ABSTRACT

Title of thesis: FAULT DETECTION AND EMERGENCY PATH PLANNING FOR FIXED WING UAVS

Ruth Gomez Quezada, Master of Science, 2023

Thesis directed by: Doctor Huan Xu
University of Maryland Department of Aerospace Engineering

The implementation of Uncrewed Aerial Vehicles (UAVs) for civilian and military purposes requires safety protocols. Control surface failures are among the most common issues in fixed-wing UAVs. This study presents two heuristic-based algorithms developed to detect such faults. The Mahalanobis Distance Fault Detection Algorithm (MDFDA) employs the Mahalanobis distance to identify anomalies in UAV states. On the other hand, the Fuzzy Logic Fault Detection Algorithm (FLFDA) uses fuzzy logic to identify jammed control surfaces. In the event of a fault, an emergency path planning algorithm is initiated. This algorithm leverages terrain and population data to pinpoint safe landing zones. However, the type of fault the system encounters will impact the aircraft's ability to reach these safe zones. In this study, unreachable zones are delineated based on the specific control surface fault detected. These zones are areas where the aircraft cannot land or traverse due to the fault. By employing the proposed protocol, the aircraft can detect a fault during mid-flight, select a safe landing zone within its reachable range, and mitigate the effects of the control surface fault. This approach enhances the chances of preserving the aircraft and ensures the safety of the surrounding population.

FAULT DETECTION AND EMERGENCY PATH PLANNING FOR FIXED WING UAVS

by

Ruth Gomez Quezada

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2023

Advisory Committee:
Doctor Huan Xu, Chair/Advisor
Doctor Jeffrey Herrmann
Doctor Michael Otte

Dedication

To Diego: *te prometí que saldríamos de ésta.*

Acknowledgments

First and foremost, this work would not have been possible without funding from Axient corporation, the Maryland Industrial Partnerships, The University of Maryland Department of Aerospace Engineering and graduate school. I would like to thank each of them for their support.

I would like to thank my advisor Dr. Huan (Mumu) Xu, thank you for finding me capable to complete these tasks, you were an awesome mentor. I would also like to thank Lina Castano for leading me and believing in me even when I did not, thanks for all the advice you have given me, I will never forget them. Thank you to Mike Briggs for directing this project from Axient.

Thank you to all the undergraduate and graduate students that helped in this project, to Edward Carney for his previous work from which this research is built upon, to Miles Begley for all your help with hardware, flight tests and for coding the Arduino fault injection file, to Nathaniel Wunderly for coding the Raspberry Pi file, to Ameya Vishwanath for helping during flights, to our pilot Jeriel Cortes, and Zach Bortoff for developing the metrics code.

On a personal note, I would like to thank my husband Jorge, thanks for loving me through this and for cheering me up when I needed it. Thanks to my son for inspiring me with his curiosity.

Gracias a mis padres Rodolfo y Guadalupe, por su amor incondicional y su apoyo durante todos estos años.

Thanks to my brother for always making me laugh even in the thoughest moments. Thanks to my friends Ileana, Karla, Sitlali, and Ana for believing in me.

A special thank you to Guada for putting out fires and silencing my demons.

Finally, thank you to the women that came before me, without you I would not be in a position to pursue my dreams.

Table of Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

BRRT	Bi-directional Rapid exploring Random Tree algorithm
CS	Control Surface
DEM	Digital Elevation Model
FLFDA	Fuzzy Logic Fault Detection Algorithm
FN	False Negative
FP	False Positive
IMM-UKF	Interactive Multiple Model and Unscented Kalman Filter
LCI	Land Cover Index
LIDAR	Light Detection and Ranging
LS	Landing Strip
LZ	Landing Zones
MDFDA	Mahalanobis Distance Fault Detection Algorithm
MCC	Mathers Correlation Coefficient
TRI	Terrain Roughness Index
UAV	Uncrewed Aerial Vehicle

Symbols

C_L	lift coefficient
$Coeff$	coefficients associated to each control surface fault
$CS_{certainty}$	certainty of fault in a control surface
DEM	elevation gathered from Digital Elevation Model
DEM_{range}	range of elevation in a LS
$Fault_{ID}$	ID number of jammed control surface
h	relative altitude
K_{DEM}	coefficient for DEM factor
$K_{DEM_{range}}$	coefficient for DEM range factor
$K_{distUAV}$	coefficient for distance from LS to UAV factor
K_{LC}	coefficient for LC factor
K_{TRI}	coefficient for TRI factor

L	lift
$MD(t)$	Mahalanobis distance
MD_{lim}	Mahalanobis distance threshold value
p	roll rate
$\dot{p}$	roll acceleration
q	pitch rate
$\dot{q}$	pitch acceleration
r	yaw rate
$\dot{r}$	yaw acceleration
S	covariance matrix
S	wing area
V	total velocity
v_a	airspeed velocity
v_x	velocity in x-axis
v_y	velocity in y-axis
v_z	velocity in z-axis
$W(i)$	weight score of the landing strip
$\overrightarrow{x_{MD}(t)}$	state vector for MDFDA
$\overrightarrow{x_{FL}(t)}$	state vector for FLFDA
δ_r	rudder deflection angle
θ	pitch
$\overrightarrow{\mu(t)}$	state cumulative average vector
ρ	air density
ϕ	roll
ψ	yaw

Chapter 1: Introduction

1.1 Motivation

As autonomous flights gain more popularity for both commercial and military purposes, likelihood of accidents increases. History has shown that when new technologies are introduced, there is often an initial period of increased property damage and injuries as people adapt to them. Some examples of it are nuclear accidents [1], microwave oven related injuries [2], and even hybrid or electric cars [3]. As the industry leans towards a future with fully autonomous flights, it becomes crucial to establish and rigorously test a hazard mitigation plan. Similar emergency path planning algorithms have already been developed for autonomous land vehicles available to the public [4] [5] [6].

According to Williams [7], who analyzed factors contributing to accidents in various US Army aircraft, aircraft malfunctions were the most common causal factors in four out of five cases. This percentage exceeded those attributed to maintenance issues. Similar conclusions were reached by Ghasri and Maghrebi [8]. This study will focus on faults in control surfaces, which fall under the category of aircraft malfunctions. Such faults occur when the signal from the autopilot isn't registered by the control mechanism. For small Uncrewed Aerial Vehicles (UAVs), this mechanism is often a servo. The severity of the fault can result in a range of mission impacts. In cases where the fault's effect is minimal, such as when one servo gets stuck at a zero-degree angle, the mission can proceed with minor adjustments, as is often the situation. However, there are cases in which the servo

might abruptly switch to its maximum deflection and stay there due to an electric failure. In these instances the mission becomes unattainable, and the UAV should either attempt to land or deploy a parachute. Such faults demonstrated to pose significant risks when the UAV is flying over populated areas [9] prompting extensive research to enhance public safety [10] [11].

If the chosen course of action is to land, the faults afflicting the UAV will inevitably impact its maneuverability. For instance, a rudder fault would constrain the UAV's ability to alter its course. These specific scenarios must be taken into consideration by the algorithm responsible for determining safe landing zones, a concept previously developed by our lab.

Our security protocol is characterized by Fig. 1.1 similarly to SafeEye project [12], follows a series of steps: First, a fault is detected. Next, a landing zone where the faulty aircraft can reach is found using terrain and population data. Lastly, a Bi-directional Rapid exploring Random Tree algorithm (BRRT) algorithm is run to find the best path from the aircraft current position to the selected landing zone.

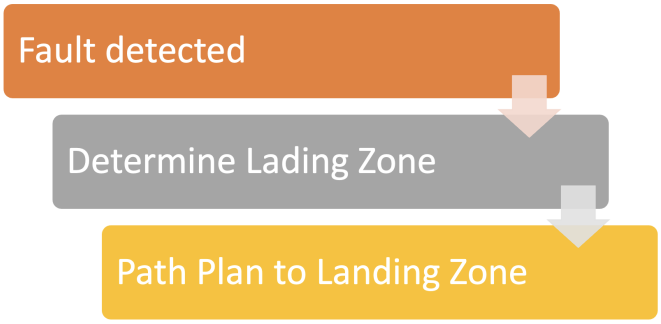


Figure 1.1: Fault detection and emergency landing protocol

1.2 Contributions

This research presents the development of two heuristic-based algorithms designed for detecting faults in fixed-wing UAVs. Unlike much of the existing literature that relies on

simulation data to evaluate their fault detection algorithms, these algorithms were trained using real flight data in which simulated faults were introduced. This approach enhances the decision-making process by increasing the likelihood of selecting the optimal course of action, ensuring the safety of individuals and the preservation of the aircraft.

While studies investigating the impact of faults on path planning are abundant in the context of multi-rotor UAVs [13] [14] [15], there is a notable gap in terms of faulty fixed wing UAVs. Utilizing both simulation and experimental flight data, restricted zones were calculated where a UAV with a specific fault would be unable to transit without complicated maneuvers. This process is best illustrated in Fig. 1.2.

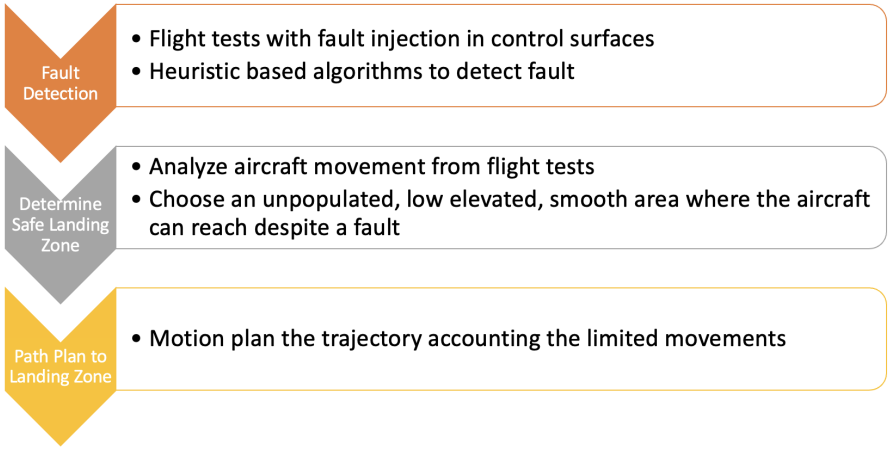


Figure 1.2: Fault detection and emergency landing contributions

Chapter 2: Fault Detection

2.1 Introduction

Autonomous flights introduce the potential for operations beyond the reach of GPS or communication signals. In the event of a system anomaly, timely detection becomes critical to preempt the aircraft from becoming uncontrollable. Prior studies have examined fault identification and isolation in UAVs. For instance, servo failure is a complex issue, often requiring substantial computational resources and extensive data for implementation. Zhang et al. [16] where a Kalman Filter is applied to this situation. A similar approach is the one from Gheorghe et al. [17] where a servo loop model was develop to compare the actual deflection vs the command coming from the autopilot. Cork and Walker [18] in 2007 implement the knowledge of the UAV nonlinear model to detect anomalies in control surfaces using their Interactive Multiple Model and Unscented Kalman Filter (IMM-UKF) algorithm. Henry in 2012 [19] used the Linear Parameter Varying technique based on H_∞. All of the above examples have one thing in common, they need a model of the controls to the actuator or the whole system. Creating such models, involving transfer functions based on force and moment equations, demands a comprehensive understanding of the vehicle's dynamics. The IMM-UKF algorithm by Cork and Walker, for instance, required numerous flight and wind tunnel tests to build its nonlinear dynamics model, in turn consuming substantial time, money, and computational resources.

Aviation history showcases instances where heuristic or rule-based approaches have

been adopted. In 2009, Heredia et al. [20] developed an algorithm to monitor small UAVs in a swarm system by having a reference UAV that measured the states of the rest of the vehicles using sensors and cameras, with this they were able to detect UAVs that were not behaving as commanded. For fault detection, in 1995 Douglas and Speyer [21] create an algorithm for fault detection using a rule-based approach algorithm, however, there was no fault isolation. Lee et al. [22] used a Random Forest algorithm to prioritize the data that was calculated by weighting similarity measure. Freeman et al. [23] compared a data-driven algorithm versus a model-driven one, both trained by the same flight test data sets for failures in control surfaces. Corbetta et al. [24] combined physics-derived principles with empirical equations to enhance their fault detection algorithms.

A crucial aspect of fault detection is fault isolation, which determines the affected control surface. There are different types of servo failures, if the servo is not responding to the incoming signal and it is stuck at a random angle or zero degree angle it is called a jammed actuator. Another kind of fault is presented when the servo is moving, but the physical degree does not match the input signal, this is known as a weak signal. This study concentrates on the stuck servo fault type.

The heuristic-based algorithms utilize flight test data to conduct real-time simulations for fault detection. The dataset includes flights with injected faults in ailerons, rudders, and elevators.

In this chapter, two heuristic based algorithms for fault detection are introduced the Mahalanobis Distance Fault Detection Algorithm (MDFDA), and the Fuzzy Logic Fault Detection Algorithm (FLFDA). These algorithms were chosen because they have been develop before in detecting anomalies in a different scope. MDFDA and FLFDA also comes with a number of how bad the fault is affecting the states of the aircraft (Mahalanobis distance for MDFDA and a scale from 0 to 1 in FLFDA). Section 2.2 describes the theory behind them and how to apply them in the presented problem. Section 2.3 describes the

experimental flight data collection methodology and the analysis for the threshold values of the algorithms. Finally, section 2.4 contains the performance of both algorithms using the flight data collected.

2.2 Algorithms

The algorithms proposed in this research are designed to detect faults that could jeopardize a mission by altering the chosen path or rendering the aircraft incapable of flight. The heuristic approach is preferred due to its rapid development, not necessitating an intricate model for implementation. For small UAVs, prompt fault detection is essential, as their dynamics are swiftly affected due to their compact size. While many previously developed algorithms for fault detection require computational power that's feasible on standard desktop computers, adapting such programs for compact onboard computers suitable for UAVs can be challenging. The featured algorithms are streamlined enough to run effectively on lightweight onboard computers, enabling fault detection within the system itself without the need to transmit data to a ground station for processing. This allows UAVs to carry out autonomous missions even in locations with limited ground control communication.

2.2.1 MDFDA

The Mahalanobis distance is a metric that quantifies the distance between multiple characteristics [25], and these characteristics have been previously correlated with states in autonomous research. Similar to the approach of Park et al. [26], this algorithm is trained using normal flight datasets, distinguishing it from rule-based approaches where unique data is required for each fault. Lin et al. [27] utilized the Mahalanobis distance to detect anomalies across the entire UAV system, identifying anomalies in over 50 distinct attributes, ranging from temperature sensors to motors and telemetry. These attributes were

grouped into smaller sets of 2 or 3 to reduce computational demands. In this study, a similar approach is adopted, employing 13 states to detect anomalies or faults specifically in control surfaces.

The state variables used in this algorithm ($\overrightarrow{x_{MD}(t)}$) are altitude ($h$ in meters), Euler angles (roll ϕ, pitch θ, and yaw ψ in degrees), angular rates (roll rate p, pitch rate q, and yaw rate r in degrees per second), angular accelerations (roll acceleration $\dot{p}$, pitch acceleration $\dot{q}$, and yaw acceleration $\dot{r}$ in degrees per squared second), and velocities in each axis (v_x, v_y, v_z in meters per second). With these state variables, the Mahalanobis distance can be calculated using Eq. 2.1:

$$MD(t) = \sqrt{(\overrightarrow{x_{MD}(t)} - \overrightarrow{\mu(t)})^T S^{-1} (\overrightarrow{x_{MD}(t)} - \overrightarrow{\mu(t)})}. \tag{2.1}$$

Where $\overrightarrow{\mu(t)}$ represents the vector containing the cumulative averages of each variable up to time t, and S is the covariance matrix for all 13 state variables. To calculate S, 13 distincts nominal state vectors were collected from data sets. These vectors created a 13x13 state matrix that was used to compute a covariance matrix.

The state matrix used in this algorithm was composed by vectors of 4 different flight tests, the reason for this was to have a wide range of flight conditions, but it can also be done gathering vectors from a single nominal flight. The number of state variables used in this algorithm was selected through tests, first a 5x5 state matrix was tested, with that false positive rate would increase. However, at a 18x18 state matrix show a slower detection time and a lower detection rate. The state variables used were set to the 13 variables in the vector $\overrightarrow{x_{MD}(t)}$ was determined to yield the best overall performance.

To detect a fault using MD a threshold value (MD_{lim}) must be determined. Through tests and data analysis that value was established at 1000. The output of MDFDA at time t is 0 if the state is nominal and 1 if a fault was detected , i.e., the $MD(t)$ is greater than

1000. Algorithm 1 shows the pseudo code for this method, where lines 5 and 6 described an altitude filter where the aircraft is either on land, taking off or landing. In these cases the algorithm will not detect a fault i.e., $Fault_t$ is 0. If the altitude exceeds 20m, the MD is calculated and compared to the MD_{lim} value of 1000. If the calculated MD surpasses this threshold, a fault is detected, and consequently $Fault_t$ assigned value is 1.

Algorithm 1 MDFDA(MD_{lim}, S^{-1}, t)

1: **for** Every t **do**
2: $\overrightarrow{x_{MD}} \leftarrow$ state vector at time t
3: $\overrightarrow{\mu} \leftarrow$ cumulative state averages
4: **if** $h(t) < 20$ **then** altitude check
5: $Fault_t \leftarrow 0$ i.e. the aircraft is not flying
6: **else**
7: $MD_t \leftarrow \sqrt{(\overrightarrow{x_{MD}} - \overrightarrow{\mu})^T S^{-1} (\overrightarrow{x_{MD}} - \overrightarrow{\mu})}$
8: **if** $MD_t > MD_{lim}$ **then**
9: $Fault_t \leftarrow 1$
10: **end if**
11: **end if**
12: **end for**

2.2.2 FLFDA

Fuzzy logic was originally introduced in the 1960s by Zadeh [28] which proposed the idea of a non-absolute logic. This novel approach acknowledges that certainties possess varying degrees of magnitude based on the fluctuations within the values of its components. In essence, traditional binary outcomes, such as 0 for 'no' and 1 for 'yes,' give way to a range between 0 and 1, signifying the degree of certainty. The extent of this certainty relies on the attributes of the components involved. For instance, the query 'Was the flight successful?' would not be characterized by a simple binary response, but rather by a score ranging from 0 to 1 that signifies the level of success. This degree of success would be contingent upon the attributes considered by the organization in question to determine a

rating. For example, if mission completion is deemed the most crucial attribute, it would be assigned a higher value compared to other attributes, such as on-time takeoff.

Fuzzy logic has found application in anomaly detection in various domains. Adhikari et al. [29] utilized it in transmission lines, Maruyama et al. [30] for air conditioning systems, and Demidova et al. [31] for energy conversion systems. In the field of UAVs, fuzzy logic has been employed for path tracking and obstacle avoidance by Donk et al.[32], for sensor fault by Gao et al.[33], and for control strategies by Hazaf et al.[34] among other applications. However, its usage in detecting servo faults from flight data has not been explored.

FLFDA uses a distinct state vector $\overrightarrow{x_{FL}(t)}$ containing the following state variables: altitude (h in meters), Euler angles (roll ϕ, pitch θ, and yaw ψ in degree), angular rates (roll rate p, pitch rate q, and yaw rate r in degree per second), and velocities in the z-axis (v_z in meter per second). Table 2.1 provides an overview of the conditions, referred to as characteristics, indicative of faults in specific actuators, along with coefficients representing the weights assigned to each condition for every control surface. These conditions correspond to spikes in various state variables linked to each faulty servo, determined through logical reasoning and analysis of collected data. The coefficients associated with each condition serve to amplify the effect of each fault on each state, with their values established through empirical selection. A sensitivity analysis was conducted to evaluate the influence of variations in these coefficients, as elaborated in Section 2.4.

Algorithm 2 describes the steps in FLFDA to detect a fault. Similar to MDFDA, lines 2 through 4 execute the altitude verification step to halt fault detection while the aircraft is on the ground. In lines 6-9, the algorithm encompasses the comparison of system state variables with the conditions specified in Table 2.1, and subsequently assigns the coefficients linked to each anomaly. This process extends across all three fault categories: aileron,

Table 2.1: FLFDA limit values and coefficients

Control Surface	Conditions	Coefficient
Elevator	$\|\theta\| > 20$ deg	3
	$\|q\| > 40$ deg/s	3
	$\|v_z\| > 10$ m/s	1
Rudder	$\|\phi\| > 30$ deg	2
	$\|q\| > 50$ deg/s	1
	$\|r\| > 50$ deg/s	3
	$\|v_z\| > 20$ m/s	1
Ailerons	$\theta <$ -15 deg	1
	$\|\phi\| > 50$ deg	2
	$\|p\| > 50$ deg/s	3
	$\|v_z\| > 10$ m/s	1

rudder, and elevator. Line 10 calculates the degree of certainty, as specified in Eq. 2.2

$$CS_{certainty} = \frac{Coeff_{applicable}}{Coeff_{total}}. \tag{2.2}$$

Here $Coeff_{applicable}$ denotes the addition of the coefficients for anomalies present of the fault to a specific control surface (CS), $Coeff_{total}$ represents the accumulation of coefficients across all conditions within the CS. In lines 12 and 13 of the algorithm 2 from the three different CS (aileron, rudder and elevator), the fault with the highest certainty is selected as the primary fault, which ID corresponds to the output of the algorithm. Where 1 represents ailerons, 2 represents elevator, and 3 represents rudder. However, the output signal for a fault will only happen if the certainty is greater than 0.4. This implies that if there is a 40% likelihood of a fault, the algorithm will transmit the ID of the control surface where the fault is most likely to exist. This 40% threshold was determined based on assessments across various control surface failure data sets.

Algorithm 2 FLFDA(CS_{coeff}, t)

for Every t **do**
  $\overrightarrow{x_{FL}} \leftarrow$ state vector at time t
  if $h < 20$ **then** altitude check
    $Fault \leftarrow 0$
  else
    for Every CS **do**
      if $\overrightarrow{x_{FL}} \in Cond_j$ **then**
        $CS_{fault} \leftarrow CS_{fault} + Coeff_j$
      end if
      $CS_{certainty} \leftarrow CS_{fault} / \sum_{1}^{k} Coeff_i$
    end for
    $Certainty \leftarrow$ Greater $CS_{certainty}$
    $Fault \leftarrow CS_{ID}$ Of greatest certainty
    if $Certainty < 0.4$ **then**
      $Fault \leftarrow 0$
    end if
  end if
end for

2.3 Flight Test and Data Analysis

Data collection was performed using a Carbon-Z Cub SS model aircraft which has a 2.1 m wingspan and a weight of about 3 Kg as illustrated in Fig. 2.1. The data used in this project was gathered from 20 flight tests taken over the span of one year. This aircraft was modified to operate autonomously, enabling data collection and the injection of faults.

This Carbon Cub was equipped with the following:

- S3 LiPo battery to power electronics
- Smart G2 LiPo battery to power motor
- Pixhawk autopilot
- 4 channel multiplexer

- primary and secondary receivers
- basic avionics
- Raspberry Pi computer
- Arduino Board

Figure 2.1: Instrumental Carbon Cub SS

The flight tests were designed to create a comprehensive dataset encompassing various control surface malfunctions. The data collection plan was adjusted after each flight test based on the successful testing of control surfaces. This adaptability was necessary due to occasional failures in generating data files for random flights within the Raspberry Pi code.

Across the 20 flight tests that were performed 6 were rudder fault, 4 were elevator fault, 7 were left aileron fault, and 3 were right aileron fault. From the 6 rudder fault, one had a fault deflection of $\pm$25 deg, two were at $\pm$15 deg, one at 5 deg, one at -5 deg and one at 0 deg. From the 4 elevator fault data sets, one was stuck at a 0 deg angle, another one was at 10 deg, one was at $\pm$5 deg, and one had faults injected at a deflection of 15 deg and -5 deg at different times. From the 7 left aileron fault, three were tested at a 25 deg deflection,

two at 0 deg, one at 20 deg, and one flight with -10 deg and -25 deg angles injected at different times. From the 3 right aileron fault, one had a 0 deg deflection, one had a $\pm$25 deg deflection, and one with the right aileron stuck at -10 deg and -25 deg.

2.3.1 Autopilot

A Pixhawk was chosen as the autopilot for this project, and it was integrated with the control surface servos, motor, telemetry, Arduino, and Raspberry Pi. Out of the total 21 flights, 16 were conducted in mission mode, while the remaining flights were carried out in manual mode. The mission configuration encompassed a sequence of 26 waypoints, creating a route that traversed back and forth within a field defined using QGroundControl.

During missions, the procedure involved taking off, stabilizing the airplane and switching to the second receiver from the radio transmitter switch, let the autopilot take control of the aircraft, switch back to manual mode, and land the aircraft. To facilitate this process, a multiplexer was incorporated into the aircraft, enabling the switch between primary and secondary receivers (manual and mission mode). Taking into consideration the augmented weight due to the added electronics and battery, the flights had a duration of approximately 5 to 7 minutes, significantly shorter compared to the original weight of the manufactured fuselage.

The Pixhawk generated a flight log that could be visualized through PX4 autopilot flight review online as shown in Fig.2.2. In the figure, the red line corresponds to the trajectory during manual mode, the blue line signifies the trajectory in stabilized mode, and the purple line represents the trajectory in mission mode. In stabilized mode, the pilot retains control, while the autopilot employs the flaps to maintain wing leveling.

Figure 2.2: Flight path visualization in PX4. Blue section shows the stabilized mode, purple is the mission mode, and red is the manual mode

2.3.2 Fault Injection

The Arduino Board was integrated into the system, establishing a connection between the autopilot and all the control surface servos. The Arduino was equipped with a fault injection code designed to simulate signal disruptions in control surfaces, enabling the imposition of two distinct deflection degrees. The injection of faults could be executed using an extra switch on the radio transmitter, which featured three positions.

In the initial position, no alterations were made, and the Arduino allowed the output signal from the autopilot to directly reach the servos. Shifting to the second position activated the fault injection, implementing the predetermined fault specified in the Arduino code at the first designated deflection angle. In the third position, the second deflection angle was injected. This setup facilitated the execution of tests, with each flight limited to evaluating one control surface at two different deflection angles.

Fig. 2.3 shows the fault switch for a flight, when the switch is at zero position, there is no fault injected i.e., the signal from the autopilot is send to the actuators without disruption.

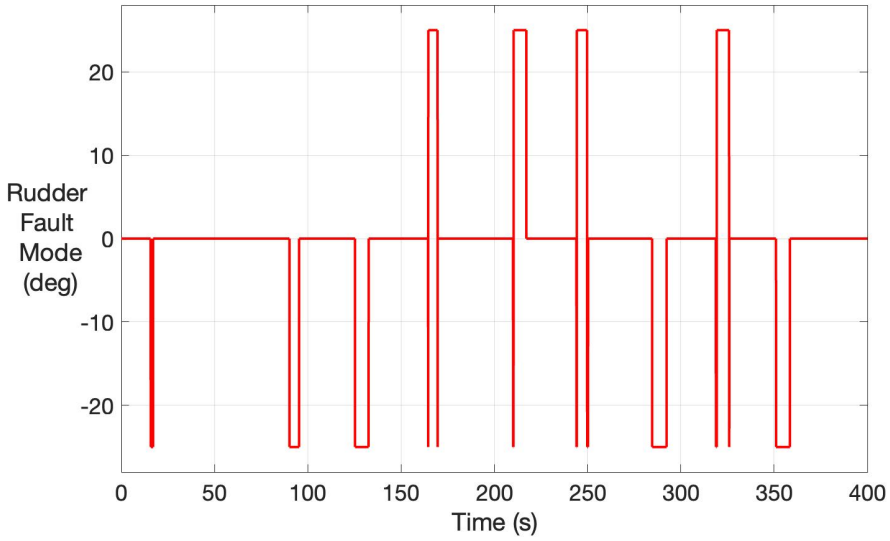


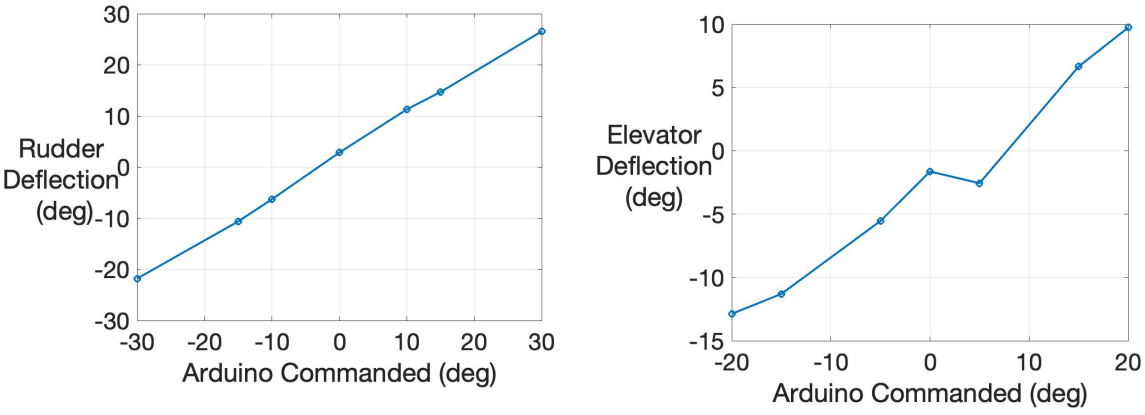
Figure 2.3: Fault mode for Rudder

The rest of the time, the mode for the first fault with +25 deg deflection or the second mode with -25 deg deflection is applied.

During flight tests the crew noticed that the angles set on the Arduino code did not perfectly corresponded to the ones physically displayed in the airplane. For instance, when the rudder was programmed at a 10 deg angle showed a physical 11.3 deg angle on the other hand, a -10 deg set in the Arduino code showed a -6.277 deg angle deflection.

The sign convection for rudder is positive if the trailing edge of the rudder is deflected towards the right wing, and negative towards the left wing. For elevator, the positive is the trailing edge pointing upwards and negative is pointing downwards. Fig. 3.5 shows the graphs for Arduino commanded vs physical angle of deflection for both rudder and elevator. In this graph, it is shown how the physical rudder deflection for commanded negative angles is always much smaller than the positive ones. This affected the performance of the algorithms as the aircraft states were less affected by the negative rudder fault.

In the elevator, the commanded positive 5 deg deflection still exhibited a physical neg-



(a) (b)

Figure 2.4: Arduino commanded angle vs physical angle of deflection for Rudder (a) and Elevator (b)

ative 2.56 deg. When the Arduino degrees increase either positive or negative, the physical increments decrease. This could be because the limits for that servo are smaller and as the signal approaches that limit, the servo can not reach it.This is probably caused by the position position of the servo on the aircraft.

Moreover, the natural trim value for the elevator is 3.27 deg, which has almost a 4 deg difference with the Arduino zero. The ailerons did not exhibit this kind of discrepancy. Across most angles the offset was around 2 degrees that were negligible for this analysis. Because of this, when the text mentions an injected fault degree, it is referring to the Arduino commanded deflection. To establish the real deflection, measurements were taken on the ground utilizing rulers and transporters.

2.3.3 Data Collection

The Arduino code directly overrides the signal from the autopilot, resulting in the Pixhawk's visualized data lacking timestamps and angles associated with injected faults. To address this, a Raspberry Pi was introduced to the hardware setup, serving as an intermediary to amalgamate both datasets. This involved the Raspberry Pi collecting telemetry data

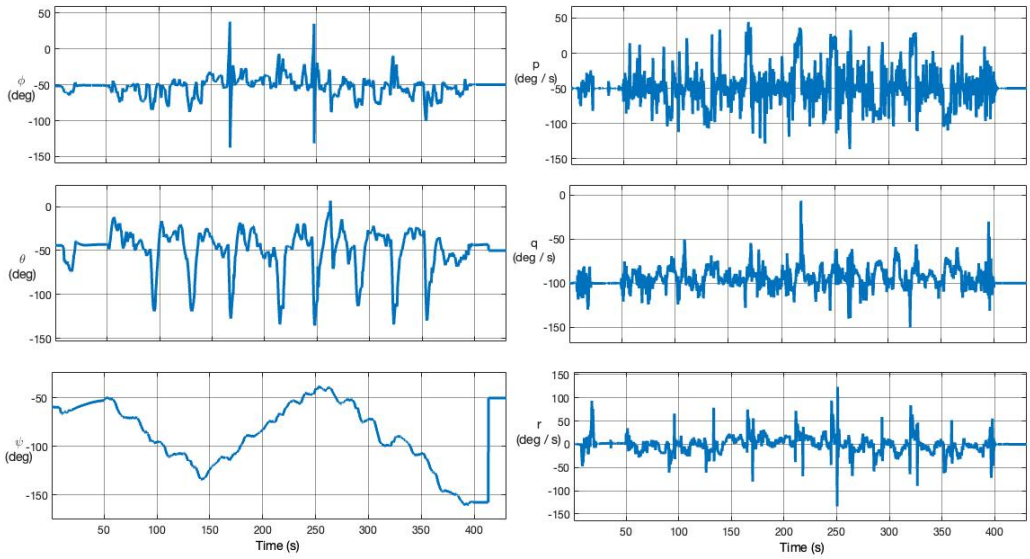


Figure 2.5: Euler angles and angular rates for flight test with rudder fault

from the autopilot, as well as the angular degrees and control surface information from the Arduino. The amalgamation process yielded a '.json' file, which was subsequently subjected to analysis using MATLAB.

In this MATLAB analysis, threshold values applicable to FLFDA were determined. By examining instances when faults were injected, discernible peaks in different state variables were identified, each corresponding to a specific actuator impacted by the fault. This pattern is visualized in Fig. 2.5, which exemplifies how MATLAB facilitates the visualization of the data.

2.4 Results

The evaluation of both algorithms' performance involved the use of the F-1 score and the Matthews Correlation Coefficient (MCC). The F-1 score is a harmonic mean that combines precision and recall, calculated using the metrics of false positives, false negatives, true positives, and true negatives. Meanwhile, the Matthews Correlation Coefficient (re-

ferred to as MCC in the tables) also utilizes the same variables as the F-1 score, but serves to quantify the disparity between the predicted and actual values of a model.

Both algorithms transmit their output as impulse signals. Consequently, a fault was deemed detected if the algorithm output presented a pulse during the time the fault was being injected. For FLFDA, only the signal corresponding to the actual $Fault_{ID}$ is being considered. For example, if the real fault was on rudder, only the impulse with the $Fault_{ID}$ 3 was actually detecting the fault.

At times, faults were injected while the aircraft was on the ground during data recording, primarily to assess the extent of deflection. To mitigate the impact of this scenario, a filter was incorporated into the algorithms, as outlined in Section 2.2. This filter functionally eliminated the injected fault signal from consideration if the aircraft's altitude was below 20 meters, serving as a confidence threshold indicating that the aircraft was not airborne. Importantly, this modification was specific to the statistical analysis code, ensuring that ground-level faults would not distort the false negative rate.

The flights were classified into distinct categories to facilitate the analysis of algorithm performance. Specifically, ten flights featured aileron faults, six flights simulated rudder faults, and four flights introduced elevator faults. The second category focused on the magnitude of the angles commanded by the Arduino. This encompassed seven flights for fault deflections less or equal than $\pm$10 deg (excluding zero angle deflection), eleven flights for deflections greater than 10 deg, and five flights for faults at 0 deg. From the data sets, three flights injected faults less than and greater than $\pm$10 deg using one fault mode for the smaller degree and the other for the bigger one, these flights were used to calculate the F-1 score and the MCC for both groups. The third and final category refers to the mode in which the flight was performed: mission mode (autonomous) or manual mode (piloted).

Fig. 2.6 shows MDFDA output on the same flight the data in Fig. 2.5 and 2.3 were collected, with rudder fault at $\pm$ 25 deg. The value at the bottom scope is the MD at each

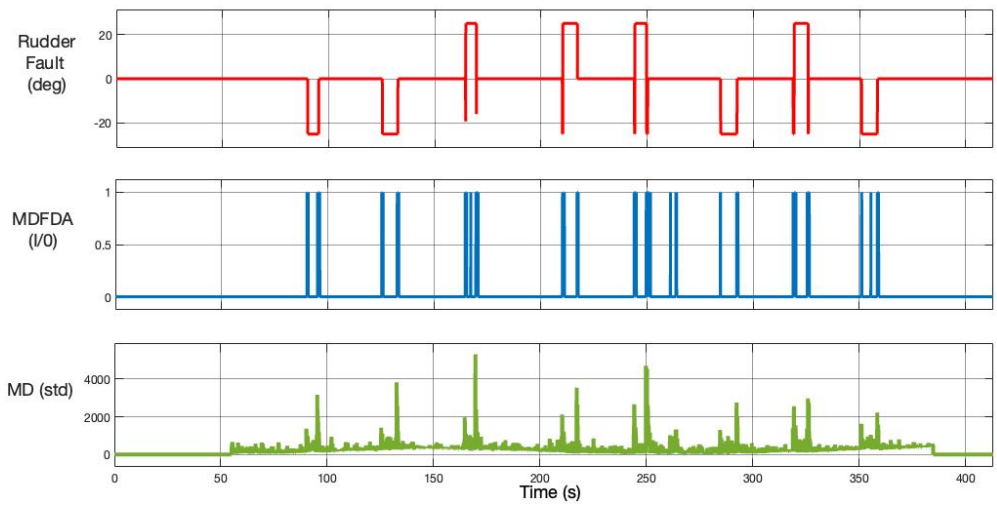


Figure 2.6: MDFDA for rudder fault at ± 25 deg. Top scope shows real fault, middle scope the detected fault, and bottom one shows MD at each time step

time step, in the middle scope is the fault detected by the algorithm, and on the top is the actual fault. Table 2.2 contains the statistics for MDFDA such as average False Negative (FN) rate (%), False Positive (FP) rate(%), F-1 score, MCC, detection time (sec) that is the time it takes for the fault to be detected once it is present, and detection rate (%) from all flights.

Table 2.2: MDFDA Statistics

	FP Rate (%)	FN Rate (%)	F-1 Score	MCC	Detection Time (sec)	Detection Rate (%)
Overall	1	16	0.8687	0.8684	3.28	79
Aileron Faults	1	12	0.8977	0.8884	3.57	86
Rudder Faults	1	2	0.9793	0.9718	2.92	89
Elevator Faults	0	47	0.6305	0.6633	3.14	50
Faults $\leq$ |10deg|	1	20	0.8519	0.8573	4.07	77
Faults = 0deg	1	33	.7156	0.7446	4.46	67
Faults > |10deg|	1	11	0.9125	0.9009	2.38	82
Mission Flights	1	16	0.8722	0.8710	3.99	78
Manual Mode Flights	2	18	0.8581	0.8604	1.17	82

Fig. 2.7 shows the FLFDA output for that same flight with ± 25 deg fault in rudder. The

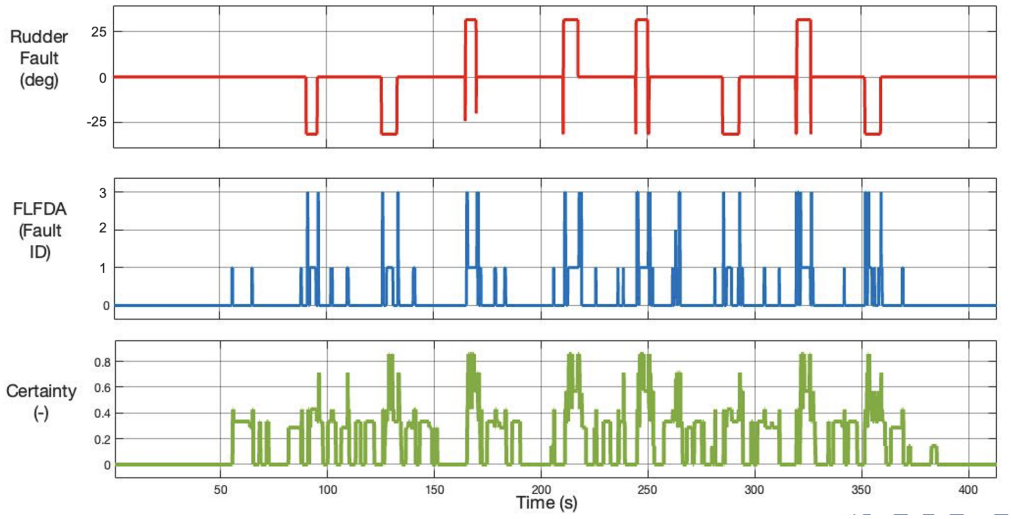


Figure 2.7: FLFDA for rudder fault at $\pm$ 25 deg. Top scope shows real fault, middle scope the detected fault ID, and bottom one shows certainty at each time step

first scope shows the degree and the time in which faults were injected. The last scope is the higher certainty for each time step. Recall Algorithm 2, where the the three certainties (between 0 and 1) are calculated at each time step, and the one with higher value is the predominant. That certainty is plotted last in Fig. 2.7. If the value reaches higher than 0.4, the fault ID will appear in the middle plot (3 for rudder, 2 for elevator, and 1 for ailerons). In some cases, there is zero probability the aircraft is experiencing actuator fault, which can be observed in scope 3. Table 2.3 contains the statistics of FLFDA (FP, FN, F-1 Score, MCC, Detection Time, and Detection Rate) for all categories of flight tests.

MDFDA was proven to be more effective overall, with a detection rate of 79%, F-1 score of 0.8687, and an MCC of 0.8684. It demonstrated faster detection of faults greater than 10 deg compared to smaller or at trimmed angle, which aligns with the effects a larger deflection angle. MDFDA have a higher detection rate in rudder, with 89%, ailerons comes next with an 86% detection rate, while elevator faults have only a 50% detection rate. This algorithm performed better for manual mode flights with a detection rate of 82%.

FLFDA exhibited lower performance when compared to MDFDA, achieving an overall

Table 2.3: FLFDA Statistics

	FP Rate (%)	FN Rate (%)	F-1 Score	MCC	Detection Time (sec)	Detection Rate (%)
Overall	1	23	0.7982	0.7910	4.15	73
Aileron Faults	1	13	0.8813	0.8632	4.54	86
Rudder Faults	1	26	0.7818	0.7818	3.27	65
Elevator Faults	3	46	0.6153	0.6156	4.49	53
Faults $\leq$ |10deg|	2	32	0.6930	0.7025	4.76	70
Faults = 0deg	2	24	.8026	0.7969	4.5	74
Faults > |10deg|	1	19	0.8364	0.8226	3.78	74
Mission Flights	2	18	0.83	0.8268	4.70	78
Manual Mode Flights	0	39	0.7029	0.6836	2.5	60

detection rate of 73% along with a longer delay and lower F-1 score and MCC values of 0.7982 and 0.7910, respectively. Nonetheless, FLFDA managed to detect aileron faults at an 86% rate, matching MDFDA's success rate. However, it detected only 65% of rudder faults and 53% of elevator faults. Notably, FLFDA performed more effectively in detecting faults at 0 degrees, surpassing the performance of the other algorithm. However, it displayed shortcomings in detecting faults with smaller and larger degrees.

In terms of flight mode, FLFDA demonstrated better performance in mission mode compared to manual mode, yet both algorithms achieved the same 78% detection rate. It's worth mentioning that the statistics for FLFDA were calculated considering instances where the detected fault ID matched the injected fault. In other words, if the algorithm detected an elevator fault (ID 2) while the injected fault was a rudder fault (ID 3), the algorithm was deemed unsuccessful in identifying the actual fault.

The results of the sensitivity analysis indicate that the coefficients presented in Table 2.1 have a significant impact on the performance of the FLFDA algorithm. When the coefficients for different conditions in ailerons were varied by ± 1, the detection rate decreased by approximately 28.57%. Additionally, the detection time exhibited variations ranging from -0.5 seconds to 1 second. In this context, a positive value denotes a slower detection,

while a negative value indicates a faster one. These findings highlight the sensitivity of the algorithm's performance to the coefficients and emphasize the importance of carefully tuning these values to achieve optimal results.

In the case of rudder faults, the sensitivity analysis revealed that varying the coefficients led to a notable improvement in the detection rate, increasing it by approximately 40% compared to the previous rate. The detection time exhibited a range of variations between -11 seconds and 0 seconds.

However, for elevator faults, the sensitivity analysis showed a decrease of around 35% in the detection rate when the coefficients were varied. The variation in detection time for elevator faults ranged from -3.5 seconds to 1.2 seconds. These results highlight the complex relationship between the coefficients and the algorithm's performance, emphasizing the need for careful calibration to achieve the desired outcomes.

The deficiency for both algorithms to detect elevator faults can be explain with the offset of injected fault vs actual physical deflection covered in Sec.2.3. where it is explain that the actual deflection of the elevators never reached more than +10 deg and less than -13 deg, where the fault injection algorithm output a $\pm$20 deg. Using this knowledge, the category of faults greater than 10 deg can be thought off differently because elevator faults were never truly above 10 deg physically. Without taking into account elevator faults, the detection rates for larger faults increased to 86% and 80% for MDFDA and FLFDA respectively.

Regarding FLFDA, the algorithm was able to detect a fault exactly when it was switched in and when it was switched out, although it may not be the same ID that the fault injected. This is interesting to discuss further, this happens because the algorithm is detecting the change in state. When the fault is injected via the radio transmitter, the affected control surface servo moves with torque of about 65 oz/in and breaks the steady state of the aircraft. The UAV tends to oscillate for the abrupt movement, considering the faults are switched in at straight and leveled flight. The same occurs when the fault is disabled and the servo

jumps back to the deflection the autopilot is commanding.

The same behavior can be observed for MDFDA, but not only for faults of large deflections. The algorithm detects the fault when it is being injected and when it is being released because, as per Eq.2.1, the calculation of the Mahalanobis distance requires the averages of the state values from the beginning of the flight until time t. This means that when a fault is injected, the algorithm notices it because the this new arrangement of state values is new, but as time progresses with the fault being active, a series of this rare state vectors are being used to calculate the mean used in the formula. The mean increases, now covering the range of values that are characterized by the anomaly, that is why the MDFDA does not output 1 throughout the entire duration of the fault injection. When the mode is switch and the autopilot recovers the control of the aircraft, another detection is made because the normal control is now an strange performance of the aircraft, until enough time has passed that it becomes the norm again. For which, it can be said that the MDFDA detects the deviation of the aircraft state vectors from the steady state (e.g. cruise stage) of the aircraft.

2.5 Conclusion

Both of the methods developed in this study performed acceptably overall; however, they both exhibited limitations in detecting elevator faults due to the narrow scope of testing. In contrast, aileron faults were detectable by both algorithms promptly. Generally, and specifically for each flight, the detection time was consistently quicker in MDFDA. The downside of this method is its inability to isolate a fault. An integration is proposed where MDFDA acts as a filter to detect anomalies, and FLFDA is employed to characterize the fault. Fuzzy logic has not been utilized in servo fault detection in autonomous systems before. Considering the performance of many methods based on this approach, it is worthwhile to continue refining and investing in this methodology.

Chapter 3: Emergency Path Planning

3.1 Introduction

If a fault is detected, the UAV will decide whether to land or perform a controlled descent in a safe area. Emergency path planning establishes a safety protocol for determining a secure landing location for the aircraft that ensures the safety of the population and property surrounding the UAV. Choosing a landing zone while the aircraft is experiencing an anomaly mitigates risks and could avert the destruction of the vehicle. This calculation should encompass the impacts of these anomalies and the constraints they impose on the aircraft's mobility.

Alam and Oluoch [35] made a survey for safe landing zones identification techniques. While Light Detection and Ranging (LIDAR) data combined with sensors is a common practice for the selection of landing zones in helicopters (Scherer et al. [36], Schmitt [37]), quadrotos (Kakaletsis et al. [38], Lane et al. [39], and Ariante et al. [40] for example) and in space exploration (Johnson et at. [41]), the combination of the elevation maps and population density data to ensure public safety proposed on 2019 by Carney et al. [42] is the based this section is build upon. The selection of a safe landing zone has no guarantee that the aircraft can reach the zone if it is experiencing a control surface fault.

Path planning for autonomous systems experiencing a fault has been developed for quadrotor UAVs like Chamseddine et al. [43], and vertical take-off and landing UAVs (Xia et al. [44]). For fixed wing UAVs, there has been a lot of interest in obstacle avoidance

path planning (Radmenesh et al. [45]), and motion planning where the dynamics are different than multi-rotor UAVs (Qu et al. [46]). The appearance of a failure in a fixed wing UAV increases the complications, where this has only been studied before by Ayhan et al. [47]. They created a path planning algorithm when experiencing engine failure, but they performed the calculation on the ground. There is a lot of improvement that can be made in here, for the algorithm proposed here can detect the fault, select a safe landing zone and path plan to it mid-flight. Mid-flight landing strip identification have been done before without accounting for dynamic limitation [48] [49].

An added level of complexity involves the requirement for a landing strip or runway for takeoff and landing. In the context of emergency path planning, this is achieved by selecting a rectangular area where the UAV can land safely. The terrain plays a significant role in this process, as a rough surface could potentially damage the aircraft. Elevation is also a crucial factor; higher elevations may hinder the aircraft's ability to climb and reach the designated landing zone. Furthermore, this algorithm takes into account population data. Using LandScan datasets, the terrain is divided into areas with high and low population densities. The potential landing zones are restricted to low-population areas only.

In this chapter, an addition has been made to our previous emergency path planning algorithm to detect failing control surfaces and adjust the range of available landing strips for the UAV. The goal is to consider the reduced mobility of the system due to the control surface failure. Section 3.2 provides a summary of our previous work and outlines the algorithms used for selecting landing strips and path planning. In Section 3.3 we detail the procedure for fault simulation using Simulink, the creation of out-of-reach zones for landing strip selection, and the subsequent path planning to avoid these zones. Finally, Section 3.4 discusses the comparison between landing strip selection in the presence of elevator faults and aileron/rudder faults.

Table 3.1: Land Cover Classifications

Land Classification	Integer	Viable for Landing
Unclassified	0	True
Water	1	False
Bare Land	2	True
Crop Field	3	True
Vegetation	4	True
Dense Vegetation	5	False
Building	6	False
Build-Up Surface	7	True

3.2 Previous Work

Using the emergency path planning algorithm, the selection of landing zones was determined through the analysis of various factors including the Digital Elevation Model (DEM), Terrain Roughness Index (TRI), Land Cover Index (LCI), and LandScan data. The DEM is derived from LIDAR data sets, while the TRI is computed from the DEM data using Eq.3.1

$$TRI(i,j) = \frac{\sum_{p=i-1}^{i+1}\sum_{q=j-1}^{j+1}|DEM(p,q) - DEM(i,j)|}{8}. \tag{3.1}$$

This metric utilizes the elevation data from the current cell and the eight surrounding cells, immediately adjacent to it. Subsequently, the data is categorized into the following groups: leveled (0 to 80m), nearly leveled (81 to 116m), slightly rugged (117 to 161m), intermediately rugged (152 to 239m), moderately rugged (240 to 297m), highly rugged (498 to 958m), and extremely rugged (greater than 958m). LCI classifies land according to the descriptions provided in Table 3.1, employing Landsat data derived from satellite imagery. Additionally, LandScan data is employed to collect population density information, which is further processed to represent binary values. True values indicate pixels with a population equal to or greater than seven people, while false values represent areas with fewer inhabitants.

To choose potential landing zones (LZ), the criteria that each cell must meet can be adjusted based on preferences for aircraft preservation. In this study, a more lenient set of criteria for LZ was employed, which included the following:

- Population value of zero
- DEM value less than 50% of the current UAV altitude
- TRI value below 2m or LCI different from 1 and 6

This simulation involves a fixed-wing UAV, which necessitates the presence of a Landing Strip (LS). The algorithm identifies an area in which the majority of cells fall into the LZ category. This area should take the form of a rectangle measuring 90m by 270m. Among the pool of potential landing strips, a weight factor is computed to assess and choose the area that most appropriately meets the landing criteria. This weight is determined by Eq. 3.2

$$\begin{aligned} W(i) = [&K_{DEM} * DEM_{avg}(i) + K_{TRI} * TRI_{avg}(i) + K_{LC} * LC_{avg}(i) \\ &+ K_{DEMrange} * DEM_{range}(i) + K_{distUAV} * DistUAV(i)]. \end{aligned} \tag{3.2}$$

Where K_{DEM}, K_{TRI}, K_{LC}, $K_{DEMrange}$, and $K_{distUAV}$ are the coefficients assigned to each factor (DEM, TRI, LC, DEM_{range}, and distance from LS to UAV). These coefficients are subsequently multiplied by the respective factors, which are the averages for all cells within the strip in the case of DEM, TRI, and LC. The coefficients were empirically chosen with an emphasis on optimizing the landing strip's quality. DEM_{range} signifies the disparity between the highest and lowest elevations, as detailed in Eq. 3.3

$$DEM_{range}(i) = |max(DEM(i)) - min(DEM(i))|. \tag{3.3}$$

Fig. 3.1 illustrates an example of the LS selection algorithm, with the green areas representing available LZs, yellow indicating regions with high population density, the blue triangle denoting the current position and orientation of the UAV, and the red rectangle symbolizing the LS with the highest weight (W), emphasizing strip quality prioritization.

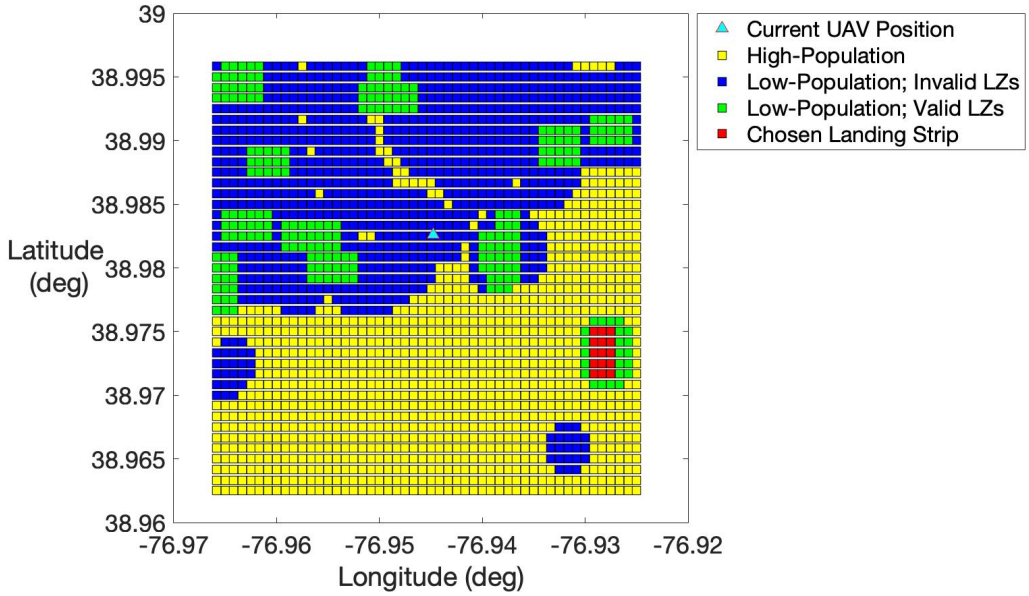


Figure 3.1: Landing Strip identification

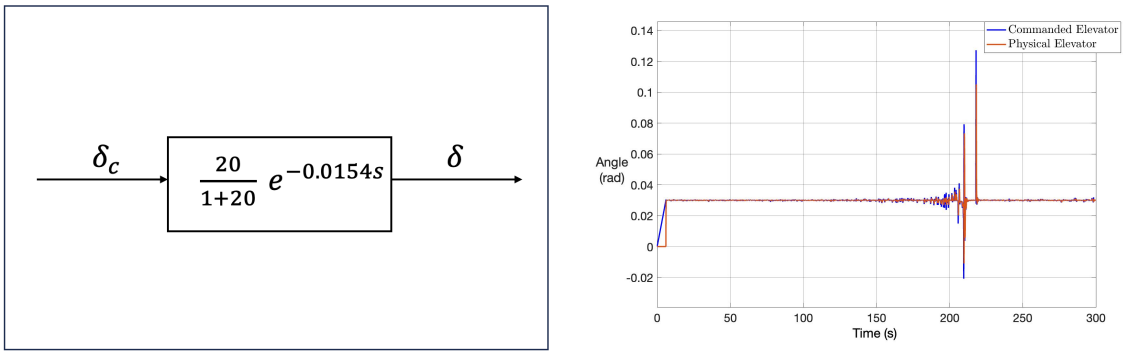
When the optimal LS is selected, a Bi-directional Rapid Exploring Random Tree algorithm (BRRT) is employed to create an exploration tree from the UAV's current position to the chosen landing site. Similar to Meng et al. algorithm [50]. The path generation algorithm is executed ten times, allowing a comparison of distances between resulting paths to select the shortest one. Additionally, a pruning algorithm is utilized to eliminate any unnecessary waypoints. This algorithm attempts to directly connect two points while adhering to motion constraints and avoiding areas with dense vegetation and buildings.

3.3 Emergency Path Planning with Faults

Our previous work focused on finding the optimal LS to ensure the safety of the aircraft and the people around it. However, in the presence of a fault that significantly impacts the dynamics of the system, certain zones may become inaccessible for the aircraft. This process begins by investigating how specific control surface faults alter the flight path and result in movement restrictions. Subsequently, out-of-reach zones are determined for each fault category and integrated with the LS identification algorithm. Finally, the path planning approach is adjusted to account for the limitations posed by each jammed control surface.

3.3.1 Fault Modeling

In order to simulate how a fault would impact the flight path within the simulation software, a fault modeling block was integrated. To achieve this, an enhancement was made to the aircraft module of the simulation, introducing a pre-defined degree of fault that persists throughout the simulation from the predetermined starting time. This approach emulates a jammed control surface scenario. For improved simulation fidelity, an actuator model was developed to determine the actual deflection of the control surfaces. This information is crucial for calculating moments and forces accurately. The actuator model used in this study was inspired by the work of Ai et al [51]. The model is illustrated in Fig. 3.2a and features a time delay of 0.0154 seconds. This model was applied to all control surfaces, including ailerons, elevator, and rudder. The response of elevator deflection to commanded deflection from the autopilot using the proposed model is depicted in Fig. 3.2.



(a) (b)

Figure 3.2: Actuator model(a) and commanded elevator deflection vs physical elevator deflection (b)

3.3.2 Landing Strip Determination

The faults experienced by the UAV can significantly impact the determination of the optimal landing strip (LS). This realization prompted an adjustment to Eq. 3.2, introducing an additional weight to account for the dynamic constraints arising from the fault. These constraints can be categorized as outlined in Table 3.2. Assuming trim faults in all control surfaces, i.e., angles set to zero degrees, for aileron faults, the criterion stipulates that any strip closer than six times the current altitude can be considered optimal, prioritizing proximity. In cases where no suitable LS falls within these limits, activating the parachute is recommended. The criteria become more intricate for aileron and rudder faults, given the inter-dependencies between these control surfaces.

Table 3.2: LS Fault Criteria

Fault	Effects	Criteria
Elevator	Restricted pitch	Discard LS farther than 6 times the current altitude
Rudder	Restricted abrupt turns	Create out of reach zones
Aileron	Restricted heading	Create out of reach zones and discard zones farther than 10 times the current altitude

Rudder: A fault in the rudder causes the aircraft to experience a yaw moment, diverting

it from its original trajectory. This deflection prompts the aircraft to fly in small ovals and execute abrupt turns, the extent of which depends on the degree of the deflection. To reorient the aircraft, the ailerons can be utilized to trace a circular path. The radius of this circle is determined by the angle at which the control surface is jammed and the airspeed velocity (v_a) of the aircraft. The velocity, which typically only contributes to forward motion in straight and level flight, becomes fragmented into x and y components upon applying a rudder deflection (δ_r), as outlined by Eq. 3.4

$$V = [v_a sin(\delta_r), v_a cos(\delta_r)]. \tag{3.4}$$

Integrating this velocity, the position can be tracked over time. These two scenarios are illustrated in Fig. 3.3 for an instance where the rudder is stuck at -5 degrees. The gray regions depict the zones inaccessible to the aircraft within the required time frame for landing. In the case of a fault with a positive deflection, the areas would be mirrored along the longitudinal axis. While these regions are still viable for aircraft access, they necessitate maneuvers that take longer to execute in order to land. In such cases, if no suitable landing strips exist outside of these areas, the aircraft can still reach them, albeit by following a longer path. The areas were calculated using flight data from autonomous missions that experienced jammed control surfaces, as well as simulations employing the method described in subsection 3.3.1.

Aileron: The scenario for an aileron fault can be explained in a manner similar to a rudder fault in terms of directional impact. However, in addition, an aileron fault can lead to a loss of altitude if the worst-case scenario is taken into account, where the control surface is stuck at the maximum angle of deflection. This reduction in altitude can be

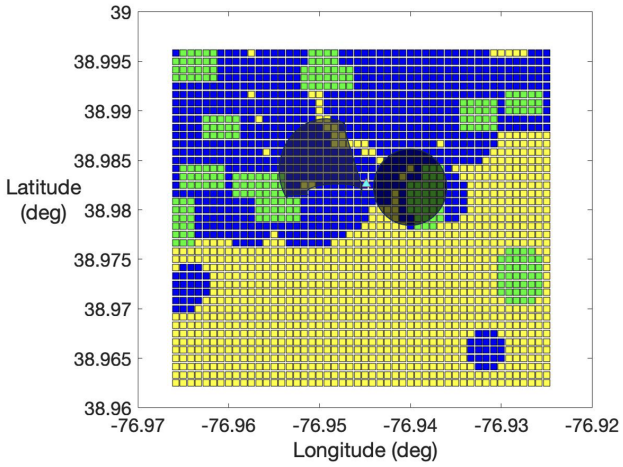


Figure 3.3: Out of reach zones for rudder and aileron fault at a negative angle

understood by examining Eq. 3.5.

$$L = C_L \frac{1}{2} \rho v^2 S. \tag{3.5}$$

Where C_L is the lift coefficient, ρ is the air density, v is velocity, and S is wing area. The area that causes the lift (S) decreases as the roll angle increases, leading to a reduction in lift. To counteract this effect, elevators are utilized to generate additional lift. However, due to these factors, it becomes necessary to impose a limitation on the distance that a landing strip should have.

3.3.3 Motion Planning

The BRRT algorithm was adapted to incorporate the out-of-reach zones when detecting aileron or rudder faults. This ensures that during path planning, any node selected within the gray areas, as seen in the example of Fig. 3.3, will be excluded. Algorithm 3 provides the pseudocode for introducing a new point to the network. The modifications are found in lines six and eleven, where besides the requirement for the new points to be valid, both

points must also lie outside of the out-of-reach zones.

Algorithm 3 BRRT

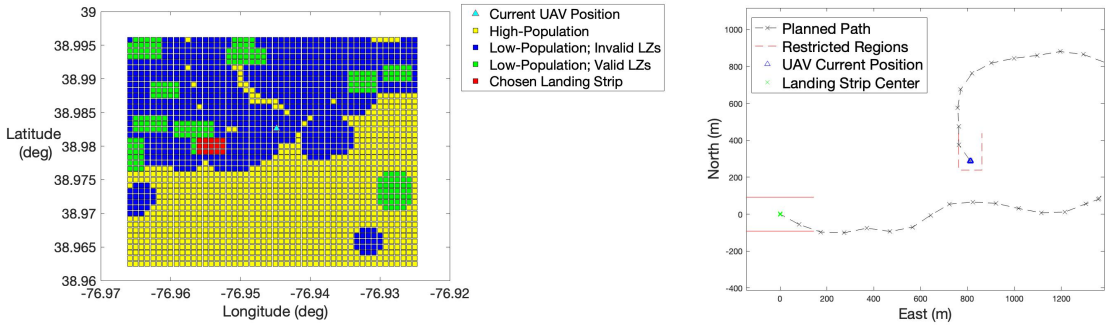
procedure A(d)dPoint
    $q_{Rand} \leftarrow$ randomly generated point
    $q_{CloseStart} \leftarrow ClosestStartPoint(q_{Rand})$
    $l_{NewStart} \leftarrow$ extend from $q_{CloseStart}$ to q_{Rand}
    $q_{NewStart} \leftarrow$ new point at end of $l_{NewStart}$
    if $q_{NewStart}$ AND $l_{NewStart}$ are valid AND out of no-go zones **then**
        add $PointStart(q_{NewStart})$
    end if
    $q_{CloseGoal} \leftarrow closestGoalNode(q_{Rand})$
    $l_{NEwGoal} \leftarrow$ extend from $q_{CloseGoal}$ to q_{Rand}
    $q_{NewGoal} \leftarrow$ new point at end of $l_{NewGoal}$
    if $q_{NewGoal}$ AND $l_{NewGoal}$ are valid AND out of no-go zones **then**
        add $PointStart(q_{NewGoal})$
    end if
end procedure

3.4 Results

Fig. 3.4a displays the optimal LS selection in the event of a negative degree rudder fault. The yellow tiles represent areas of high population, the blue tiles denote zones with low population, the green tiles mark potential LZs that fit the relaxed criteria described in section 3.2, and the red rectangle designates the optimal LS with the highest score calculated using Eq. 3.2.

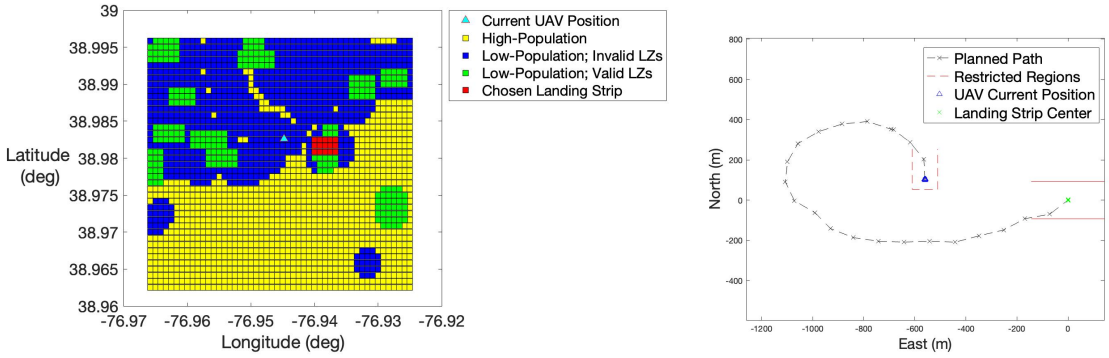
In the case of a rudder fault, the out-of-reach zones are categorized by the maximum possible deflection that the control surfaces can experience. This result is also applicable to aileron faults, where the distance between the LS and UAV falls within the criteria outlined in Table 3.2. Fig. 3.4b illustrates the BRRT path leading from the UAV's starting position to the chosen LS while avoiding the areas deemed out of reach.

Fig. 3.5a portrays the LS selection in a scenario where the elevators are jammed. In



(a) (b)

Figure 3.4: Landing Strip identification (a) and Path Planning (b) when experiencing a negative degree fault for aileron or rudder



(a) (b)

Figure 3.5: Landing Strip identification (a) and Path Planning (b) when experiencing elevator fault

this case, the distance between the LS and the airplane takes precedence. Notably, there is an additional LS depicted in this figure compared to the ones for rudder and aileron faults. This LS falls within the out-of-reach zones for the aforementioned scenario, but it is suitable for elevator faults. Fig. 3.5b illustrates the BRRT path planning from the current position of the UAV to the centroid of the selected LS.

These techniques were tested for both positive and negative aileron faults, positive and negative rudder faults, and elevator faults. In all instances, the landing strip complied with the criteria established in Table 3.2. The BRRT algorithm avoids the out-of-reach zones while adhering to the required heading constraints at the start of the path and during the

vehicle's approach for landing.

3.5 Conclusion

In this chapter, we enhanced the existing algorithm that amalgamated population and terrain data for landing zone selection, and incorporated out-of-reach zones based on control surface faults using a combination of experimental and simulation data. This contribution augments the efficacy of emergency path planning in faulty UAVs, ensuring the vehicle's ability to reach the designated landing strip.

Future endeavors could encompass expanding the scope of out-of-reach zones for path planning, factoring in elevation constraints in areas where the vehicle cannot navigate above. Additionally, the path planning algorithm has the potential to be adapted to encompass three-dimensional motion.

Chapter 4: Conclusions

During this study, heuristic-based algorithms for fault detection were developed using flight data. The Mahalanobis distance and fuzzy logic fault detection algorithms are designed to be simple enough to run on small onboard computers of UAVs. Further flight data can be gathered to refine the algorithms and enhance the detection rate for elevator faults. By combining both algorithms, a rapid fault detection method can be created, capable of identifying jammed control surfaces without relying on an aircraft model.

The dynamic limitations imposed by the detected fault on the system were integrated into an existing program used to determine safe landing zones. Each control surface fault comes with distinct criteria for defining out-of-reach zones. These zones were categorized through simulations and flight data analysis. This algorithm could be further developed for 3D path planning, incorporating high-elevation zones as no-fly areas for faulty aircraft.

Bibliography

[1] N M Barbin, S A Titov, and M Kobelev. Accidents that occurred at nuclear power plants in 1952-1972. *IOP Conference Series: Earth and Environmental Science*, 666(2):022018, mar 2021.

[2] John C. DeToledo and Merredith R. Lowe. Microwave oven injuries in patients with complex partial seizures. *Epilepsy Behavior*, 5(5):772–774, 2004.

[3] Edwin Verheijen and Jan Jabben. Effect of electric cars on traffic noise and safety. 2010.

[4] Jens Hilgert, Karina Hirsch, Torsten Bertram, and Manfred Hiller. Emergency path planning for autonomous vehicles using elastic band theory. In *Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, volume 2, pages 1390–1395 vol.2, 2003.

[5] CaiJing Xiu and Hui Chen. A behavior-based path planning for autonomous vehicle. In Honghai Liu, Han Ding, Zhenhua Xiong, and Xiangyang Zhu, editors, *Intelligent Robotics and Applications*, pages 1–9, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[6] David Madås, Mohsen Nosratinia, Mansour Keshavarz, Peter Sundström, Rolland Philippsen, Andreas Eidehall, and Karl-Magnus Dahlén. On path planning methods for automotive collision avoidance. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 931–937, 2013.

[7] Kevin W. Williams. A summary of unmanned aircraft accident/inciden data: Human factors implication. Technical report, Department of Transportation/ Office of Aerospace Medicine, 2004.

[8] Milad Ghasri and Mojtaba Maghrebi. Factors affecting unmanned aerial vehicles' safety: A post-occurrence exploratory data analysis of drones' accidents and incidents in australia. *Safety Science*, 139:105273, 2021.

[9] Christopher Lum, Kristoffer Gauksheim, Chris Deseure, Juris Vagners, and Tad McGeer. *Assessing and Estimating Risk of Operating Unmanned Aerial Systems in Populated Areas*.

[10] Eliot Rudnick-Cohen, Jeffrey W. Herrmann, and Shapour Azarm. Risk-Based Path Planning Optimization Methods for Unmanned Aerial Vehicles Over Inhabited Areas. *Journal of Computing and Information Science in Engineering*, 16(2):021004, 04 2016.

[11] Joris Guerin, Kevin Delmas, and Jérémie Guiochet. Certifying emergency landing for safe urban uav. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 55–62, 2021.

[12] O. Bektash, Jacob Naundrup Pedersen, Aitor Ramirez Gomez, and Anders la Cour-Harbo. Automated emergency landing system for drones: Safeeye project. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1056–1064, 2020.

[13] R. Schacht-Rodríguez, J. C. Ponsart, C.-D. García-Beltrán, C.-M. Astorga-Zaragoza, D. Theilliol, and Y. Zhang. Path planning generation algorithm for a class of uav multirotor based on state of health of lithium polymer battery. *J Intell Robot Syst*, 91:115–131, 2018.

[14] M. A. Kamel, K.A. Ghamry, and Y. Zhang. Real-time fault-tolerant cooperative control of multiple uavs-ugvs in the presence of actuator faults. *J Intell Robot Syst*, 88:469–480, 2017.

[15] Xiang Yu, Zhixiang Liu, and Youmin Zhang. Fault-tolerant formation control of multiple uavs in the presence of actuator faults. *International Journal of Robust and Nonlinear Control*, 26:2668 – 2685, 2016.

[16] Qian Zhang, Xueyun Wang, Xiao Xiao, and Chaoying Pei. Design of a fault detection and diagnose system for intelligent unmanned aerial vehicle navigation system. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 233(6):2170–2176, 2019.

[17] Anca Gheorghe, Ali Zolghadri, Jerome Cieslak, Philippe Goupil, Remy Dayre, and Herve Le Berre. Model-based approaches for fast and robust fault detection in an aircraft control surface servo loop: From theory to flight tests [applications of control]. *IEEE Control Systems Magazine*, 33(3):20–84, 2013.

[18] Lennon Cork and Rodney Walker. Sensor fault detection for uavs using a nonlinear dynamic model and the imm-ukf algorithm. In *2007 Information, Decision and Control*, pages 230–235, 2007.

[19] D. Henry, A. Zolghadri, J. Cieslak, and D. Efimov. A lpv approach for early fault detection in aircraft control surfaces servo-loops. *IFAC Proceedings Volumes*, 45(20):806–811, 2012.

[20] Guillermo Heredia, Fernando Caballero, Iv'e1n Maza, Luis Merino, Antidio Viguria, and An'edbal Ollero. Multi-unmanned aerial vehicle (uav) cooperative fault detection employing differential global positioning (dgps), inertial and vision sensors. *Sensors*, 9(9):7566–7579, 2009.

[21] Randal K. Douglas and Jason L. Speyer. Robust fault detection filter design. *Journal of Guidance, Control, and Dynamics*, 19(1):214–218, 1996.

[22] Sanghyuk Lee, Wookje Park, and Sikhang Jung. Fault detection of aircraft system with random forest algorithm and similarity measure. *The Scientific World Journal*, 2014, 2014.

[23] Paul Freeman, Rohit Pandita, Nisheeth Srivastava, and Gary J. Balas. Model-based and data-driven fault detection performance for a small uav. *IEEE/ASME Transactions on mechatronics*, 18(4):1300–1309, 2013.

[24] Matteo Corbetta, Katelyn J Jarvis, and Stefan Schuet. Hybrid modeling of unmanned aerial vehicle electric powertrain for fault detection and diagnostics. In *AIAA AVIATION 2023 Forum*, page 3861, 2023.

[25] Prasanta C. Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Science of India*, 2:49–55, 1936.

[26] Kyung Ho Park, Eunji Park, and Huy Kang Kim. Unsupervised fault detection on unmanned aerial vehicles: Encoding and thresholding approach. *Sensors*, 21(6), 2021.

[27] Raz Lin, Eliyahu Khalastchi, and Gal A. Kaminka. Detecting anomalies in unmanned vehicles using the mahalanobis distance. In *2010 IEEE International Conference on Robotics and Automation*, pages 3038–3044, 2010.

[28] Tekla S. Perry. Lotfi zadeh and the birth of fuzzy logic. *IEEE Spectrum*, 32(6):32–35, 1995.

[29] Shuma Adhikari, Nidul Sinha, and Thingam Dorendrajit. Fuzzy logic based on-line fault detection and classification in transmission line. *SpringerPlus*, 5(1002), 2016.

[30] Newton Maruyama, Mourad Benouarets, and Arthur L. Dexter. Fuzzy model-based fault detection and diagnosis. *IFAC Proceedings Volumes*, 29(1):6441–6446, 1996. 13th World Congress of IFAC, 1996, San Francisco USA, 30 June - 5 July.

[31] Galina Demidova, Anton Rassõlkin, Toomas Vaimann, Ants Kallaste, Janis Zakis, and Aleksandrs Suzdalenko. An overview of fuzzy logic approaches for fault diagnosis in energy conversion devices. In *2021 28th International Workshop on Electric Drives: Improving Reliability of Electric Drives (IWED)*, pages 1–7, 2021.

[32] Tao Dong, XH Liao, Ran Zhang, Zhao Sun, and YD Song. Path tracking and obstacles avoidance of uavs-fuzzy logic approach. In *The 14th IEEE International Conference on Fuzzy Systems, 2005. FUZZ'05.*, pages 43–48. IEEE, 2005.

[33] Yun Hong Gao, Ding Zhao, and Yi Bo Li. Uav sensor fault diagnosis technology: A survey. *Applied Mechanics and Materials*, 220:1833–1837, 2012.

[34] Ahmed Taimour Hafez and Mohamed A. Kamel. Fault-tolerant control for cooperative unmanned aerial vehicles formation via fuzzy logic. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1261–1266, 2016.

[35] Md Shah Alam and Jared Oluoch. A survey of safe landing zone detection techniques for autonomous unmanned aerial vehicles (uavs). *Expert Systems with Applications*, 179:115091, 2021.

[36] Sebastian Scherer, Lyle Chamberlain, and Sanjiv Singh. Autonomous landing at unprepared sites by a full-scale helicopter. 60(12), 2012.

[37] Marc Schmitt and Peter Stütz. Multi-uav based helicopter landing zone reconnaissance: Information level fusion and decision support. In *Engineering Psychology and Cognitive Ergonomics: Cognition and Design: 14th International Conference, EPCE 2017, Held as Part of HCI International 2017, Vancouver, BC, Canada, July 9-14, 2017, Proceedings, Part II 14*, pages 266–283. Springer, 2017.

[38] Efstratios Kakaletsis and Nikos Nikolaidis. Potential uav landing sites detection through digital elevation models analysis. *arXiv preprint arXiv:2107.06921*, 2021.

[39] Sarah Lane, Zsolt Kira, Ryan James, Domenic Carr, and Grady Tuell. Landing zone identification for autonomous uav applications using fused hyperspectral imagery and lidar point clouds. In *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XXIV*, volume 10644, pages 106–117. SPIE, 2018.

[40] Gennaro Ariante, Salvatore Ponte, Umberto Papa, and Giuseppe Del Core. Safe landing area determination (slad) for unmanned aircraft systems by using rotary lidar. In *2021 IEEE 8th International Workshop on Metrology for AeroSpace (MetroAeroSpace)*, pages 110–115. IEEE, 2021.

[41] Andrew E. Johnson, Allan R. Klumpp, James B. Collier, and Aron A. Wolf. Lidar-based hazard avoidance for safe landing on mars. *Journal of Guidance, Control, and Dynamics*, 25(6):1091–1099, 2002.

[42] Edward Carney, Lina Castano, and Huan Xu. Determination of safe landing zones for an autonomous uas using elevation and population density data. *AIAA Scitech 2019 Forum*.

[43] Abbas Chamseddine, Youmin Zhang, and Camille A. Rabbath. Trajectory planning and re-planning for fault tolerant formation flight control of quadrotor unmanned aerial vehicles. In *2012 American Control Conference (ACC)*, 2012.

[44] Kewei Xia, Wonmo Chung, and Hungsun Son. Dynamics estimator based robust fault-tolerant control for vtol uavs trajectory tracking. *Mechanical Systems and Signal Processing*, 162:108062, 2022.

[45] Reza Radmanesh, Manish Kumar, Donald French, and David Casbeer. Towards a pde-based large-scale decentralized solution for path planning of uavs in shared airspace. *Aerospace Science and Technology*, 105:105965, 2020.

[46] Yaohong Qu, Yintao Zhang, and Youmin Zhang. Global path planning algorithm for fixed-wing uavs. *J Intell Robot Syst*, 92:691–707, 2018.

[47] Bulent Ayhan, Chiman Kwan, Bence Budavari, Jude Larkin, and David Gribben. Pre-flight contingency planning approach for fixed wing uavs with engine failure in the presence of winds. *Sensors*, 19(2), 2019.

[48] Leticia O. Rojas-Perez, Roberto Munguia-Silva, and Jose Martinez-Carranza. Real-time landing zone detection for uavs using single aerial images. In *10th International Micro Air Vehicle Competition and Conference, Melbourne, Australia*, pages 243–248, 2018.

[49] Bo Jiang, Zhonghui Chen, Jintao Tan, Roukun Qu, Chenglong Li, and Yadong Li. A real-time semantic segmentation method based on stdc-ct for recognizing uav emergency landing zones. *Sensors*, 23(14):6514, 2023.

[50] Li Meng, Song Qing, and Zhao Qin Jun. Uav path re-planning based on improved bidirectional rrt algorithm in dynamic environment. In *2017 3rd International Conference on Control, Automation and Robotics (ICCAR)*, pages 658–661, 2017.

[51] Bing Ai, Luis Sentis, Nicholas Paine, Song Han, Aloysius Mok, and Chien-Liang Fok. Stability and performance analysis of time-delayed actuator control systems. *Journal of Dynamic Systems, Measurements, and Control*, 138, 2-16.