefficient
C_{Nr}	Rudder coefficient

where b is the wingspan, and $\bar{c}$ is the mean aerodynamic chord. The coefficient C_{La}, C_{Me}, and C_{Nr} are from PX4's configuration of the base model [23]. Table I summarizes these values.

Fault model. We consider VTOLs in which one or more rotors may become faulty during flight, but the rest of the system always works correctly. The severity level of a faulty rotor is represented by $\mathsf{s}_i \in [0, 1]$, where i is the rotor index. An individual rotor's thrust is thus redefined as $T_i' = \min\{(1 - \mathsf{s}_i)T_i, (1 - \mathsf{s}_i)\mathsf{f}_{\mathsf{max}}\}$, where T_i is the thrust commanded for this rotor by the control algorithm and $\mathsf{f}_{\mathsf{max}}$ is the nominal thrust capacity of the rotor. We assume no limit on the number of rotors that can become faulty, but at most one can fail at the same time for simplicity. Finally, we focus on rotor faults that occur during cruise flight, but can impact the vehicle safety during vertical motion. In our setting, the cruise trajectory is a straight trajectory with steady speed.

We assume the desired state at any given time t is given, and we can extract orientation information (including the roll angle ϕ and the pitch angle θ) by using a 9-axis inertial measurement unit (IMU). ϕ is defined as the UAV's rotation about $\hat{\mathbf{b}}_x$, and θ is defined as the rotation about $\hat{\mathbf{b}}_y$. Since our FDI approach relies on errors in roll and pitch, we define a limited state error vector $\mathbf{e} = [e_\phi, e_\theta]^\top$, where e_ϕ is roll angle error and e_θ is pitch angle error. The median value of $\mathbf{e}$ over a time window Δ_x is given by $\overline{\mathbf{e}}_x$.

III. APPROACH OVERVIEW

The work in [25] demonstrated how to use the *ratio* of steady-state error values $|e_\phi|$ and $|e_\theta|$ for fault isolation. It has two key insights: first, $\frac{|e_\phi|}{|e_\theta|}$ remains steady across fault severity levels and trajectories for a given rotor location; and second, fault severity can be estimated as a low-order bivariate polynomial that relates error magnitude and fault location to fault severity. However, their method is tailored specifically for pure multirotor systems, such as ModQuad [26] and does not apply to VTOLs due to its reliance on steady state error (which cannot be achieved within the short time window when switching from cruising to landing mode). Our simulations show that during cruise flight, VTOL controllers will deploy their ailerons to mask rotor faults until they switch to a vertical mode of flight. We adopt the core concept of using an error ratio, but apply it to examine the ratio of peaks in roll and pitch error following the occurrence of a fault; since we examine a system with fewer redundant rotors, the effect of a fault is more pronounced.

Fault detection. Fault detection establishes whether there exists a faulty rotor in the system. One simple approach is to perform error thresholding: if an error parameter (e.g., roll error) ever exceeds some preset threshold, we declare a fault to have occurred. However, this can result in a lot of false positives and false negatives, as noisy signals caused by factors such as wind can significantly affect the system. Instead, we adopt a dynamic approach as first proposed for MAVs [16].

This approach involves: (i) collecting a baseline window of data and computing the standard deviation in the magnitude of error in both roll and pitch over this window; (ii) periodically collecting new 'test' windows of data to compare to the baseline; (iii) comparing the difference in errors recorded over both windows to the weighted values of the standard deviation computed in i.

The baseline window is collected once the vehicle enters cruise flight, and it is updated each time a fault is isolated, thereby ensuring that our technique can detect multiple sequential faults.

Fault isolation. Once a fault has been detected, the next step is to isolate (identify) the specific faulty rotor and to estimate the severity of the fault that has occurred in that rotor. Our methodology solves this problem using three key insights: (I1) To narrow the search space, we first determine the half-space that contains the faults by checking the signs of roll and pitch error peaks after the fault is detected. (I2) Based on our observations in simulations, the range of error ratio of both overshoots is stable under noisy data across different severity levels. The faulty rotor can be found by comparing the error ratio with pre-recorded 'error profiles'. (I3) The impulse amplitude of the aileron which offsets the steady-state pose error can be expressed as a quartic function of severity level. The severity level of the determined candidate rotor then can be estimated to an accuracy of 2%.

This initial work performs FDI while the system is cruising (i.e., the majority of flight time), thus enabling recovery before the system switches to a vertical motion mode (where redundancy in the form of ailerons does not help).

IV. FAULT DETECTION AND ISOLATION

A. Detecting the existence of a rotor fault

Our detection method relies on comparing statistics over periodically collected windows of roll and pitch error data to a baseline window of data. Notably, after the introduction of a fault, the error will quickly re-stabilize to 0° while the system continues to cruise, as the ailerons mask the effects of a faulty rotor by adjusting the aircraft's roll and maintaining stability. When a rotor fault occurs, the imbalance in thrust would typically cause the aircraft to roll uncontrollably. However, the ailerons compensate for this imbalance by generating corrective roll moments by the embedded control algorithm, thus maintaining the desired attitude and reducing the immediate impact of the fault on the aircraft's trajectory.

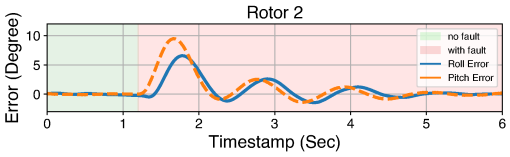

Fig. 4. Pose error over time when 25% fault is injected to Rotor 2. Green: Before fault injection; Red: After fault injection.

However, this masking no longer works as soon as the fixed wings become less useful, e.g., during landing, and the system may crash as a result, as illustrated in Fig. 2. If we examine the error closer to when the fault occurs, as in Fig. 4, we can make several observations. First, upon the occurrence of a fault, the system experiences a sudden spike in attitude error; these errors both have the same sign. Second, the error oscillates but dampens over time, and within a few seconds reaches close to $0°$; this means that the impact of a fault is completely masked soon after the fault occurs (at least, for the lower-severity, harder-to-detect faults that we are concerned with). However, this masking behavior does not occur during vertical flight, at which point the destabilization can be so rapid that a crash occurs. Thus, fault detection must be done before the fault is masked. Given the phase difference between these two errors, a fault is declared if the roll error **or** pitch error exceeds a predefined threshold. To address noise in inertia and gyroscope measurement, we apply a baseline buffer Δ_b which collects the steady-state error during cruise flight and takes the median $\bar{\eta}_b$ as the baseline to the error thresholds.

After Δ_b is collected, we proceed to collect a new window of data in a pose error buffer Δ_n while the system continues to cruise; this buffer is periodically cleared (i.e., we overwrite the buffer). The buffer's length introduces a trade-off between the sensitivity of fault detection and noise tolerance. A shorter buffer length results in fault detection that is more sensitive but less resilient to noise impact, possibly leading to a false positive detection. In contrast, a long buffer length is less sensitive and consumes more memory, which can be problematic for certain embedded systems, such as that of the Deltaquad QuadPlane our PX4 simulator used. For the robustness to noise, we compute the median error in the collected data ($\bar{\boldsymbol{\eta}}_n = [\eta_\phi, \eta_\theta]$) of Δ_n, and compare $\bar{\eta}_n$ to $\bar{\eta}_b$ to determine the presence of a fault. The η_ϕ represents the collected roll error and the η_θ represents the collected pitch error.

Formally, we perform detection by checking whether (3) or (4) is satisfied, i.e., checking whether

$$\|\bar{\boldsymbol{\eta}}_n - \bar{\boldsymbol{\eta}}_b > m\sigma_{\boldsymbol{\eta}_b}\|_1 \geq 1, \text{ or} \tag{3}$$

$$\|\bar{\boldsymbol{\eta}}_n - \bar{\boldsymbol{\eta}}_b\|_1 > m'\|\sigma_{\boldsymbol{\eta}_b}\|_1, \tag{4}$$

where m and m' are tunable error threshold constants and have range $R \in (0, 1]$ that controls the sensitivity of FDI, and $\sigma_{\boldsymbol{\eta}_b}$ is the standard deviation of pose error in the baseline buffer. In (3), the $l1$-norm $\|.\|_1$ indicates that at least one of the pose errors (roll or pitch) exceeds or equals to the threshold, signaling the presence of a fault. To enhance processing efficiency, all median variables are calculated using a streaming algorithm, as outlined in (3).

(4) is helpful in cases where only one of the roll or pitch error experiences a large spike upon the occurrence of a fault.

B. Locating and isolating the rotor fault

Once a rotor fault is identified, our algorithm employs a two-step process to pinpoint the faulty rotor. First, it narrows down the search area by determining the relevant half-space. Then, it uses the error ratio to determine the rotor most likely to be at fault. Once isolated, we can compute the severity of the fault by examining how the aileron control command changed in response to the fault. This, in turn, enables recovery actions that help the aircraft to land safely.

Locate half-space. We categorize rotors into two 'half-spaces' based on the sign of the roll and pitch errors. This technique of employing pose errors for fault localization draws from the approaches previously used in octorotors [27] and MAVs with dozens of rotors [16]. We extend the concept to VTOLs, in the process verifying that the core concept of 'faulty rotor sectors' as described by [16] carries over from MAVs to VTOL systems. In essence, if both roll error and pitch error are negative, then we claim that the fault exists in Half-Space I (Rotor 1, 3, and 5); if both errors are positive, then we claim that the fault exists in Half-Space II (Rotor 2, 4, and 6), where the rotor IDs are as per Fig. 3.

Identify faulty rotor. After the half-space is identified, the list of potential faulty rotors is reduced from 6 to 3 in our six-rotor model. Next, we establish the criteria to quickly determine the faulty rotor from the set of three. Our method analyzes the ratio of the peak observed error magnitudes in roll and pitch, $|e_\phi|$ and $|e_\theta|$, after the fault is detected, for all time points currently stored in Δ_n. The error ratio Q_e is the ratio of these two peaks, as formally written in (5):

$$Q_e = \frac{\max\limits_{t\in\Delta_n} |e_\phi(t)|}{\max\limits_{t\in\Delta_n} |e_\theta(t)|}. \tag{5}$$

To determine the usefulness of this ratio in identifying faulty rotors we performed an simulation analysis. Figure 5 shows the error ratio distribution for each rotor within the two half-spaces. Here, we show amalgamated results across all rotors, where for each rotor we tested a severity in the set $\mathsf{s} \in \{1\%, 2\%, ..., 30\%\}$ and ran 10 trials for each such rotor-severity pair. The PX4 simulator we used (see Sec. V) injected noise such as randomized wind into these trials, so these results show trends that exist despite the presence of external disturbances. We limited our tests to severities of value $\mathsf{s} \leqslant 30\%$, as larger-severity faults resulted in immediate crashes, and from a fault detection perspective it is clear that a fault exists when this occurs.

For a given half-space, notice how the error ratio ranges are concentrated, with the largest variance observed for Rotors 5 and 6, which also happen to be the rotors with the least impact on motion on account of being positioned along the fixed wing. Furthermore, the error ratios are concentrated at different values, e.g., the error ratio for Rotor 1 remains close

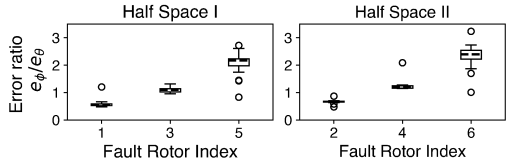

Fig. 5. (Left) Distribution of error ratios for rotors (1, 3, 5) located in Half-space I. (Right) Distribution of error ratios for rotors (2, 4, 6) in Half-space II.

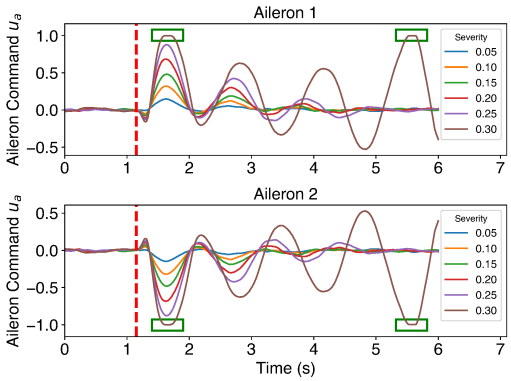

Fig. 6. Aileron responses when faults of varying severity are injected into Rotor 2. Green boxes are where aileron command saturation occurs.

to 0.5, while the ratio for Rotor 3 remains near 1.25. There is similarity *across* half-spaces, e.g., the ratio for Rotor 1 is quite similar to the observed ratio for Rotor 2. However, since half-space identification precedes this isolation step, we need not be concerned with similarity in error ratios for rotors in different half spaces. Since these error ratios are so consistent, even with external disturbances, we can keep these stored in memory. When a fault occurs at runtime, we can simply compare the at-runtime error ratio to the stored error ratios to isolate the faulty rotor within a half-space.

Compute severity level. Once a faulty rotor has been identified, the final piece of the FDI problem is to compute the severity of the fault. Our method is based on the following insight: When a fault occurs, VTOL controllers will try to compensate for attitude error caused by the faulty rotor(s) with other actuators, including the ailerons. Thus, based on how much the aileron control signal changes when attempting to fly in a straight line, we can compute the fault severity. For example, consider Fig. 6, which shows how the aileron control signals vary in response to faults of varying severity in Rotor 2. The servo motors operate within a range of $u_a \in [-1, 1]$, where u_a is a PX4 simulator-specified control signal sent to the VTOL system. As fault severity increases, the aileron control signal experiences larger amplitude oscillations, until they eventually reach saturation, as indicated by the green boxes for parts of the curve for $\mathsf{s} = 0.3$. If the ailerons reach saturation, the system tends to destabilize, demonstrated in this figure by the fact that the curve for $\mathsf{s} = 0.3$ does not experience dampening over time; eventually, this results in the system crashing.

To quantify fault severity, we analyze the peak aileron control signal after the fault has been injected and within a

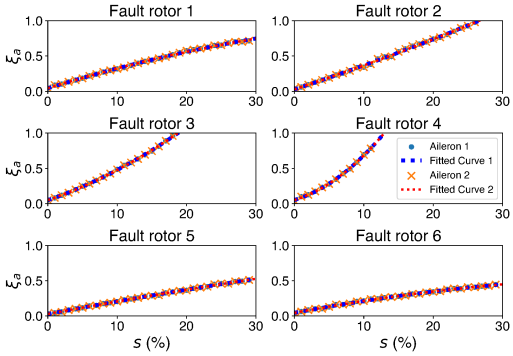

Fig. 7. Relationship between severity levels and the inital overshoot of aileron signal following rotor fault injection. Scattered data points represent actual measurements, while dashed lines depicts an overlaid quartic model.

TABLE II

EXPERIMENTALLY FOUND COEFFICIENTS FOR (6).

Rotor ID	c_4	c_3	c_2	c_1	c_0
1	171.90	-226.13	106.04	18.24	-0.96
2	-22.18	50.33	-40.87	40.78	-1.51
3	-20.67	43.21	-36.77	34.12	-1.61
4	22.85	-26.71	-4.140	24.00	-1.02
5	109.79	-121.56	55.97	47.50	-1.30
6	-206.66	215.30	-13.78	51.68	-2.13

time window, Δ_n, as depicted in Fig. 7. This figure plots the peak control signal of *either* aileron, ξ_a, across a spectrum of severity levels for all six rotors, ranging from 1% to 30% or those severe enough to induce a crash. We then model the severity level as a quartic function of ξ_a, which can be collected from the altitude controller, illustrated by (6).

$$s = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 & c_4 \end{bmatrix} \begin{bmatrix} 1 & \xi_a & \xi_a^2 & \xi_a^3 & \xi_a^4 \end{bmatrix}^T \quad (6)$$

Once the faulty rotor is determined, we can use the measured peak aileron signal ξ_a over time window Δ_n to compute the severity s with the simulated coefficients reported in Table II. Due to symmetry, ξ_a is consistent for symmetrically placed rotors in each half-space.

V. IMPLEMENTATION

A. Simulation model and environment

This section outlines the simulation framework and the adaptations made to a base model for our simulaton analysis. Our simulator is available as an open source project [22].

Base model. Our simulation utilizes the Deltaquad Quad-Plane VTOL as the foundational model, a hybrid fixed-wing and multirotor system. It uses four rotors dedicated to VTOL operations, transitioning to forward flight through integrated control surfaces. Unlike traditional fixed-wing models, it employs only one rotor for forward propulsion, and does not have elevators or rudders. This base model is depicted on the left side of Fig. 8.

Modification to the base model. To better simulate popular commercial eVTOLs, such as that of Joby Aviation [24], we made a few modifications. We added two rotors on the front part of the fixed wings; these were symmetrically placed 0.707 m from $\hat{\mathbf{b}}_x$. The horizontal back rotor was disabled,

Fig. 8. (Left) PX4's Deltaquad QuadPlane VTOL model. (Right) The modified model has six vertical rotors and disables the horizontal rotor.

so only the six vertical rotors could be used. The modified aircraft is shown on the right of Fig. 8.

Simulation environment. We built on top of the PX4 simulation stack [23], which rendered the environment and robot in Gazebo [28]. We selected PX4 due to the relatively high fidelity of its simulations, ease of portability to future real-world experiments, and built-in disturbances (e.g., wind) that would ensure the simulations were not operating in an ideal environment. We used QGroundControl [29] for flight planning and telemetry monitoring.

B. Simulation setup

The FDI module operated at a sampling frequency of 42 Hz, and accordingly stored data for the data buffers Δ_b and Δ_n. The duration of Δ_n was configured to 1.2 s (which corresponds to fifty data points). We chose this buffer size to balance the sensitivity of fault detection against the likelihood of false positives.

In each simulation, we used QGroundControl to set a waypoint for the system to follow, and the VTOL would takeoff before following a line to the desired waypoint and land. We evaluated the system across a range of fault severities $s \in \{1\%, 3\%, 5\%, 10\%, 15\%, 20\%, 25\%\}$ to cover a wide range of hard-to-detect faults (each rotor-severity pair was tested with 50 trials).

We conducted 50 trials for each severity level across all rotors, and in each case assessed the FDI system's accuracy rate. Here, a trial is considered as "accurate" when the FDI system accurately estimates the severity level *within a ±2% margin* of error with correct fault rotor index identified.

Flight parameters. The PX4 simulator was configured to model the VTOL's operational parameters, including a takeoff to a height of 9.4 m and a cruising speed of 5 m/s. When traveling between waypoints, we assume the system to be cruising, i.e., control surface deflection and angular velocity is small. Two types of noise were present in the simulations: sensor noise from the Inertial Measurement Unit (IMU) and actuator control signals, and environmental noise modeled as variable wind speeds ranging from 0 to 2.5 m/s.

VI. Evaluation

To evaluate our proposed FDI method, we performed a series of experiments using the simulation platform described in Section V. Our key questions were: 1) How accurately can our technique identify the faulty rotor and severity? 2) How quickly can it isolate a faulty rotor? And finally, 3) once a fault is isolated, how effective is our method in enabling recovery actions to allow the system to land safely?

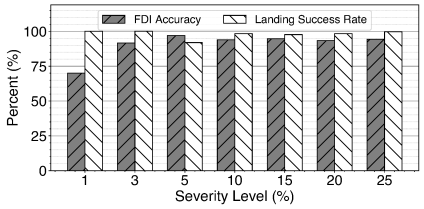

Fig. 9. Landing success rates remain high even as FDI accuracy improves with faulty severity.

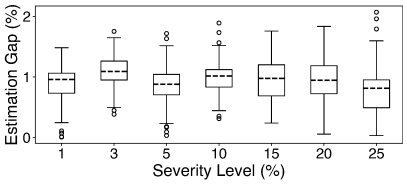

Fig. 10. Fault severity estimates are close to those of injected faults.

FDI accuracy. Figure 9 presents the performance of our method across various fault severity levels (total 300 trials for each severity level, and 50 trials for each rotor), showcasing both the precision in fault rotor identification and the accuracy in severity estimation. We observe that accuracy improves as fault severity grows. This is expected, as higher-severity faults cause more pronounced impulses in the state errors they produce, and thus are easier to identify. In contrast, lower-severity faults produce a smaller effect and are harder to detect. Even for faults of very low severity, at 3%, our method still achieves an accuracy rate of 91.67%. Fig. 10 further shows that, even when our estimate is not perfect, the estimated fault severity remains very close to the injected value.

Our method does have some limitations. First, if fault severity is below 3%, the fault signal may be masked by white noise, making it difficult for our FDI method to distinguish the fault. Second, if the fault severity is above 30%, the sudden loss of so much thrust causes the provided controller to destabilize. We expect that these problems can be mitigated by using a more robust controller.

Speed of isolation. Fig. 11 shows the temporal statistics for those simulations that we accurately performed fault detection and isolation for, across different severity levels and for all six rotors. The y-axis of Fig. 11 represents the "time spent" in seconds (i.e., the moment a fault is injected until the FDI system successfully detects and isolates the fault). For the lowest-severity faults, FDI takes longer, since these faults produce the least impact on motion. However, even in moving from a 1% severity fault to a 3% severity fault, we already see substantial improvement in the speed of FDI, and for faults of severity greater than 3%, FDI remains near-constant around 2.3s.

Adjusting certain parameters, such as the amount of recent sensor data we store, could speed up FDI, at the cost of consuming more memory (which is often constrained on embedded platforms).

Recovery performance in landing. Ultimately, the goal of any FDI technique is to enable recovery actions. While this paper focuses on developing such an FDI method, we

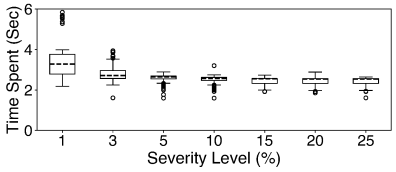

Fig. 11. The time spent by the FDI module to determine fault severity after fault injection.

simulated with a simple recovery method to assess whether we could indeed improve safety outcomes by triggering recovery post-FDI. Our simple recovery strategy does the following: if the estimated severity level is determined $s' \approx s$, the updated rotor input becomes $T_i' = \min(\frac{T_i}{1-s'}, \mathsf{f_{max}})$. The results of this simulation are also shown in Fig. 9 as the columns for 'Landing Success Rate'. We observe a 100% success rate for both 1% and 3% severity levels – that is, the vehicle lands safely in all cases despite faults across all tested faults and trials. Even for more severe faults, landing success rates remain higher than 90%. As severity level increases from 5% to 25%, the landing success rate increases from 91.93% to 99.67%, which is likely due to the improvement of FDI accuracy with severity level.

VII. Conclusion

Fault detection and isolation presents a significant challenge in VTOL systems, but it is becoming increasingly important due to the growing number of real-world ventures building these products to operate in urban environments to interact or even carry humans. In this work, we focus on developing an FDI method capable of working on the hard-to-detect faults that are easily masked during cruise flight, but which can cause catastrophic failures during vertical motion, when the fixed wings cannot play a role in fault masking. Our approach first identifies the half-space containing a rotor fault, then performs impulse analysis of the error ratio to isolate rotors, followed by impulse analysis of aileron control signals to estimate fault severity to within 2%.

When compared with the state-of-the-art recovery technique (with unique controller design) outlined in [6], our methodology stands out for its efficiency. Our FDIR method stabilizes the VTOL with only one-fourth of time required by the recovery technique in [6]. This significant improvement not only highlights the efficacy of our method in addressing the critical aspects of VTOL reliability but also underscores its potential to enhance operational safety and effectiveness without the complexity and latency associated with conventional FDIR solutions.

References

[1] R. Rothfeld, M. Fu, M. Balać, and C. Antoniou, "Potential urban air mobility travel time savings: An exploratory analysis of Munich, Paris, and San Francisco," *2021 Sustainability*, 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/4/2217

[2] N. N. Australia, "Shocking footage shows military helicopter crash into us warship — 9 news australia," https://www.youtube.com/watch?v=AQz_Paiikzg, 2022, accessed: 2024-02-29.

[3] WFAA, "Video: Pilot ejects from F-35B near White Settlement, Texas," https://www.youtube.com/watch?v=t9GBHNaYzcs, 2022, accessed: 2024-02-29.

[4] G. Ducard and M.-D. Hua, "Modeling of an unmanned hybrid aerial vehicle," in *2014 CCA*, 2014.

[5] W. Khan and M. Nahon, "Modeling dynamics of agile fixed-wing uavs for real-time applications," in *2016 ICUAS*, 2016.

[6] M. Mousaei, J. Geng, A. Keipour, D. Bai, and S. Scherer, "Design, modeling and control for a tilt-rotor VTOL UAV in the presence of actuator failure," in *2022 IROS*, 2022.

[7] G. Fuggetti, A. Ghetti, and M. Zanzi, "Safety improvement of fixed wing mini-uav based on handy fdi current sensor and a failsafe configuration of control surface actuators," in *2015 IEEE Metrology for Aerospace (MetroAeroSpace)*, 2015, pp. 356–361.

[8] R. Deng, J. Chen, M. Wang, Z. Shi, and Y. Zhong, "Fault detection and isolation for a fixed-wing uav swarm system with uncertainties and disturbances," in *CCC*, 2019, pp. 4919–4924.

[9] P. Rosa and C. Silvestre, "Fault detection and isolation of lpv systems using set-valued observers: An application to a fixed-wing aircraft," *Control Engineering Practice*, vol. 21, no. 3, pp. 242–252, 2013.

[10] B. Ghalamchi and M. W. Mueller, "Vibration-based propeller fault diagnosis for multicopters," *ICUAS 2018*.

[11] B. Ghalamchi, Z. Jia, and M. W. Mueller, "Real-time vibration-based propeller fault diagnosis for multicopters," *TMECH*, vol. 25, pp. 395–405, 2020.

[12] X. Zhang, Z. Zhao, Z. Wang, and X. Wang, "Fault detection and identification method for quadcopter based on airframe vibration signals," *Sensors*, vol. 21, no. 2, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/2/581

[13] M. Saied, B. Lussier, I. Fantoni, C. Francis, H. Shraim, and G. Sanahuja, "Fault diagnosis and fault-tolerant control strategy for rotor failure in an octorotor," *ICRA 2015*.

[14] A. Freddi, S. Longhi, and A. Monteriù, "Actuator fault detection system for a mini-quadrotor," *ISIE 2010*.

[15] Z. Cen, H. Noura, T. Susilo, and Y. Younes, "Robust fault diagnosis for quadrotor uavs using adaptive thau observer," *JINT*, vol. 73, pp. 573–588, 2014.

[16] N. Gandhi, J. Xu, D. Saldaña, and L. T. X. Phan, "Detecting and isolating faulty rotors in aerial vehicles with dozens of rotors," *Submitted to IROS*, 2024, available: https://drive.google.com/file/d/1vgUjzz3qKXuVN_IHn2ggO6MVE3X4xDW2.

[17] P. K. Goel, G. Dedeoglu, S. I. Roumeliotis, and G. S. Sukhatme, "Fault detection and identification in a mobile robot using multiple model estimation and neural network," *ICRA 2000*.

[18] V. Verma and R. G. Simmons, "Scalable robot fault detection and identification," *Robotics Auton. Syst.*, vol. 54, pp. 184–191, 2006.

[19] U. Lerner, R. E. Parr, D. Koller, and G. Biswas, "Bayesian fault detection and diagnosis in dynamic systems," *AAAI/IAAI 2000*.

[20] A. L. Christensen, R. O'Grady, M. Birattari, and M. Dorigo, "Fault detection in autonomous robots based on fault injection and learning," *Autonomous Robots*, vol. 24, pp. 49–67, 2008.

[21] R. Puchalski, A. Bondyra, W. Giernacki, and Y. Zhang, "Actuator fault detection and isolation system for multirotor unmanned aerial vehicles," *MMAR 2022*.

[22] J. Lian and N. Gandhi and Y. Wang and L. T. X. Phan, "Open-source software for IROS 2024 – Online Rotor Fault Detection and Isolation for VTOLs," 2024, publicly available from https://drive.google.com/drive/folders/11ctbnoKoGDQTbBqFuvHSoNZN59h1RD0Q?usp=sharing.

[23] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 ICRA*, 2015.

[24] A. Stoll and J. Bevirt, "Development of evtol aircraft for urban air mobility at joby aviation," pp. 1–11, 05 2022.

[25] N. Gandhi, D. Saldaña, V. Kumar, and L. T. X. Phan, "Self-reconfiguration in response to faults in modular aerial systems," *IEEE Robotics and Automation Letters*, 2020.

[26] D. Saldaña, B. Gabrich, G. Li, M. Yim, and V. Kumar, "Modquad: The flying modular structure that self-assembles in midair," in *2018 ICRA*, 2018.

[27] M. Saied, B. Lussier, I. Fantoni, C. Francis, H. Shraim, and G. Sanahuja, "Fault diagnosis and fault-tolerant control strategy for rotor failure in an octorotor," in *2015 ICRA*, 2015, pp. 5266–5271.

[28] N. P. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulators," *IROS*, 2004.

[29] QGroundControl Development Team, "QGroundControl," 2019, software available from http://qgroundcontrol.com.