A Fault Diagnosis Framework for MAVLink-Enabled UAVs Using Structural Analysis

Georgios Zogopoulos-Papaliakos, Michalis Logothetis and Kostas J. Kyriakopoulos

***Abstract*— MAVLink is a popular message protocol for small Unmanned Aerial Vehicles (UAVs). In this work, we present a Fault Detection and Isolation (FDI) framework for fixed-wing UAVs which takes advantage of the information conveyed in MAVLink telemetry streams and produces a bank of residual generators. Structural Analysis is employed to systematically handle the varying set of available measurements, identify the observable faults and adjust the FDI system accordingly. Structural detectability and isolability analyses are carried out. A case-study on a real-life telemetry log of a UAV crash demonstrates the efficacy of the proposed approach.**

I. Introduction

As Unmanned Aerial Vehicles (UAVs) become an established technology and provide a platform for other aspects of research, their number grows steadily. Micro Air Vehicles (MAVs), being handier and low-cost, are by far the most populous category. A few autopilot software suites have been established as the go-to options for attitude and trajectory control of MAVs, within academic, research and small business sectors. More often than not, one of the PX4 [1], ArduPilot [2] or Paparazzi [3] software is used.

On the other hand, while Fault Detection and Isolation (FDI) is deemed a topic of utmost importance in the roadmap for the integration of UAVs in the national air space [4], it has not been populated with solutions which systematically cover a wide class of its problems. Admittedly, UAV literature has inherited the theoretical advances on fault diagnosis from manned aviation, but such approaches are oriented towards tailor-made systems, requiring significant investments in time and money for integration and tuning and often have substantial requirements in hardware [5]–[7].

These approaches are in stark contrast to the philosophy of Micro Air Vehicles (MAV), which are designed to have fast design iteration cycles and be cheap and expendable. In these cases, most traditional fault diagnosis systems come short in the aforementioned aspects.

Some of the more modern approaches to Fault Diagnosis have utilized the telemetry stream that almost every MAV emits. In [8], the measurements of airspeed, angular rates and received control inputs have been used to detect faults on the airspeed probe and control surfaces, through direct comparison, parameter estimation and statistical processing. More recently, in [9], maximum likelihood methods and neural networks were applied on raw engine telemetry data to perform fault diagnosis.

The authors are with Control Systems Lab, School of Mechanical Engineering, National Technical University of Athens, Greece {gzogop, logothm, kkyria}@mail.ntua.gr

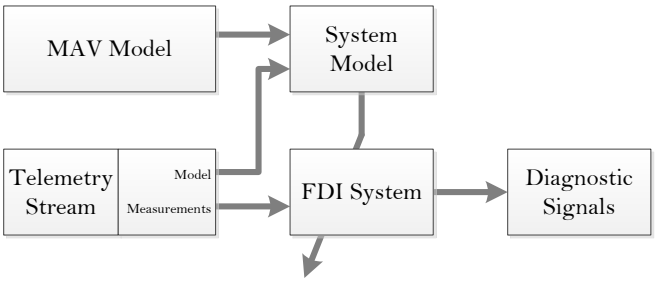

Fig. 1. Diagnostic Signals Generation Overview

Nevertheless, the above works do not take full advantage of the existing underlying infrastructure and information provided by MAVs. Notably, one of the most significant and modern infrastructure is the MAVLink message protocol [10]. MAVLink is an open-source, rigid specification for message structure and content, tailor-made for the needs of MAVs in telemetry, command and information exchange. It was met with large acceptance and is now used exclusively or at least supported by all the open-source autopilots and many of the closed-source ones. To our best knowledge, no previous work has explored the diagnostic potential of the MAVLink protocol.

MAVLink specifies a very large set of measured quantities, in the order of hundreds. Obviously, manually designing a diagnostic system around the arbitrary subset of measurements each MAV emits is infeasible. Parity-based fault diagnosis [11] offers appropriate methods to handle systems with high numbers of equations, inputs and outputs; a system model is explored systematically for redundant equations which form residual generator functions, sensitive to faults.

One such methodology, Structural Analysis (SA) [12], [13], is especially suitable for non-linear systems. It is an abstraction methodology which converts the mathematical model of a system into a graph, called Structural Model. Upon that, graph-based algorithms are applied for the exploration of diagnostic possibilities and the population of the residual generators bank.

Structural Analysis has been used in the past in Fault Diagnosis for manned and unmanned aircraft [14]–[16], although without making an effort to utilize existing telemetry information. Instead, either overly limited or unrealistic measurements were considered in those works.

The major contributions of this work are a) a framework which populates a bank of diagnostic residual generators based on available MAVLink telemetry and b) an examination of the MAVLink protocol regarding its suitability for Parity-based diagnosis.

II. System Modeling

The system model is comprised by two separate parts. The first is a fixed model of the MAV, which should be representative of a large class of MAVs to ensure reusability. For demonstration purposes, we choose to work with a fixed-wing MAV, balanced between practicality and usefulness, but a multirotor model would be equally applicable. The second is the varying set of measurement equations, as implied by the MAVLink telemetry.

A. Fixed-Wing MAV Model

As discussed above, the more generic the employed model is, the greater class of different MAVs it can cover. However, a balance should be struck between a generality and detail, because the latter provides diagnostic information. Incorporating model parameters is certain to make the model applicable to only one aircraft. The only constraint is that any subset of equations must either be a set of Analytical Equations (AE) or a set of semi-explicit, Index-1 Differential-Algebraic Equations (DAE). This is to ensure that it can be automatically solved by typical numerical methods. In our previous publication [16], criteria which guarantee this requirement were given. One such possible, albeit short, model, used in this work, is presented in Table I and is subsequently discussed upon.

Before proceeding, however, let us cover the variable notation, which loosely follows the MAVLink nomenclature: *p_n, p_e, p_d, v_n, v_e, v_d* are the position and speed of the MAV in the NED coordinate system. *climb, v_wind_hor, dir_wind, v_w_n, v_w_e* represent the climb rate and the wind magnitude, direction, north-east components. *cog, v_ground, v_air* are the course-over-ground, groundspeed and airspeed. *vcc, v_servo, voltage_battery, current_battery* refer to the autopilot board and servo motor rail voltage, battery voltage and current. *roll, pitch, yaw* are the Euler angles. *alt_agl* is the altitude above ground level. *eph, epv* are the horizontal and vertical dilution of precision of the GPS sensor. *load, freemem, drop_rate_comm* is the percent load of the autopilot processor, the unallocated RAM and the communications channel drop rate. $h\_\cdot$ are the components of the magnetic field of the Earth. *error_xtrack, error_rp_dcm error_yaw_dcm* are the L1 controller cross-track error and Euler angles estimation errors of the autopilot. m_2 is Bernoulli's equation. k_8 is the wind triangle equation [8].

The model is constructed with maximum fault isolation but also simplicity in mind; e.g. aerodynamics are excluded, since they require the knowledge of the aerodynamic coefficients, unavailable in most MAVs.

Kinematic equations are the most common and first to be introduced k_1-k_5, along with wind-triangle equations k_6-k_8 [8]. The dynamic pressure equation is also included (m_2). The equations propagating the Euler angles are not included because it has been observed that they do not increase the diagnostic capabilities of this system.

Limit-checking equations can also be incorporated. To that goal, we introduce the lim function notation as follows:

$$lim(x, x_{min}, x_{max}) = max(max(x_{min} - x, 0), max(x - x_{max}, 0)) \quad (1)$$

The equation $lim(x, x_{min}, x_{max}) = 0$ holds when the value of x is within its specified limits. In case of discrepancy, it no longer holds and any residual using this equation will trigger. l_1-l_{17} are limit-checking equations, populated with variables commonly measured by MAVLink-enabled autopilots.

Finally, equations involving error variables (such as a controller error) and implicitly equivalent variables (such as desired and actual regulated state variables) can also be added in this model ($e_1 - e_8$).

TABLE I
Example Fixed-Wing MAV Model

Label	Constraint
k_1	$climb = -v\_d$
k_2	$v\_ground = \sqrt{v\_n^2 + v\_e^2}$
k_3	$\dot{p\_n} = v\_n$
k_4	$\dot{p\_e} = v\_e$
k_5	$\tan(cog) = v\_e/v\_n$
k_6	$v\_w\_n = v\_wind\_hor * \cos(dir\_wind)$
k_7	$v\_w\_e = v\_wind\_hor * \sin(dir\_wind)$
k_8	$v\_air = sqrt(v\_ground^2 + v\_wind\_hor^2 - 2 * v\_ground * v\_wind\_hor * \cos(dir\_wind - cog))$
l_1	$lim(load, 0, load\_max)$
l_2	$lim(voltage\_battery, voltage\_battery\_min, \infty)$
l_3	$lim(current\_battery, 0, current\_battery\_max)$
l_4	$lim(drop\_rate\_comm, 0, drop\_rate\_comm\_max)$
l_5	$lim(eph, 0, eph\_max)$
l_6	$lim(epv, 0, epv\_max)$
l_7	$lim(alt\_agl, alt\_agl\_min, alt\_agl\_max)$
l_{10}	$lim(vcc, vcc\_min, vcc\_max)$
l_{11}	$lim(v\_servo, v\_servo\_min, v\_servo\_max)$
l_{12}	$lim(freemem, freemem\_min, freemem\_max)$
l_{13}	$lim(vibration\_x, 0, vibration\_x\_max)$
l_{14}	$lim(vibration\_y, 0, vibration\_y\_max)$
l_{15}	$lim(vibration\_z, 0, vibration\_z\_max)$
l_{16}	$lim(v\_air, v\_air\_min, v\_air\_max)$
l_{17}	$lim(error\_v\_air, error\_v\_air\_min, error\_v\_air\_max)$
m_1	$h\_total = sqrt(h\_x^2 + h\_y^2 + h\_z^2)$
m_2	$press\_diff = 0.5 * \rho * v\_air^2$
e_1	$roll = roll\_c$
e_2	$pitch = pitch\_c$
e_3	$yaw = yaw\_c$
e_4	$error\_alt = 0$
e_5	$error\_v\_air = 0$
e_6	$error\_xtrack = 0$
e_7	$error\_rp\_dcm = 0$
e_8	$error\_yaw\_dcm = 0$

B. Measurement Model

The second part of the system model is comprised of measurement equations which are incurred by the MAVLink messages present in each particular application. Different MAVs may emit different subsets of the MAVLink message set. The same may hold for the a MAV under different modes of operation. The set of measurement equations which end up in the system model varies accordingly and the ability of Structural Analysis to seamlessly adapt around this variation is one of its significant advantages.

Each measurement equation is assumed to be of the form:

$$s_i : \quad x = x_m \quad (2)$$

where x is a variable of the MAV model and x_m the instantaneous reported value. x_m may be a raw measurement from a hardware sensor, but may also very well represent a tracking error of an internal controller or the output of a Kalman filter.

C. Fault Modeling

No specific fault modeling is required to apply SA methodologies. This is inline with other SA works [13], [17]; the contribution of any fault type (additive, multiplicative, parametric, etc) can be represented as an additive one. A multiplicative measurement fault can be written as:

$$y = (1 + f)x = x + fx = x + f_x \tag{3}$$

Since SA is concerned only with the structural form of the equations, merely the membership of the fault variable f in the equation is required to build the structural model. For simplicity, one can assign an additive fault in the equation.

No fault is attributed to the ideal MAV model of Table I, since it is not possible for equations modeling kinematics, atmosphere, etc, to stop being true. It is the values they are fed that are incompatible in case of a fault. This reasoning also applies to limit- and error-checking equations: for example, equality between an ideal error variable and zero never stops to hold, but the actual measured variable value doesn't satisfy the equation in the presence of fault.

Thus, all faults are instead attributed to measurement equations. This also allows grouping the faults into subsystems, which are usually coupled tightly together and cannot be further isolated anyway.

III. Structural Analysis

In the previous section, a system model was constructed, subject to a set of faults. The extraction of residual generators from it centers around the discovery of analytically redundant subsets of equations. These contain alternate paths of calculation for certain variables, leading to the construction of parity relations and providing opportunities for internal cross-checks.

However, manually parsing the large, non-linear system model in search of residual generators is effectively intractable. More importantly, since not all MAVs generate the same set of MAVLink messages, a pre-compiled, fixed residual generator bank is not a viable approach. A process is required which accepts the generic aircraft model, along with the available subset of MAVLink messages, as input and automatically returns *Analytical Redundancy Relations* (ARRs), which form residual generators (Fig. 1).

Structural Analysis is suitable for this purpose [12], [13], [18]. Contrary to traditional Parity-space formulations for linear systems [19], it is an abstraction methodology. It converts the mathematical model of a system into a qualitative bipartite graph called Structural Graph (SG), which describes whether there exist relations between model equations and model variables. Although this graph contains less information than the original model, it is a form suitable for processing by automated, graph-theoretic algorithms.

More information on the premises of Structural Analysis can be found in [13] while in our past work [16] emphasis has been placed on its application in fixed-wing UAV models.

A. Basic Definitions

The system model of the previous section forms the equation set $\mathcal{C}$ with elements c_i. The set of included unknown variables is $\mathcal{X}$ with elements x_j. Known variables (constants and measurements) are not considered, without loss of generality. We denote the set of variables which appear in c_i as $var(c_i)$ and the set of equations which include x_j as $eqs(x_j)$.

The SG is generated as a bipartite graph $\mathbf{G} = (\mathcal{C}, \mathcal{X}, \mathcal{E})$, whose edge set $\mathcal{E}$ connects each variable vertex to the equations it participates into according to the rule

$$e_i = (c_j, x_k) \in \mathcal{E} \Leftrightarrow x_k \in var(c_j) \tag{4}$$

According to the Dulmage-Mendelsohn decomposition [20], any bipartite graph $\mathbf{G}$ can be uniquely partitioned to three (potentially empty) subgraphs, $\mathbf{G}^-$,$\mathbf{G}^0$ and $\mathbf{G}^+$, named under-, just- and over-constrained part. For each part it holds that $|\mathcal{C}^-| < |\mathcal{X}^-|$, $|\mathcal{C}^0| = |\mathcal{X}^0|$, and $|\mathcal{C}^+| > |\mathcal{X}^+|$ respectively. Explicitly, there are more variables than equations in the underconstrained part, equal in the just-constrained part and less in the underconstrained part.

This leads to an important result: faults can be diagnosed (from a structural sense) only for equations belonging to the over-constrained part of a system. This decomposition is a core tool of Structural Analysis, because it efficiently reduces the search scope for analytical redundancy. Depending on the available measurements, the equation set comprising $\mathbf{G}^+$ varies. Thus, faults which end up in $\mathbf{G}^+$ are detectable, while the rest are not.

If for a subgraph $\mathbf{G}_i$ holds that $\mathbf{G}_i$=$\mathbf{G}_i^+$, then it is *Proper Structurally Overdetermined* (PSO) [13], [21]. In a PSO there are more equations than variables; in a structural sense, it is possible to calculate all the unknown variables from the available equations, with one or more equations remaining unused. The remaining equations have all of their variables known and can be used as residual generators.

B. Disconnected Subgraphs

Given an overdetermined structural graph $\mathbf{G}^+$, structural redundancy is defined as $\phi = |\mathcal{C}| - |\mathcal{X}|$ [21]. Let $\mathbf{G}^+$ be subject to n_f faults; The upper limit of the number of candidate residual generators is [21]

$$n_{rg} = \sum_{k=1}^{\phi-1} \binom{n_f}{k} \tag{5}$$

For a graph with a high structural redundancy and number of faults, the number of potential residual generators quickly becomes intractable. However, the situation is a lot more favorable if $\mathbf{G}^+$ contains disconnected components. Let

$$\mathbf{G}^+ = \bigcup_i \mathbf{G}_i^+, \quad i = 1, ..., n \tag{6}$$

where $\mathbf{G}_i^+$ are disconnected subgraphs:

$$\mathbf{G}_i^+ \bigcap \mathbf{G}_j^+ = \emptyset, \quad i \neq j \tag{7}$$

Then, the upper limit of the number of potential residual generators is

$$n'_{rg} = \sum_i n_{rg,i} = \sum_i \left(\sum_{k=1}^{\phi_i} \binom{n_{f,i}}{k} \right) \leq n_{rg} \tag{8}$$

because

$$\phi = \sum \phi_i, \quad \phi_i \geq 1 \tag{9}$$

$$n_f = \sum n_{f,i}, \quad n_{f,i} \geq 1 \tag{10}$$

This will play an important role, as we will see in our case study.

C. Graph Matchings

The final step for residual generation is the *matching* procedure. Formally, a graph matching $\mathcal{M}$ is an edge set, subset of $\mathcal{E}$ such that $\mathcal{M}=\{m_i=(c_i,x_i)\in\mathcal{E} \; | m_i \neq m_j$ iff $c_i \neq c_j \wedge x_i \neq x_j, \forall i,j\}$. In other words, a set of edges such, that any two edges do not have a variable or an equation in common. Efficient matching algorithms have been known to exist for many years [22]–[24].

A matching $\mathcal{M}$ extracted from a PSO $\mathbf{G}_i^+$ is a non-unique pairing between equations and variables, such that every unknown variable in $\mathcal{X}_i^+$ is assigned to be solved by one equation of $\mathcal{C}_i^+$, ensuring that each equation will be used only once. The remaining uncovered equations are available as ARRs. ■

A known drawback of SA is that it abstracts away any quantitative information of the given model and hence cannot provide estimates regarding the actual fault sensitivity and robustness of the residual signal. These estimates can be readily accessed by further analysis in linear systems [25], but for the general non-linear case there are only a few related works [26]. In a parallel work [27], we tackle the problem of sensitivity and robustness of nonlinear residual generators using numerical methods.

IV. The MAVLink Protocol

MAVLink [10] is a message protocol, designed to be suitable for communications between MAVs and their Ground Control Station (GCS). It is perhaps the most widely used message protocol in Unmanned Aerial Systems (UAS). Systems of commercial, non-commercial and academic nature employ MAVLink as a platform to exchange remote commands, mission descriptions and telemetry data among a Ground Control Station (GCS), a MAV and its peripheral payloads. PX4 and ArduPilot use it exclusively, while Paparazzi has support for it.

MAVLink telemetry messages consist the source of the measured information, which was modeled as measurement equations in Section II. Each message is a variable-length packet, consisting of a header, payload and two checksum bytes. Each message is defined according to a rigid set of *message definitions*, fixed for each protocol version. These definitions assign each message ID to a message type and fully describe its contents and size.

As an example, the *sys_status* message is assigned the ID 1. Some of its fields are *voltage_battery*, a 16-bit unsigned integer, expressing the MAV battery voltage in mV and the *onboard_control_sensors_health*, a 32-bit bitmask showing which onboard controllers and sensors are operational.

TABLE II
Applicable MAVLink Message IDs

Message IDs
1, 2, 24, 27, 29, 30, 32, 33, 35, 36, 62, 65, 74, 116, 125, 150, 152, 163, 165, 168, 174, 178, 182, 193, 241

A. Message classification

The payload of each message type essentially contains a set of quantities (also termed fields). For the rest of this work, a specific field of a specific message will be referred to with the <*message_ID/field_name*> naming scheme. For reasons that will become apparent at the end of this section, we attempt to classify the entire set of messages.

1) Time Domain: Each message may represent an event or can be emitted quasi-continuously. Commands from the GCS and mission status updates are examples of event messages, while raw sensor measurements and controller outputs are of continuous-time. Typical telemetry rates for continuous-time quantities are 4, 2 and 1Hz.

2) Value Domain: Each field may represent a quantity whose domain of values is qualitative and discrete (e.g. the autopilot mode) or continuous (e.g. barometric altitude).

Structural Analysis belongs to Parity-Based Fault Diagnosis, which is traditionally applicable to quantities which represent continuous variables in continuous time [11]. This becomes apparent by noting that both the MAV model of II-A and the measurement model of II-B involve continuous functions in the real domain.

Consequently, we are only interested in message fields which satisfy these conditions. For this work, we use the *common* MAVLink message set, along with the *ardupilotmega* extension [28], as emitted by the ArduPilot/ArduPlane v.3.5.2 software. The list of message IDs which can be seen in Table II are applicable for fault diagnosis and used.

Due to lack of space and the sheer number of messages, the interested reader can refer to the Message Definitions documentation, found online [10].

V. Case Study

A. Detectability and Isolability Analyses

In total, the overall system model is comprised of 116 equations and 237 variables and is subject to 77 faults. We proceed to extract the structural model through Structural Analysis. To that goal, we employed our own MATLAB-based software, which can be found at `http://georacer.github.io/fault-diagnosis/demos/mavlink/mavlink.html`. Some characteristic performance numbers are given in Table III. These results were produced offline with an Intel Core i7-6500 CPU @ 2.5GHz and 8GB of RAM.

It should be noted that the original structural graph had a high-level of structural redundancy (ϕ=59), which would result in a maximal number of possible PSOs with faults in the order of 9.4×10^{21}. Parsing this many potential PSOs would make the problem computationally intractable.

TABLE III

STRUCTURAL ANALYSIS PERFORMANCE CHARACTERISTICS

Metric	Value
Initial Structural Graph generation time	0.7 s
Non-connected subgraph generation time	24.9 s
PSO discovery time (using MTES [21])	20.9 s
Valid matchings discovery time	13.1 s
Generation of calculable residual generators time	103.6 s
Total number of residual generators	108

TABLE IV

NON-ISOLABLE FAULT GROUPS

Group No	Faulty MAVLink Message ID/Field
1	168/direction, 168/speed, 24/cog
2	30/rollspeed, 116/xgyro
3	30/pitchspeed, 116/ygyro
4	30/yawspeed, 116/zgyro
5	116/xmag, 116/ymag, 116/zmag

Thankfully, the original problem was able to be broken down into 34 non-connected subgraphs, containing 7952 PSOs at maximum, which were parsed in a relatively very short time. The Minimum Test Equation Support (MTES) algorithm [21] was used for the discovery of PSOs from each SG.

The separation of the initial graph into subgraphs is considerably expensive, and so is the discovery of PSOs, even in the fragmented subgraphs. The matching procedure is less time-consuming. The implementation of the resulting residual calculation sequences into software functions is the greatest hurdle. The procedure initially calls Matlab's symbolic solver on the original analytical expressions; symbolic solvers are known to be slow. If an analytical solution for the residual is not possible, then numerical solver tools are automatically employed.

In total 108 residual generators were extracted. Parsing the SG, each residual generator results in a fully-ordered sequence of function evaluations, each of which is implemented as numerical function. This minimizes the residual evaluation time, allowing them to be used in real-time.

Out of the initial 77 faults, only 6 belong in the non-detectable part of the SG ($\mathbf{G}^{-} \bigcup \mathbf{G}^{0}$). All PSO matchings were able to be converted into a calculable residual generator. The resulting Isolability Matrix [13], [29] can be seen in Fig. 2. Under the single-fault assumption, most faults can be isolated from the rest, as witnessed by the singular König-Hall components of the matrix. Table IV presents the groups of non-isolable faults, which correspond to the 5 non-singular components. However, this isolability analysis is overly optimistic, since in such tightly-coupled systems such as UAVs the manifestation of a fault may quickly cause more faults to appear.

B. Accident Investigation Example

Contrary to General Aviation incidents [30]–[32], where human observers can report on the accident as experienced first-hand, many UAV flights are flown "blind". That is, there is no visual nor auditory feedback from the MAV to the pilot, nor any other biological sensory signal. The state of the vehicle is available only as telemetry and logs.

In a typical UAV accident investigation, the investigator parses all available measurements time series, locates abnormalities, sorts them temporally and creates a causal sequence of events. A thorough analysis requires a great amount of time, especially in the presence of ample data.

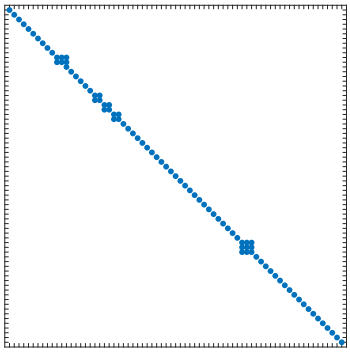

Fig. 2. Isolability Matrix

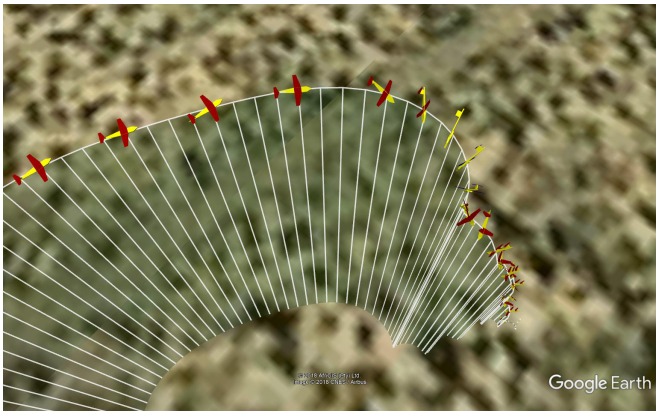

Fig. 3. The Unfortunate Conclusion of the Mapping Mission

This is an ideal application of the previously generated bank of residual signals. The usefulness of the presented approach is highlighted by a real-world example.

In October of 2016, the Remote Sensing Laboratory of NTU Athens flew a mapping mission in South Africa, using a 2m fixed-wing MAV. The MAV was controlled by a Pixhawk autopilot with ArduPilot ArduPlane v3.5.2. Mid-way through the automated mission the MAV suddenly spiraled down and crashed without warning. A visual representation of the trajectory of the MAV can be seen in Fig. 3.

Fig. 4 is the Fault Grid corresponding to the last few seconds of flight, a stack of boolean time series of candidate faults. All measurements provided by the MAVLink messages were resampled and aligned at 1 Hz. The residual signals were generated and compared against static thresholds. For those residuals which exceeded their thresholds, the corresponding faults are plotted in the Fault Grid on the y-axis. In other words, at each time sample t_k, the fault f_l is marked with a rectangular interval if f_l is part of the fault signature of any residual triggered at time t_k.

By observing the evolution of candidate fault occurrences in conjunction to time, the investigator can acquire a holistic view of fault occurrences, before and after the accident.

Loss of control occurred at t=322 s. Even before the moment of failure, triggered residuals point to a few faults: 24/lon, 30/yawspeed, 33/lon, 116/zgyro, 182/lat and 182/lng

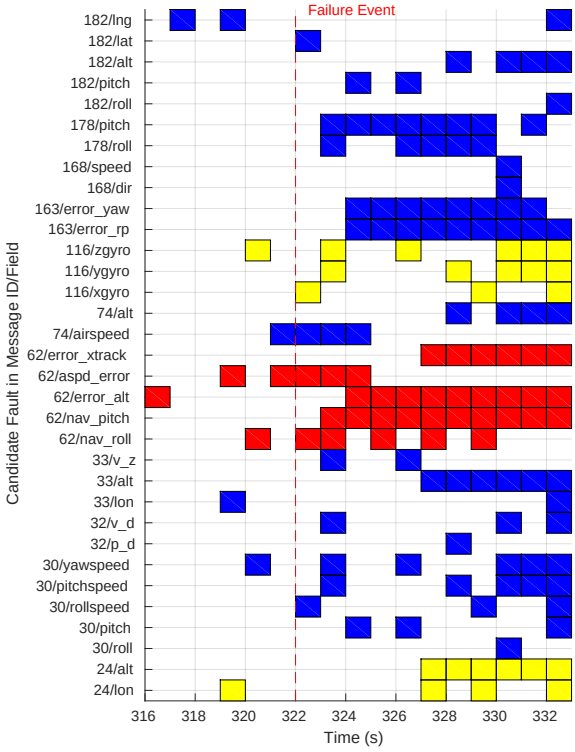

Fig. 4. The Fault Occurrence Grid: Yellow - Raw sensor faults. Blue - Navigation faults. Red - Autopilot faults

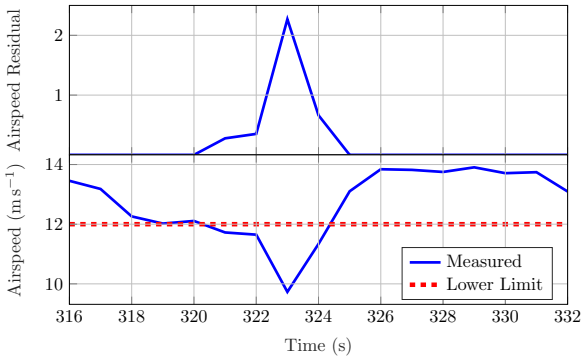

Fig. 5. Airspeed-Related Time-Series

are probably caused by inconsistencies between the direct measurements of the sensors related to the navigation system and the 3 different navigation algorithms which ArduPilot runs simultaneously. More concerning is the departure from the airspeed envelope and the airspeed error, occurring at time t=321 s-325 s, as witnessed by the faults 62/airspeed_error and 74/airspeed. Plotting the actual airspeed data series (Fig. 5) we see a significant drop at that interval.

Some moments later (t=322 s-324 s), alarms for faults related to state control are raised (62/nav_pitch, 62/error_alt, 32/nav_roll). Indeed, plotting the roll, pitch and altitude time series, we clearly see that the aircraft has by now entered a downward spiral, leading to the crash. Even though the airspeed recovered a few seconds after the initial stall, the autopilot was not able (nor programmed) to perform the maneuver required to exit the spiral.

Interestingly, shortly after the failure and during the downward spiral, a lot of residuals related to faults of the sensor and navigation subsystems triggered (33/alt, 163/error_rp, 163/error_yaw, 178/roll, 178/pitch). This can be attributed to the separate state estimation filters under-performing because of the rapid rotation and producing diverging estimates.

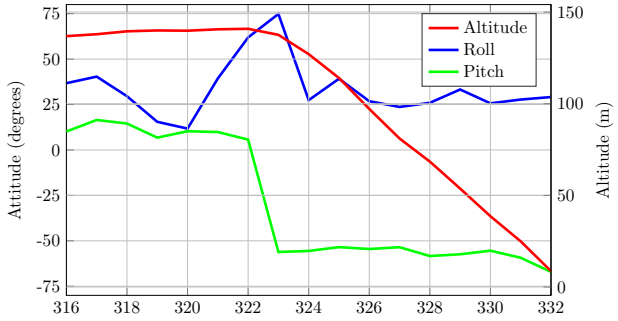

Fig. 6. Attitude and Altitude Time Series During the Crash

Regarding the root cause of the accident, we can safely rule out airspeed estimation, since the related residuals did not trigger. Instead, the first severe failure was the inability of the autopilot to maintain the airspeed error within a +-3m/s bound. This allowed the airspeed to drop below its prescribed envelope, resulting in a tip-stall. Past that point, the autopilot was no longer able to control the state of the aircraft.

The inability of the autopilot to regulate airspeed should probably not be attributed to an external fault. Instead, it is much more likely that the kinetic energy control loop was badly tuned, especially taking into account the high Mean-Sea-Level altitude the aircraft was flying at. However, further investigation would require analysis of the internal structural of the controller architecture and implementation, and such information is not conveyed by the MAVLink stream.

The presented analysis was applied on a telemetry log, but it is also applicable on real-time telemetry streams. The costly procedure of residual generators implementation can be carried out offline, while their evaluation into residual signals can take place online and even on-board. The residual thresholds (static or adaptive) should be available beforehand however, since SA does not provide them.

VI. Conclusions

In this work we presented a Fault Diagnosis framework for fixed-wing UAVs which emit telemetry in MAVLink format. The large information content of the telemetry stream was handled and exploited thanks to Structural Analysis methodologies.

A generic structural model of a fixed-wing UAV was synthesized, enriched with the measurement equations from the telemetry. Detectability and isolability analyses were carried out. Residual generators were constructed systematically. It was also shown that dividing the original system graph into its disconnected subgraphs and treating each subproblem separately has significant computational benefits. The efficacy of the framework was showcased with a case study, involving a real-life telemetry log of a UAV which crashed during a mapping mission.

Demonstrating the applicability of the suggested framework on-board, in real-time is a desirable future direction.

REFERENCES

[1] L. Meier, D. Honegger, and M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2015, pp. 6235–6240.

[2] The ArduPilot Development Team (2018). The ArduPilot autopilot suite. [Online]. Available: http://ardupilot.org

[3] B. Gati, "Open source autopilot for academic research - The Paparazzi system," in *2013 American Control Conference*. IEEE, jun 2013, pp. 1478–1481.

[4] T. H. Cox, C. J. Nagy, M. a. Skoog, I. a. Somers, and R. Warner, "Civil UAV capability assessment," Tech. Rep., 2004. [Online]. Available: http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:Civil+UAV+Capability+Assessment{#}0

[5] K. Goebel and B. Saha, "Prognostics Applied to Electric Propulsion UAV," in *Handbook of Unmanned Aerial Vehicles*, K. P. Valavanis and G. J. Vachtsevanos, Eds. Dordrecht: Springer Netherlands, 2015, ch. 43, pp. 1053–1070.

[6] F. Bateman, H. Noura, and M. Ouladsine, "Fault Diagnosis and Fault-Tolerant Control Strategy for the Aerosonde UAV," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 3, pp. 2119–2137, jul 2011.

[7] J. Marzat, H. Piet-Lahanier, F. Damongeot, and E. Walter, "Model-based fault diagnosis for aerospace systems: a survey," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 226, pp. 1329–1360, 2012.

[8] S. Hansen, M. Blanke, and J. Adrian, "A framework for diagnosis of critical faults in Unmanned Aerial Vehicles," *IFAC Proceedings Volumes*, vol. 19, no. 3, pp. 10 555–10 561, 2014.

[9] H. Fan, H. Fang, Y. Dong, H. Shi, and S. Ren, "UAV engine fault and diagnosis with parameter models based on telemetry data," in *2017 Prognostics and System Health Management Conference (PHM-Harbin)*. IEEE, jul 2017, pp. 1–6.

[10] L. Meier, J. Camacho, B. Godbolt, J. Goppert, L. Heng, M. Lizarraga *et al.*, "Mavlink: Micro air vehicle communication protocol," *Online. https://mavlink.io/en/.*, 2009.

[11] J. Gertler, *Fault Detection and Diagnosis in Engineering Systems*. CRC Press, may 1998.

[12] P. Declerck and M. Staroswiecki, "Characterization of the canonical components of a structural graph for fault detection in large scale industrial plants," in *European Control Conference*, 1991.

[13] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, *Diagnosis and Fault-Tolerant Control*, 3rd ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016.

[14] R. Izadi-Zamanabadi, "Structural analysis approach to fault diagnosis with application to fixed-wing aircraft motion," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, vol. 5. IEEE, 2002, pp. 3949–3954 vol.5.

[15] M. Fravolini, V. Brunori, G. Campa, M. Napolitano, and M. La Cava, "Structural Analysis Approach for the Generation of Structured Residuals for Aircraft FDI," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 45, no. 4, pp. 1466–1482, oct 2009.

[16] G. Zogopoulos-Papaliakos and K. J. Kyriakopoulos, "Generating semi-explicit DAEs with Structural Index 1 for fault diagnosis using structural analysis," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2017, pp. 5812–5817.

[17] C. Svärd and M. Nyberg, "Residual generators for fault diagnosis using computation sequences with mixed causality applied to automotive systems," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 6, pp. 1310–1328, nov 2010.

[18] E. Frisk, A. Bregon, J. Aslund, M. Krysander, B. Pulido, and G. Biswas, "Diagnosability Analysis Considering Causal Interpretations for Differential Constraints," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 42, no. 5, pp. 1216–1229, sep 2012.

[19] R. J. Patton and J. Chen, "Review of parity space approaches to fault diagnosis for aerospace systems," *Journal of Guidance, Control, and Dynamics*, vol. 17, no. 2, pp. 278–285, mar 1994.

[20] a. L. Dulmage and N. S. Mendelsohn, "Coverings of bipartite graphs," *Canadian Journal of Mathematics*, vol. 10, pp. 517–534, jan 1958.

[21] M. Krysander, J. Aslund, and E. Frisk, "A Structural Algorithm for Finding Testable Sub-models and Multiple Fault Isolability Analysis," *21st International Workshop on the Principles of Diagnosis*, 2010.

[22] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, mar 1955.

[23] K. G. Murty, "Letter to the Editor—An Algorithm for Ranking all the Assignments in Order of Increasing Cost," *Operations Research*, vol. 16, no. 3, pp. 682–687, jun 1968.

[24] J. E. Hopcroft and R. M. Karp, "An nˆ{5/2} Algorithm for Maximum Matchings in Bipartite Graphs," *SIAM Journal on Computing*, vol. 2, no. 4, pp. 225–231, dec 1973.

[25] S. Ding, *Model-Based Fault Diagnosis Techniques*. Springer, 2013.

[26] D. Jung and C. Sundstrom, "A Combined Data-Driven and Model-Based Residual Selection Algorithm for Fault Detection and Isolation," *IEEE Transactions on Control Systems Technology*, pp. 1–15, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/8122029/

[27] G. Zogopoulos-Papaliakos and K. J. Kyriakopoulos, "Parity-Based Diagnosis in UAVs: Detectability and Robustness Analyses," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2019.

[28] The MAVLink Development Team (2018). (2018) MAVLINK ArduPilotMega Message Set. [Online]. Available: https://mavlink.io/en/messages/ardupilotmega.html

[29] J. Åslund, A. Bregon, M. Krysander, E. Frisk, B. Pulido, and G. Biswas, "Structural Diagnosability Analysis of Dynamic Models," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 4082–4088, jan 2011.

[30] Transportation Safety Board of Canada. Aviation investigation reports. [Online]. Available: http://www.tsb-bst.gc.ca/eng/rapports-reports/aviation/

[31] Air Accidents Investigation Branch. Air Accidents Investigation Branch reports. [Online]. Available: https://www.gov.uk/aaib-reports

[32] The Judge Advocate General's Corps. AIB reports. [Online]. Available: https://www.afjag.af.mil/AIB-Reports/