A Concept for the Automated Reconfiguration of Quadcopters

Kaja Balzereit[1][0000−0001−9203−5902], Marta Fullen[1], and Oliver Niggemann[2]

[1] Fraunhofer IOSB, Industrial Automation Branch (INA), Fraunhofer Center for Machine Learning, Lemgo, Germany
{name.surname}@iosb-ina.fraunhofer.de
[2] Faculty of Machine Construction, Helmut-Schmidt-University, Hamburg, Germany
oliver.niggemann@hsu-hh.de

Abstract. Quadcopters are susceptible to internal and external influences, many of which may lead to faults. To ensure a safe and reliable flight, the quadcopter needs to recover autonomously from faults. However, existing approaches mainly rely on parametrical faults or require a predefinition of possible faults which is not realistic for a complex real-world scenario. The recovery from unforeseen faults and structural faults like a failing engine is still an open research gap.
Hence, in this paper, a concept for the automated reconfiguration, i.e. the automated recovery from a fault, which only uses information about non-faulty system behavior and is able to handle structural changes is presented. From the information about non-faulty behavior a non-faulty system model is created using established machine learning methods. Thus, faults are detected by learned model and no pre-definition of faults is needed. The system structure is modeled using a logical calculus which allows for modeling available system parts and the causal coherences between these.
The approach is applied to a simulation of a quadcopter which underlies a structural fault. It is shown that the approach extends the capabilities of a quadcopter to handle faults autonomously and ensure stability and reliability.

Keywords: Automated Reconfiguration · Symptom Generation · Fault Recovery · Quadcopter

1 Introduction

Unmanned Aerial Vehicles (UAVs) are an emerging technology of great interest in military and civil applications [21,11]. The market for UAVs has emerged especially in the last years and is expected to rise continuously [11,7]. One type of UAVs, the quadcopters which consist of four rotors that can be controlled independently from each other, is in the scope of most research studies [21]. Since

quadcopters operate in an open world, the requirements towards reliability and safety are very high. Today, due to many safety concerns, the usage of quadcopters is subject to massive restrictions [11]. It stems from a fact that minor UAV fault can lead to major consequences, including huge damage to humans or environment. Hence, quadcopters need to be designed to be robust to environmental disturbances and tolerant towards internal faults. Currently, controllers used for Cyber-Physical Systems, including quadcopters, are static and at most applicable to a predefined set of faults [24]. An automated reconfiguration of the control is needed to maintain a stable flight in presence of faults [5]. Reconfiguration is the task of recovering a valid system state after a fault has occurred [3]. However, the automated reconfiguration for quadcopters is still an open research gap due to some unanswered research questions, two of which follow.

RQ1: How can automated reconfiguration handle unforeseen faults? Existing approaches on fault-tolerant control are mostly based on an enumeration of known faults and the storage of control instructions specific to these faults [21]. However, when it comes to unknown faults, these approaches can no longer guarantee stable control. Hence, in this paper, a concept which needs no information about known faults but only works on information about non-faulty behavior is presented. Quadcopters contain a multitude of sensors, continuously logging data during the flight. This huge amount of data can be analyzed in an intelligent way using Machine Learning (ML) methods. These methods enable learning a model of the system from historical data and can then be used to detect anomalous behavior, which might indicate the presence of faults [18]. The data used for training contains only measurements from non-faulty flights, thus deviations from non-faulty behavior are handled as candidates for faults.

RQ2: Which formalism is able to handle structural faults? Control theory in general is concerned with a static system model. However, when major faults like an engine failure or a rotor ripping off occur, the system is no longer representative, leading to the control becoming invalid. Logical reasoning can be used here to draw conclusions about the impact of a fault as well as still available and functional components and actions [3]. Thus, new control instructions can be determined even in the presence of major faults.

The contribution of this paper is twofold: (1) A concept for the automated reconfiguration of quadcopters that handles unforeseen faults is presented. Therefore, only information and data about non-faulty flights is used. Thus, faults are detected as deviations from non-faulty behavior and do not have to be known a-priori. (2) An encoding of the reconfiguration problem into first-order logic (FOL) is presented that allows for analyzing the extent of structural faults is presented. Thus, also major faults like component failure (e.g. an engine) can be handled.

Please note that the scope of this paper is not to handle one given fault in an optimal way but to present a concept that restores a safe flight in the presence of unforeseen faults and disturbances. The paper is structured as follows: First, in section 2 the related work is presented and discussed. Then, the solution concept

is presented in section 3. In section 4, using a simulation of a quadcopter, the applicability of the solution concept is evaluated.

2 Related Work

Most research on quadcopter control is concerned with fault-tolerant control (FTC) [22,17]. The goal of FTC is to maintain a stable flight in the presence of wind as well as actuator and sensor faults. Therefore, the quadcopter is described in quantitative terms [26] using the equations

$$\begin{aligned} x(t+1) &= f(x(t), u(t)) \\ y(t) &= g(x(t), u(t)) \end{aligned} \tag{1}$$

where x describes the system states, u describes the input (e.g. the rotor velocities) and y describes the output (e.g. the altitude and attitude of the quadcopter). f, g are functions describing the properties of the quadcopter. Using this equation system, the optimal input to change the attitude or altitude of the quadcopter can be determined, and thus, the quadcopter can adapt to environmental changes of the wind speed or sensor faults. However, major disturbances like a rotor ripping off or the battery failing cannot be represented by a static model (1) and require a new type of model. These disturbances lead to changed dynamics such that the functions f, g are no longer valid and need to be adapted. However, this adaption cannot be done online but requires expert knowledge. To handle these changes, a reconfiguration of the controller is needed [5].

Lunze [14] presented a concept towards reconfigurable control for UAVs using overdetermined sets of equations. However, this approach requires explicit modeling of the quadcopter behavior and thus a large amount of expert knowledge. Wang et al. [27] presented a combination of classical control and constraint satisfaction. The scope of this work is slightly different: no faults are handled but an optimal control for a given path is searched. Chen et al. [6] presented an approach to reconfiguration of actuator faults by an advanced estimation procedure. However, the faults need to be modeled explicitly. Thus, no unknown faults can be handled. Adaptive control algorithms, as presented by Huynh et al. [10] among others, are concerned with continuous disturbances like wind or varying parameters. Unknown or structural faults cannot be handled [21]. Robust control algorithms as presented by Thanh et al. [23] and Ton et al. [25] handle parametric uncertainties and are even adaptable to nonlinear disturbances. However, no structural faults can be handled [21].

The reconfiguration concept presented here can be seen as an extension to classical control theory. The goal is not to determine control instructions for an optimal flight, but to identify the necessary actions to recover a stable flight in the presence of faults.

3 Solution Concept

The goal of automatic reconfiguration is to manipulate the system inputs to restore valid system behavior [3]. To perform automatic reconfiguration, some kind of redundancy is necessary [5]. This can be either physical redundancy, e.g. duplicate components or sensors, or analytical redundancy, i.e. information about the relation between different values measured by the system. Quadcopters in general contain numerous redundancies to ensure safety and reliability requirements are fulfilled. Typical examples for physical redundancy are multiple engines, multiple batteries or multiple sensors. Analytical redundancy can be asserted through knowledge about coherences and relations between sensor and actuator values.

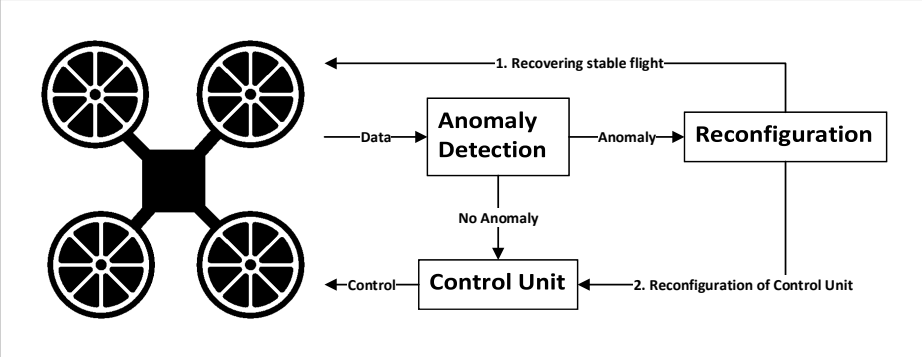

Fig. 1. Concept for the automated reconfiguration of quadcopters. First, a symptom generation is performed on the data. If at least on symptom is present, reconfiguration is performed in two steps.

Faults can be differentiated using the component they concern. Thus, we divide actuator faults which are usually modeled as loss of effectiveness of one rotor, sensor faults which are modeled as sensors returning wrong values and strutuctual/system faults which affect system components like engines or batteries. Actuator faults can be handled using robust control methods, sensor faults usually are handled using Kalman filters [15]. In general, not every possible fault and its consequences is known a-priori because quadcopters operate in a non-deterministic environment, many different factors like wind speed, rain and air humidity have an impact on the behavior of the quadcopter. Thus, foreseeing every possible consequence of environmental influences on the quadcopters behavior and enumerate every possible fault is impossible. The goal of reconfiguration is to adapt the rotor speeds to recover a stable flight in case of every unforeseen fault or at least to perform a safe emergency landing.

The basic concept of automatic reconfiguration for quadcopters is shown in Figure 1. The reconfiguration system operates while the quadcopter is flying and continuously checks for deviations. First, the data delivered by the sensors

of the quadcopter is compared to the learned models, checking for deviations and the presence of symptoms. If no symptoms are present, no further actions are needed and neither the control unit nor the system goal are changed. In case of at least one symptom, the reconfiguration unit first estimates the extent of the deviation. Then, reconfiguration is performed in two steps [5]:

1. A set of actions that moves the quadcopter back into a valid flight is searched and applied directly.
2. A controller that stabilizes the quadcopter in its valid flight in the presence of faults (e.g. a different control structure due to a failure of one sensor) is determined using well known controller design methods.

In some cases, the reconfiguration to a valid flight is no longer possible since the damage is too high. Then, during the first reconfiguration step, actions to perform an emergency landing or a return to launch, if possible, are applied until the quadcopter has landed. The control unit is not reconfigured, since a stable flight cannot be recovered.

3.1 Modeling the System Structure

To enable the reconfiguration unit to handle structural faults, it needs to reason about the consequences of a fault. Therefore, information about the causal coherences, i.e. the impact of a change in one component to other components is needed. Logic is used widely in Artifical Intelligence since it allows for modeling causal coherences and drawing logical conclusions about the system [20]. Basic physical and mathematical knowledge can be modeled in logic to support the reconfiguration unit in its decision making [13]. Using the logical calculcus *Satisfiability Modulo the Theory of Linear Arithmetic* [4], also continuous variables (e.g. rotor velocity, wind speed, ...) can be modeled.

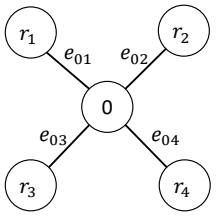

Fig. 2. Topology of a quadcopter.

To model the causal coherences of the quadcopter, first, the system topology is analyzed. Thus, the coherences between the components can be described in terms of a logical calculus. Therefore, the fuselage 0 and each rotor r_1, r_2, r_3, r_4 are modeled as nodes, the set of all nodes is represented by N. Since every rotor is connected to the fuselage, the edges are represented by $E = \{e_{01}, e_{02}, e_{03}, e_{04}\}$. Fig. 2 shows the resulting graph for a quadcopter.

3.2 Symptom Generation

The behavior of every component, i.e. the rotors and the fuselage, is monitored using component models [8]. Today, due to sophisticated learning methods, these models no longer need to be created manually but can be learned [19,2]. Comparing the current behavior of a component to the model, deviations indicating faults are detected. This information is encoded by a binary assignment $\omega : N \rightarrow \{\top, \bot\}$ which is true if the component behaves non-faulty and false otherwise. A connection between two components may only be used if the components are non-faulty, i.e. $b^+_{e_{ij}} \Rightarrow \omega(i) \wedge \omega(j)\ \forall e_{ij} \in E$.

Thus, component faults are directly taken into account by the reconfiguration unit. Structural changes due to a component fault (e.g. a failing engine) can be represented by the component model of the corresponding rotor showing a deviation to the current behavior. Hence, the reconfiguration unit identifies the necessary changes under consideration of the impact of the fault.

Therefore, every components behavior is monitored using sophisticated machine learning methods utilizing a normal behavior system model. Here, a modeling formalism which transforms the data into a black- or gray-box representation is used. Such a structure does not explicitly model each observation but creates a new representation based on the normal behavior data. Depending on the formalism used, various measures describe how well the current status fits into the model. An example well-known algorithm is Self-Organizing Map [12], a type of neural network. The measure of fitting in this case is the quantization error, which is the difference between the current status location mapping and the best-matching neuron neighborhood in the model. If this quantization error is high, the component is assumed to behave anomalously, so a symptom is reported to the reconfiguration unit to trigger further actions.

For the creation of these models only data about non-faulty system behavior is required. Thus, no fault modes or an enumeration of known faults has to be given.

3.3 Reconfiguration

As mentioned above, the reconfiguration is performed in two steps: One step to regain a stable flight and one step to maintain this stable flight. Whilst the second step can be done using well–known controller design methods, the first step is still an open research gap [5].

For each rotor, the impact of acceleration and deceleration on the attitude and altitude of the quadcopter is modeled in terms of logical constraints, for example

If rotor r_1 is accelerated, the pitch angle decreases.

or

If all rotors are accelerated proportionally, the height increases.

Thus, the impact of changing rotor velocities on the behavior of the quadcopter can be modeled. Based on this, the reconfiguration unit is able to choose an intelligent combination of rotor accelerations and decelerations to recover a

stable flight, if possible. Otherwise, the reconfiguration unit tries to bring the quadcopter to a safe state (e.g. by a return to launch or an emergency landing).

To enable the reconfiguration method to change the velocity of the rotors, every connection is assigned with two binary variables that lead to an increase or decrease of the current velocity of the corresponding rotor. Therefore, for each edge $e \in E$ two binary variables $b^+_{e_{ij}}, b^-_{e_{ij}}$ that trigger an increase or decrease of the corresponding rotor are introduced. Thus, $b^+_{e_{ij}} \rightarrow inc(r_j), b^-_{e_{ij}} \rightarrow dec(r_j)$. The predicates inc, dec indicate that the velocity of the corresponding rotor needs to be increased or decreased. How this is realized in detail needs to be defined by an expert, e.g. that an increase is always realized by increase the velocity given a fixed difference or a percentage amount. To avoid that both variables are set to true at the same time (which would require a simultaneous increase and decrease of the velocity of one rotor) the constraint $b^+_{e_{ij}} \oplus b^-_{e_{ij}} \ \forall e_{ij} \in E$ is needed.

Thus, the reconfiguration problem is modeled as a first-order logic formula. If at least one symptom occurs, i.e. one component behaves anomalously, the reconfiguration unit determines for every connection, if the velocity of each rotor has to be increased, decreased, or does not have to be changed by setting the corresponding binary variable to true. Thus, the changes for recovering a stable flight are identified by a combination of acceleration and deceleration of rotor velocities.

4 Results

This section presents the results of symptom generation and reconfiguration experiments. The symptom generation approach has been tested on real quadcopter data, to validate the approach as feasible in real-life scenarios. Reconfiguration experiments utilize a simulation to verify the outcome of reconfigured and non-reconfigured fault scenario. The used simulation of the quadcopter is described in the appendix. The free variables in the logical formula created as described above are assigned with the current values of the sensors. Then, the formula is checked for satisfiability to determine which rotors needs to be accelerated and which need to be decelerated to recover a stable flight in the presence of faults using the Z3 solver [16].

4.1 Symptom Generation

To evaluate the concept, we show that it is indeed possible to detect anomalies in quadcopter behavior using machine learning-based modeling formalisms. Self-Organizing Map (SOM) model formalism is used to perform a preliminary analysis and investigate whether the methods are feasible to detect anomalies in quadcopter flight. We consider an approach feasible for symptom generation if it is possible, at least partially, to differentiate between normal behavior and anomalous behavior using the model.

The symptom generation is performed on data from an industrial drone, to ensure the first step of the concept is viable in real-life applications. Quadcopters are operated using the PX4, an open source flight control software for

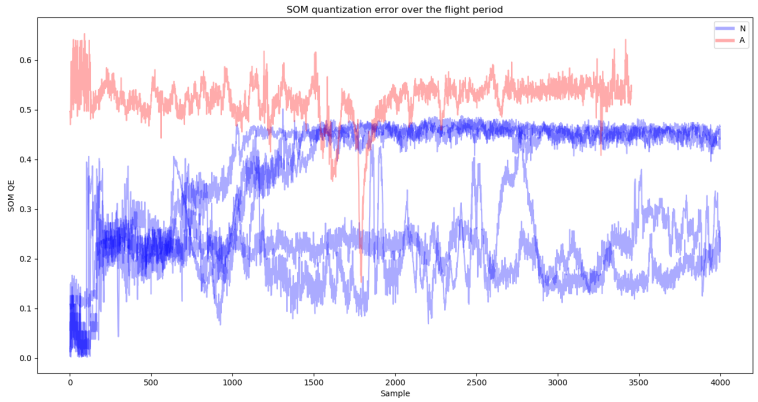

Fig. 3. SOM quantization error (QE) over time for anomalous (red) and non-anomalous (blue) flights. QE generated by a 10x10 SOM from a select combination of sensors and learned on non-anomalous data.

quadcopters and other unmanned vehicles. It allows logging device inputs (sensors etc.), internal states (CPU load, attitude etc.) and log messages. The log structure requires that sensors are organized within predefined sensor groups, however the sampling rates, and therefore timestamps, of different sensor groups are independent from each other. It is therefore only possible to match the values from one sensor group at a time. The SOM model of normal behavior has been learned from chosen sensor logs of quadcopter flights where no faults occurred and the drone was considered to behave entirely correct. This model has then been used to detect anomalies in faulty flight logs.

It is expected that the anomalous behavior data at least partially overlaps the normal behavior in terms of quantization error, however, the results have shown that some sensor combinations generate a quantization error much higher in the case of faulty behavior than normal behavior. This outcome creates a perfect opportunity for symptom generation, where the maximum quantization error of normal behavior is used as the error threshold for a symptom, and a symptom is reported as soon as the error crosses the threshold. Figure 3 illustrates the difference in quantization error over the initial flight period of flight: the error generated by SOM from the anomalous behavior data is decidedly higher than for normal behavior data.

4.2 Fault Scenario: Engine Failure

The Engine Failure fault scenario focuses on a quadcopter flight, where one of the four engines that provide acceleration to the rotors fails mid-flight. The flight begins with a stable flight at the height of 10 meteres, until one engine fails and its corresponding rotors velocity decreases to 0. Only the three remaining rotors can

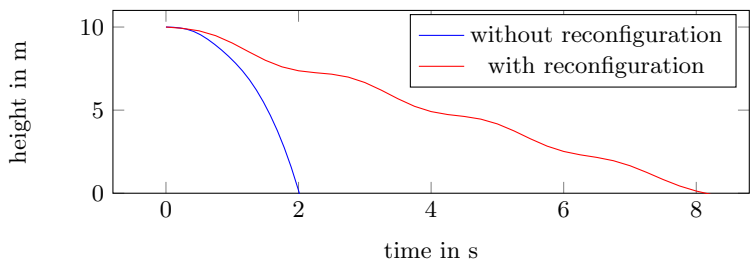

Fig. 4. Height of quadcopter with one engine failing at time $t_f = 0$s.

be used to control the flight and the goal is to adjust their corresponding motor parameters in such a way that the drone does not suffer a catastrophic failure. Classical control is not capable of handling the new situation with one less rotor: the control unit only recognizes a deviation in the attitude of the quadcopter and tries to adapt the velocity of the quadcopter rotor such that stability is regained. It is not taken into consideration, that one rotor is non-functional. Even in the presence of a fault, all of the engines are controlled similarly. No distinction between available and crashed engine can be made. The control unit uses all four rotors to adapt to the deviation, even though only three rotors are still available. Thus, no stable flight can be regained leading to a crash of the quadcopter. The reconfiguration method based on logical calculus enables the quadcopter to handle the fault and avoid a crash. First, the faulty component, in this case the failing engine is identified using the component models. Then, the reconfiguration unit calculates new control instructions to stabilize the flight by increasing the velocities of the rotors which are still available. However, a stable flight cannot be regained since the disturbance of a failing engine is to severe. Thus, an emergency landing is performed.

The simulated behavior of quadcopters equipped with classical control approach and reconfiguration approach is shown in Figure 4. Using the classical control, which does not adapt to the changed structure of the quadcopter, the quadcopter crashes within approximately 2 seconds. Since the quadcopter accelerates while falling, the velocity when touching the ground is around 13.4 meters per second which can lead to massive damages of the quadcopter and the surroundings. When the reconfiguration is enabled, the fall of the quadcopter is decelerated. The quadcopter touches the ground after approximately 8.4 seconds with a velocity of approximately 1.3 meters per second. Thus, damage to the quadcopter and the surrounding can be reduced significantly.

5 Summary and Outlook

Quadcopter flights are susceptible to internal disturbances, like failing components, as well as environmental disturbances like winds. Today, FTC is commonly used to enable a quadcopter to maintain a stable flight even in the presence of

faults. However, FTC is focused on handling numerical faults, which, in addition, often have to be predefined. Thus, major faults which cause structural changes of the quadcopter cannot be handled by classical FTC. Therefore, this paper presents a concept for the automated reconfiguration to enable quadcopters to handle structural and unforeseen faults. The concept is based on the combination of a logic-based reconfiguration method. Faults are detected as deviations from models which are learned from non-anomalous behavior. Based on this information, reconfiguration is initiated – if necessary. During reconfiguration, stable flight of a quadcopter is described in terms of a logical calculus which allows for modeling condition of a stable flight and requirements to recover a stable flight, if possible. The approach is evaluated using an interactive simulation of a quadcopter. One of the four engines failing represents the structural fault. It is shown that without reconfiguration, the quadcopter crashes within 2 seconds and the velocity when touching the ground is high. With reconfiguration, the quadcopter touches the ground after 8 seconds with a far lower velocity.

Future work will focus on further fault scenarios like failures leading to a reduced engine performance or failing battery cells to prove the applicability of the concept. Then, also the scalability of the approach will be examined by using more detailed models.

Acknowledgments

This work was founded by the Fraunhofer Cluster of Excellence "Cognitive Internet Technologies".

A Appendix

Evaluation is performed using a simulation implemented in Modelica [9] is used. The flight of the quadcopter is described as a state space model (referring to [1])

$$\ddot{\phi} = \dot{\theta}\dot{\psi}a_1 + \dot{\theta}a_2\Omega_r + b_1U_2 \tag{2}$$
$$\ddot{\theta} = \dot{\phi}\dot{\psi}a_3 - \dot{\phi}a_4\Omega_r + b_2U_3 \tag{3}$$
$$\ddot{\psi} = \dot{\phi}\dot{\theta}a_5 + b_3U_4 \tag{4}$$
$$\ddot{x} = (\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi))U_1/m \tag{5}$$
$$\ddot{y} = (\cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi))U_1/m \tag{6}$$
$$\ddot{z} = -g + (\cos(\phi)\cos(\theta))U_1/m \tag{7}$$

with

$$U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2), U_2 = b(\Omega_2^2 - \Omega_4^2), U_3 = d(-\Omega_1^2 + \Omega_3^2), \tag{8}$$
$$U_4 = d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2), \Omega_r = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4, \tag{9}$$
$$a_1 = \frac{I_{yy} - I_{zz}}{I_{xx}}, a_2 = \frac{J_x}{I_{xx}}, a_3 = \frac{I_{zz} - I_{xx}}{I_{yy}}, a_4 = \frac{J_r}{I_{yy}}, a_5 = \frac{I_{xx} - I_{yy}}{I_{zz}}, \tag{10}$$

$$b_1 = \frac{l}{I_{xx}}, b_2 = \frac{l}{I_{yy}}, b_3 = \frac{1}{I_{zz}}. \tag{11}$$

Variable	Value	Unit	Description
m	1.1	kg	mass of drone
J_r	$8.5 \cdot 10^{-4}$	$\mathrm{kg} \cdot \mathrm{m}^2$	rotor inertia
$I_{xx} = I_{yy}$	$1.96 \cdot 10^{-2}$	$\mathrm{kg} \cdot \mathrm{m}^2$	quadcopter inertia around x/y-axis
I_{zz}	$2.62 \cdot 10^{-2}$	$\mathrm{kg} \cdot \mathrm{m}^2$	quadcopter inertia around z-axis
l	0.21	m	length of arms
b	$9.29 \cdot 10^{-5}$	$\mathrm{N} \cdot \mathrm{s}^2$	thrust coefficient
d	$1.1 \cdot 10^{-6}$	$\mathrm{N} \cdot \mathrm{m} \cdot \mathrm{s}^2$	drag coefficient

Table 1. Values for the parameters of the quadcopter

The values of the parameters are shown in Table 1 (also referring to [1]). The inputs of the system are represented by the velocities of the rotors $\Omega_1, \Omega_2, \Omega_3, \Omega_4$.

References

1. Alexis, K., Nikolakopoulos, G., Tzes, A.: Model predictive quadrotor control: attitude, altitude and position experimental studies. IET Control Theory & Applications **6**(12), 1812–1827 (2012)
2. Balzereit, K., Maier, A., Barig, B., Hutschenreuther, T., Niggemann, O.: Data-driven identification of causal dependencies in cyber-physical production systems. In: 11th International Conference on Agents and Artificial Intelligence (02 2019)
3. Balzereit, K., Niggemann, O.: Automated reconfiguration of cyber-physical production systems using satisfiability modulo theories. In: 3rd IEEE International Conference on Industrial Cyber-Physical Systems (2020)
4. Barrett, C., Tinelli, C.: Satisfiability modulo theories. In: Handbook of Model Checking, pp. 305–343. Springer (2018)
5. Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M.: Diagnosis and fault-tolerant control, vol. 2. Springer (2006)
6. Chen, F., Lei, W., Tao, G., Jiang, B.: Actuator fault estimation and reconfiguration control for the quad-rotor helicopter. International Journal of Advanced Robotic Systems **13**(1), 33 (2016)
7. Cohn, P., Green, A., Langstaff, M., Roller, M.: Commercial drones are here: The future of unmanned aerial systems. McKinsey & Company (2017), `https://www.mckinsey.com/industries/capital-projects-and-infrastructure/our-insights/commercial-drones-are-here-the-future-of-unmanned-aerial-systems#`, called 22.04.2020
8. De Kleer, J., Brown, J.S.: Mental models of physical mechanisms and their acquisition. Cognitive skills and their acquisition pp. 285–309 (1981)
9. Elmqvist, H., Mattsson, S.E., Otter, M.: Modelica: The new object-oriented modeling language. In: 12th European Simulation Multiconference, Manchester, UK (1998)

10. Huynh, M.Q., Zhao, W., Xie, L.: L 1 adaptive control for quadcopter: Design and implementation. In: 2014 13th International Conference on Control Automation Robotics & Vision (ICARCV). pp. 1496–1501. IEEE (2014)
11. Joshi, D.: Drone technology uses and applications for commercial, industrial and military drones in 2020 and the future. Businees Insider (2019), https://www.businessinsider.com/drone-technology-uses-applications?r=DE&IR=T, called 22.04.2020
12. Kohonen, T.: Self-Organizing Maps, Springer Series in Information Sciences, vol. 30. Springer-Verlag, Berlin Heidelberg, 3 edn. (2001)
13. Lake, B.M., Ullman, T.D., Tenenbaum, J.B., Gershman, S.J.: Building machines that learn and think like people. Behavioral and brain sciences **40** (2017)
14. Lunze, J.: From fault diagnosis to reconfigurable control: A unified concept. In: 2016 3rd Conference on Control and Fault-Tolerant Systems (SysTol). pp. 413–421. IEEE (2016)
15. Moghadam, M., Caliskan, F.: Actuator and sensor fault detection and diagnosis of quadrotor based on two-stage kalman filter. In: 2015 5th Australian Control Conference (AUCC). pp. 182–187. IEEE (2015)
16. de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer (2008)
17. Nguyen, N.P., Hong, S.K.: Fault diagnosis and fault-tolerant control scheme for quadcopter uavs with a total loss of actuator. Energies **12**(6), 1139 (2019)
18. Niggemann, O., Frey, C.: Data-driven anomaly detection in cyber-physical production systems. at-Automatisierungstechnik **63**(10), 821–832 (2015)
19. Niggemann, O., Lohweg, V.: On the Diagnosis of Cyber-Physical Production Systems - State-of-the-Art and Research Agenda. In: Proceedings of the Twenty-Ninth Conference on Artificial Intelligence (2015)
20. Nilsson, N.J.: Logic and artificial intelligence. Artificial intelligence **47**(1-3), 31–56 (1991)
21. Shraim, H., Awada, A., Youness, R.: A survey on quadrotors: Configurations, modeling and identification, control, collision avoidance, fault diagnosis and tolerant control. IEEE Aerospace and Electronic Systems Magazine **33**(7), 14–33 (2018)
22. Tabata, A., Satoh, Y., Nakamura, H., Kato, K.: Adaptive fault tolerant control of quadcopter by using minimum projection method. In: IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society. pp. 2201–2206. IEEE (2018)
23. Thanh, H.L.N.N., Hong, S.K.: Quadcopter robust adaptive second order sliding mode control based on pid sliding surface. IEEE Access **6**, 66850–66860 (2018)
24. Tomiyama, T., Moyen, F.: Resilient architecture for cyber-physical production systems. CIRP Annals **67**(1), 161–164 (2018)
25. Ton, C.T., Mackunis, W.: Robust attitude tracking control of a quadrotor helicopter in the presence of uncertainty. In: 2012 IEEE 51st IEEE Conference on Decision and Control (CDC). pp. 937–942. IEEE (2012)
26. Venkatasubramanian, V., Rengaswamy, R., Yin, K., Kavuri, S.N.: A review of process fault detection and diagnosis: Part i: Quantitative model-based methods. Computers & chemical engineering **27**(3), 293–311 (2003)
27. Wang, Y., Ramirez-Jaime, A., Xu, F., Puig, V.: Nonlinear model predictive control with constraint satisfactions for a quadcopter. In: Journal of Physics: Conference Series. vol. 783, p. 012025. IOP Publishing (2017)