Loss-of-Control Prediction of a Quadcopter Using Recurrent Neural Networks

Altena, A.; van Beers, J.J.; de Visser, C.C.

DOI
10.2514/1.I011231

Publication date
2023

Document Version
Final published version

Published in
Journal of Aerospace Information Systems (online)

Citation (APA)
Altena, A., van Beers, J. J., & de Visser, C. C. (2023). Loss-of-Control Prediction of a Quadcopter Using Recurrent Neural Networks. *Journal of Aerospace Information Systems (online)*, *20*(10), 648-659. https://doi.org/10.2514/1.I011231

JOURNAL OF AEROSPACE INFORMATION SYSTEMS
Vol. 20, No. 10, October 2023

Loss-of-Control Prediction of a Quadcopter Using Recurrent Neural Networks

Anique V.N. Altena,* Jasper J. van Beers,† and Coen C. de Visser‡
Delft University of Technology, 2629 HS Delft, The Netherlands

https://doi.org/10.2514/1.I011231

Loss of control (LOC) is a prevalent cause of drone crashes. Onboard prevention systems should be designed requiring low computing power, for which data-driven techniques provide a promising solution. This study proposes the use of recurrent neural networks (RNNs) for LOC prediction. Four architectures were trained in order to identify which RNN configuration is most suitable and if this model can predict LOC for changing aerodynamic characteristics, wind conditions, quadcopter types, and LOC events. One-hundred and seventy-two real-world LOC events were conducted using a 53 g Tiny Whoop, a 73 g URUAV UZ85, and a 265 g GEPRC CineGO quadcopter. For these flights, LOC was initiated by demanding an excessive yaw rate (2000 deg/s), which provokes an unrecoverable upset and subsequent crash. All RNNs were trained using only onboard sensor measurements. It was found that the commanded rotor values provided the clearest early warning signals for LOC because these values showed saturation before LOC. Moreover, all four architectures could correctly and reliably predict the impending LOC event 2 s before it actually occurred. Furthermore, to investigate generality of the methodology, the predictors were successfully applied to flight data in which the quadcopter mass, blade diameter, and blade count were varied.

I. Introduction

THERE is an ever-increasing number of applications for unmanned aerial vehicles (UAVs), and they prove invaluable for tasks such as aerial observation, infrastructure inspection, package delivery, and entertainment. However, their popularity also raises safety concerns. In particular, the extreme agility of many UAV types, such as the quadcopter, can rapidly lead to loss-of-control (LOC) events, potentially resulting in a crash. A 2017 study showed that out of 100 reported UAV mishaps, 34 could be categorized as LOC, making it the largest contributor to UAV mishaps [1].

Indeed, LOC is not only dangerous for UAVs: it is a concern for many controlled systems. Between 2011 and 2020, 20.5% of all fatal accidents in commercial aviation resulted from a LOC event [2]. This emphasizes the importance of extending research on new LOC prevention techniques.

One of the leading studies on LOC prevention is part of NASA's Aviation Safety Program. Belcastro and Jacobson [3] proposed a holistic LOC prevention approach, which highlights the value of onboard integrated systems that help in detecting, avoiding, mitigating, and recovering from LOC scenarios. Some direct results of this proposal are the envelope-aware flight management system from Ref. [4] and the flight safety assessment and management system modeled as a Markov decision process from Ref. [5], which was also shown to work on off-nominal systems [6]. The authors presented an integrated flight envelope estimation, flight planning, and flight safety assessment and management framework to diagnose high-risk events that may induce LOC, and they provided automatic LOC recovery if required. However, this framework does not explicitly forecast a time horizon to LOC, and it requires a priori knowledge on the safe flight envelope (SFE).

Another study addressing aircraft LOC was presented by Rohith [7]. The author applied bifurcation analysis to identify abnormal flight dynamics. Bifurcation happens when small changes in system parameters result in large changes in behavior. The author proposed to use a sliding mode controller for LOC recovery. However, the study does not address LOC prevention. On the other hand, a study that does focus on LOC prevention was published by Zhao and Zhu [8]. The authors designed a closed-loop system with bandwidth adaptation for a real-time tradeoff between tracking performance and robustness against LOC. The study focuses on wind-induced LOC events only.

As opposed to the previously mentioned studies, most LOC prevention techniques currently rely on knowledge of the safe flight envelope [9–17]. Some of these studies are concerned with LOC prevention. They assume that either a predefined flight envelope or a real-time dynamic flight envelope is available. Tekles et al. [11] proposed a command-limiting flight envelope protection scheme. The pilot inputs are limited or overridden by the protection scheme to avoid excursions for critical flight parameters, such as the angle of attack. A different approach was taken by Stepanyan et al. [13]. Instead of overriding or limiting the pilot commands, the authors designed a system that estimates LOC boundaries given pilot inputs, historical aircraft state data, and estimated aircraft dynamics. Aural, visual, and tactile cues are presented to the pilot to keep control inputs within limits. Finally, Kirkendoll [17] addressed aircraft LOC prevention in the vicinity of terrain. The author designed an automatic LOC prevention system through multiple monitoring and controlling schemes based upon flight envelope limits.

An important topic that is addressed by these studies is the real-time updating of the flight envelope. The flight envelope may change over time due to changing aerodynamic characteristics, operating conditions, and failures. Thus, continuous and online recalculation of the SFE is necessary. An example study on this topic was presented by Schuet et al. [10]. However, in this study, a global (full-envelope) aircraft dynamical model must be available for envelope estimation, which may not be available following a failure. Another study on this topic has been published by Zhang et al. [14]. The flight envelope in this case is estimated based on an onboard stored database of flight envelopes, which are precomputed for various characteristic failure modes.

A downside of the previous studies is that these applications are mainly tested on aircraft and in simulation. Determining the SFE for a quadcopter is challenging by virtue of their high state-space dimension and extreme control authority, particularly in roll and pitch. Nevertheless, Sun and de Visser [18] applied a Monte Carlo

Received 16 December 2022; revision received 15 May 2023; accepted for publication 8 June 2023; published online Open Access 2 August 2023. Copyright © 2023 by the authors. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. All requests for copying and permission to reprint should be submitted to CCC at www.copyright.com; employ the eISSN 2327-3097 to initiate your request. See also AIAA Rights and Permissions www.aiaa.org/randp.

*Ph.D. Candidate, Aircraft Noise and Climate Effects Section, Faculty of Aerospace Engineering, Kluyverweg 1; A.V.N.Altena@tudelft.nl.

†Ph.D. Candidate, Control and Simulation Section, Faculty of Aerospace Engineering, Kluyverweg 1; J.J.vanBeers@tudelft.nl.

‡Associate Professor, Control and Simulation Section, Faculty of Aerospace Engineering, Kluyverweg 1; C.C.deVisser@tudelft.nl.

simulation in combination with reachable set theory to determine the set of states in which a quadcopter can maneuver freely and return to safe conditions within a predefined period of time, leading to an approximation of the SFE. One other attempt to estimate the flight envelope in real time on a fixed-wing UAV was presented in Ref. [15]. However, in this study, the flight envelope is derived from a general nonlinear aircraft model, which is not available or suitable for all types of UAVs.

Additionally, there are other drawbacks to this approach. Continuous and online recalculation of the SFE of a nonlinear quadrotor currently represents a computationally intractable problem. This is especially true on small UAVs with limited onboard computational capabilities [19]. Therefore, new LOC prediction and prevention methods should be designed that are computationally efficient. Ideally, such approaches should rely only on onboard sensor information and be able to accommodate varying operating and aerodynamic characteristics. Data-driven techniques can meet these requirements.

One study that touches upon LOC prediction using a data-driven technique was performed by Campbell et al. [16]. Here, an autoencoder neural network is trained to detect latent features in the sensor data of an aircraft and labels dramatic shifts in these features as anomalies. These anomalies are an indication that a LOC event might occur. Nonetheless, this research still depends on a priori knowledge of the SFE. To avoid false alerts, information regarding the current position of the aircraft states with respect to the SFE is used for training, which is undesirable for LOC prediction on UAVs for the aforementioned reasons.

The main contribution of this paper is a novel data-driven LOC prediction methodology for quadcopters that requires only onboard measurements, and which does not require a priori knowledge of the SFE. We compare different data-driven techniques for prediction of LOC on quadcopter flight data and investigate their ability to generalize LOC prediction for different operating and aerodynamic characteristics. In particular, various recurrent neural network (RNN) architectures are analyzed and compared. Furthermore, different onboard measured parameters will be used to identify the most informative early warning signals for LOC, providing insights into the LOC problem itself. This work, to the best of the authors' knowledge, represents the first attempt to use real-world flight data, which was obtained during more than 100 LOC events flown with a micro-quadcopter (56 g), to train the LOC predictors. The new methodology can reliably predict excessive yaw-rate-induced LOC events 2 s before they occur on two different larger (73 and 265 g) quadcopters, demonstrating the generality of the approach.

The remainder of this paper is structured as follows: First, an introduction to RNNs is provided in Sec. II. Information about the flight tests and data processing is described in Sec. III. Section IV continues with the results, after which they are discussed in Sec. V. The conclusions are presented in Sec. VI.

II. Recurrent Neural Networks

Neural networks have the ability to detect patterns in large amounts of data. This is a favorable characteristic for the LOC prediction problem because it is unknown what exact combination of quadcopter aerodynamic characteristics, operating conditions, and vehicle behavior induces a LOC event.

Recurrent neural networks are especially attractive for analysis of time-series data, such as that obtained from the quadcopter's sensors. In a classical recurrent neural network (RNN) layer, nodes are replaced by memory cells that receive at time t the current input data in addition to their own output produced at time $t-1$ [20]. Due to these cells, RNNs can identify temporal dependencies in sequential data.

However, when long-term dependencies must be identified, traditional RNNs suffer from vanishing or exploding gradient issues during the training process. To address this concern, other types of memory cells can be used, of which the long short-term memory (LSTM) [21] and gated recurrent unit (GRU) [22] are most common. The principle behind these variants is to control the flow of data within the memory cell by introducing gates. These ensure that long-term features are maintained over a longer period, instead of being overwritten each time step. The GRU has fewer gates as compared to the LSTM, making it less complex. This means that this network type is computationally more efficient during training, while hardly losing performance. On the other hand, for large datasets, LSTM might be better at capturing any longer temporal features present in the data [23].

Besides replacing traditional memory cells with more advanced cells, there are two other options to improve the performance of an RNN. The first option is using a bidirectional network, wherein a data sequence is analyzed both forward and backward (i.e. it extracts features from both past and future states) [24]. Another option is to precede the RNN with other types of neural networks, such as a convolutional neural network (CNN) [25]. This additional network extracts high-level features from data, after which the RNN extracts more complex features. However, these advanced RNN structures require more fine-tuning to achieve better performance than their less complex counterparts. Moreover, training these networks is more demanding computationally.

Several studies have already demonstrated the suitability of RNN structures for UAVs. Sadhu et al. [26] proposed a complex CNN-BILSTM network to detect the anomalous behavior of a quadcopter, after which a traditional fully connected neural network classifies the cause or fault leading to this behavior. Another application of LSTM networks was investigated by Wang et al. [27]. The authors applied an LSTM network to create an estimation of gyroscope sensor data and used this to detect gyroscope drift and bias. The residuals between the true and estimated data are smoothed to mitigate effects of noise. Once drifted or biased sensor data are detected, the estimated sensor data are used to replace this.

Nevertheless, for the studied LOC prediction problem, it is unclear how obvious any precursors to LOC are, how long any temporal dependencies last, and if these are even observable through the limited onboard sensor information. Consequently, four network architectures are applied to anticipate LOC to determine the complexity of the problem and which of these network configurations is most equipped to anticipate LOC. The considered networks are an LSTM network, a bidirectional LSTM (BILSTM) network, an LSTM network preceded by a CNN (CNN-LSTM), and a GRU network.

III. Methodology

To effectively train the RNNs, the moment of LOC should be established in the flight data. For this, a suitable definition of LOC is necessary. In this research, LOC is defined as the moment in time wherein the magnitude of the pitch or roll angle exceeds 90 deg. Although seemingly arbitrary, the motivation behind this definition will be clarified in the subsequent subsections, which outline the flight tests conducted, the RNN configurations, and data processing.

A. Flight Tests

Flight data are obtained through 172 real-world flight tests. Of these, 144 flights are conducted using an Eachine Trashcan Tiny Whoop 75 mm quadcopter equipped with a JHEMCU SH50 flight controller hosting 8.0 MB of onboard memory dedicated to flight log storage. These logs contain Inertial Measurement Unit (IMU) sensor data, rotor commands, and true rotor outputs made available through the bidirectional DSHOT (Digital shot) protocol. The controller update rate is 4 kHz, whereas the logging rate is 1 kHz. The maximum power output of the motors is limited to 50% of the nominal output to emulate the lower thrust-to-weight ratios of larger UAVs. In addition, the 50% limitation ensures the practicality of conducting the LOC events manually in our motion-capturing laboratory by keeping the vertical acceleration (at the maximum commanded yaw rate) close to zero. Without the limitation, even higher yaw rates can be obtained, but the additional thrust produced by the active diagonal pair of motors would quickly accelerate the drone out of the motion-capturing facility. Figure 1 depicts the employed Tiny Whoop, and Table 1 summarizes its characteristics.

During the experiments, the quadcopter is deliberately brought into a LOC condition by demanding an excessively high yaw rate of ± 2000 deg/s. Although at first the vehicle rotates around its

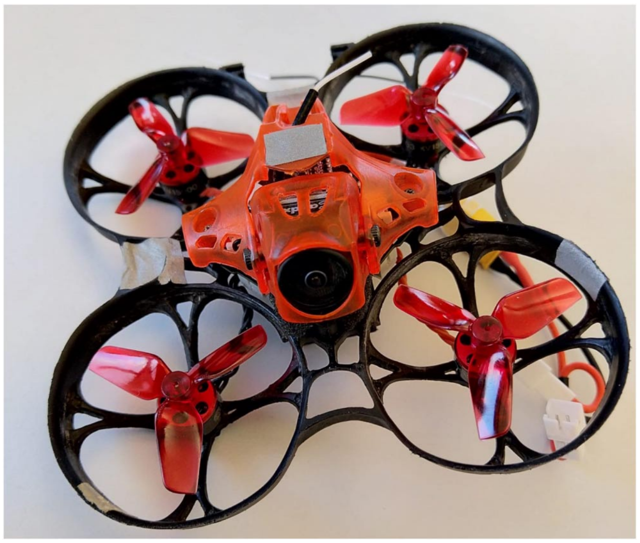


Fig. 1 The black-box-equipped 75 mm, 56 g Tiny Whoop used during experiments.

Table 1 Tiny Whoop characteristics

Characteristic	Details
Mass including batteries	56 g
Axis-to-axis diameter	75 mm
Four-blade propeller diameter	40 mm
Three-blade propeller diameter	35 mm
Motor	TC0803 with 15,000 kV
Batteries	Two 1S (one cell) BETAFPV with 4.35 V and 300 mAh
Flight controller (FC)	JHEMCU SH50 F4 2S with 8.0 MB black box
FC software	Betaflight 4.2

z axis as desired, it quickly begins oscillating around the x (roll) and y (pitch) axes, wherein the amplitude of the roll and pitch angles oscillate with increasing magnitude. Once the x- and y-axis rotations reach an amplitude of 90 deg, the quadcopter makes a sudden, and unpredictable, aggressive turn around one of these axes, after which it is unrecoverable. The crux for correct LOC prediction is to consistently label the moment in time where LOC starts. From visual inspection of both the flights and data, this moment is identified as that where the quadcopter cannot continue rotating purely around its z axis, and where the roll or pitch rotations become dominant. Therefore, the onset of LOC is defined as the moment where the absolute value of either the pitch or roll angle first exceeds 90 deg and continues to increase in magnitude thereafter.

It should be emphasized that this is a specific definition tailored for the considered LOC scenario. Because this research is primarily concerned with determining if LOC can be predicted through sensor data at all, this tailored definition is chosen as a starting point. This definition may then be extended to a more generic case, should the proposed methods prove fruitful in anticipating LOC. For example, a candidate generalized definition may relate to the time to recovery from a particular state, which is more in line with reachability-analysis-based LOC prediction methods; see, e.g., Refs. [10,28]. Another option would be to create a bank of neural networks tailored for one or more scenarios.

In addition to identifying the best-performing RNN architecture for the LOC prediction problem, it is also appealing to investigate their generalization capabilities. For this purpose, different flight tests are performed:

1) In test 1, the quadcopter is brought into LOC in its standard configuration; i.e., no additional weight is added, the default four-blade propellers are used, and it is under windless conditions.

Table 2 Total quadcopter mass per run

	Mass, g
Nominal	56
Run 1	62
Run 2	66
Run 3	72
Run 4	76
Run 5	82

The majority of the flights (74) are executed in this way to collect sufficient data for training.

2) Test 2 only alters the mass of the quadcopter. Five flights are executed with different masses as compared to the nominal quadcopter. These mass changes per run are summarized in Table 2. The LOC scenario remains unchanged.

3) In test 3 the flight characteristics of the quadcopter are changed by using different propeller types. Six flights are performed using 35 mm three-blade propellers instead of 40 mm four-blade propellers.

4) Test 4 investigates if networks trained on LOC due to high yaw rate can generalize to other LOC events. Fifty-three flights are performed, leading to a similar LOC event by demanding an excessive pitch rate of ± 2000 deg/s. The longest two runs are chosen to check the generalization capabilities.

5) In test 5, the generalization performance for different operating conditions is tested. Five flights are performed during which the quadcopter flies in a wind stream. LOC is again achieved by demanding an excessive yaw rate.

6) In test 6, lastly, to test to what extent the algorithm can be transferred from one vehicle to another, 28 flights are executed with two other quadcopters. During this test, four flights are performed with the URUAV UZ85 quadcopter, which is approximately 1.5 times heavier than the Tiny Whoop; and 24 flights are performed with the GEPRC CineGO, which is almost five times as heavy as the Tiny Whoop. The details of these quadcopters are summarized in Table 3.

All considered networks are trained using data obtained from test 1. After training, data from other tests are fed into the networks to evaluate which network is most adept at generalization for different conditions. During the training phase, 80% of the data are used for training and 20% are used as a test set. This division is done randomly for training each model.

B. Neural Network Architecture

The main goal of the neural networks is to predict the amount of time, in seconds, left until the LOC event starts. However, to avoid false predictions during nominal flight, another branch is added to all networks that detects dangerous maneuvers. This leads to the configuration shown in Fig. 2, wherein the four different architectures considered in this research are illustrated.

For the LSTM, BILSTM, and GRU network, the first three layers are not used; instead, their first two layers are labeled through the LSTM, BILSTM, and GRU cells, respectively. For the CNN-LSTM

Table 3 URUAV UZ85 and GEPRC CineGO characteristics

Characteristic	URUAV UZ85	GEPRO CineGO
Mass including batteries	73 g	265 g
Axis-to-axis diameter	85 mm	361 mm
Propeller diameter	52.17 mm	76.5 mm
Motor	1102 with 10,000 kV	Emax Eco 1407 with 3300 kV
Batteries	Two 1S (one cell) BETAFPV with 4.35 V and 300 mAh	One 4S (four cell) Tattu R-Line with 14.8 V and 550 mAh
Flight controller (FC)	JHEMCU SH50 F4 2S	MATEKSYS F722-mini 2-8S
FC software	Betaflight 4.2	Betaflight 4.2

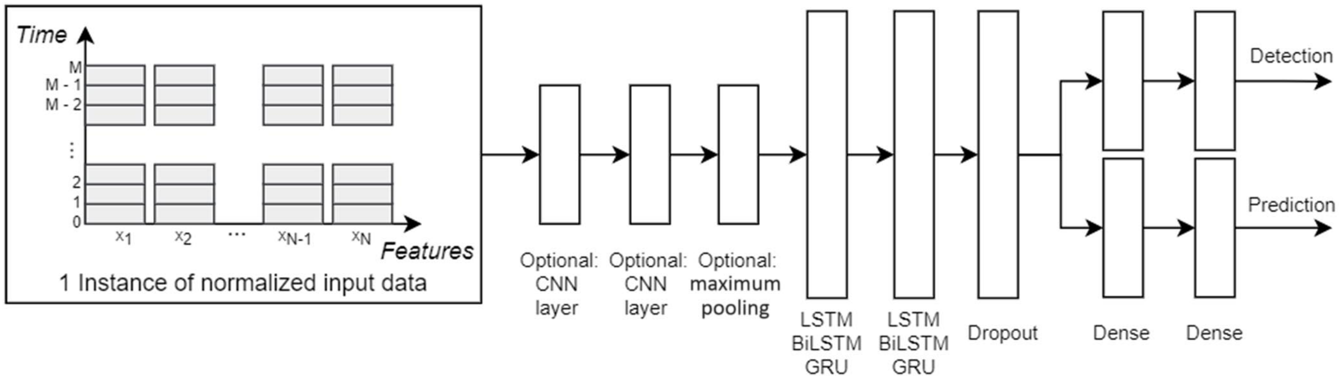


Fig. 2 Architectures of the different recurrent neural networks used in this research.

network, the first three layers are used and the following two layers consist of LSTM cells. In all architectures, the first and second RNN layers contain 50 and 100 neurons, respectively. All networks are followed by a dropout layer with a dropout rate of 0.1, which is a regularization method to mitigate overfitting on the training set and promote better generalization [29].

Subsequently, the networks are split into two branches, with both containing two dense layers. The last layer outputs a floating-point value. For the detection branch, the label is either *0* or “1”: 0 indicates that the quadcopter is in nominal flight, and 1 specifies that a dangerous maneuver is being flown that ends in a LOC event. For the prediction branch, the label is “-1” before the start of the dangerous maneuver. Once this maneuver has started, the label changes to the time in seconds left until the LOC event starts. The output of this branch only becomes relevant once the detection branch detects a dangerous maneuver. Examples of the individual outputs of these branches, along with their aggregated output, are shown in Fig. 3. Note that in Fig. 3c, the left y axis belongs to the detection branch and the right y axis relates to the prediction branch.

The performance of the trained networks is assessed using the root-mean-squared error (RMSE). This metric represents the average error between the model’s predicted and true times to LOC. Thus, the lower the RMSE value, the better the network performance. The benefit of using this metric is that the unit is the same as the quantity that is being estimated, which is the time in seconds for the LOC prediction problem. The RMSE value is calculated through

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2} \tag{1}$$

where $\hat{Y}_i$ represents the model’s estimated time to LOC, and Y_i is the actual time to LOC. Furthermore, n is the total number of time steps, or data points, present in a failure run.

It should be noted that, although different network hyperparameter configurations were experimented with in the preliminary phase of the research, the optimization of these hyperparameters is out of the scope of the present study. Here, we are more concerned with the feasibility of an RNN-based LOC predictor. Consequently, such hyperparameter optimization is not a focus of this paper but rather an avenue of future work.

C. Data Processing

Data obtained during the experiment must be processed before they are fed into the networks to improve prediction performance and to reduce training and inference time. Inference here refers to using a trained neural network on a new, unknown dataset to get a prediction [29].

To minimize the training time, the flight data are truncated about the dangerous maneuver that induces the LOC event. The onset of this maneuver is identified by observing the pilot stick input commands, and data within 5 s preceding this input are preserved as a lead up to the maneuver, as is visible in Fig. 3. Data after LOC are also removed.

Another way to reduce the training and inference time is by limiting the number of network inputs. Thus, only sensor measurements that provide useful early warning signals for LOC should be selected. To do so, all variables that are logged during flight are plotted, and those that exhibit a trend preceding LOC are chosen. These are: 1) rotor commands for each of the four rotors ($\omega 1_{\mathrm{cmd}}$, $\omega 2_{\mathrm{cmd}}$, $\omega 3_{\mathrm{cmd}}$, and $\omega 4_{\mathrm{cmd}}$), 2) true rotor outputs for each of the four rotors ($\omega 1_{\mathrm{true}}$, $\omega 2_{\mathrm{true}}$, $\omega 3_{\mathrm{true}}$, and $\omega 4_{\mathrm{true}}$), 3) smoothed accelerometer measurements (a_x, a_y, and a_z), 4) measurements from the gyroscope (roll rate p, pitch rate q, and yaw rate r), and 5) quadcopter attitude (heading roll, heading pitch, and heading yaw).

Figure 4 illustrates the responses of the selected variables from the start of the dangerous maneuver until the onset of the LOC event for one of the failure runs, i.e., flights. The difference in behavior between the rotor command and the true rotor output is due to the electronic speed controller. This device adjusts the rotor commands, which are outputted by the BetaFlight PID (proportional–integral–derivative) controller, to the true revolutions per minute of the rotors. To investigate which variables show the clearest early warning

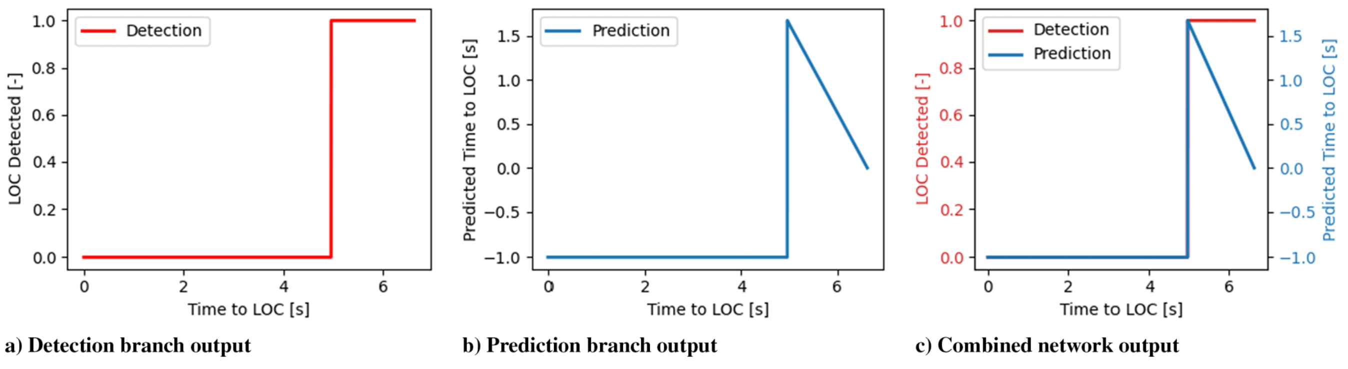


Fig. 3 Output of the two network branches and combined output. The first 5 s are nominal flight: after which, a dangerous maneuver leads to loss of control after 1.7*s*.

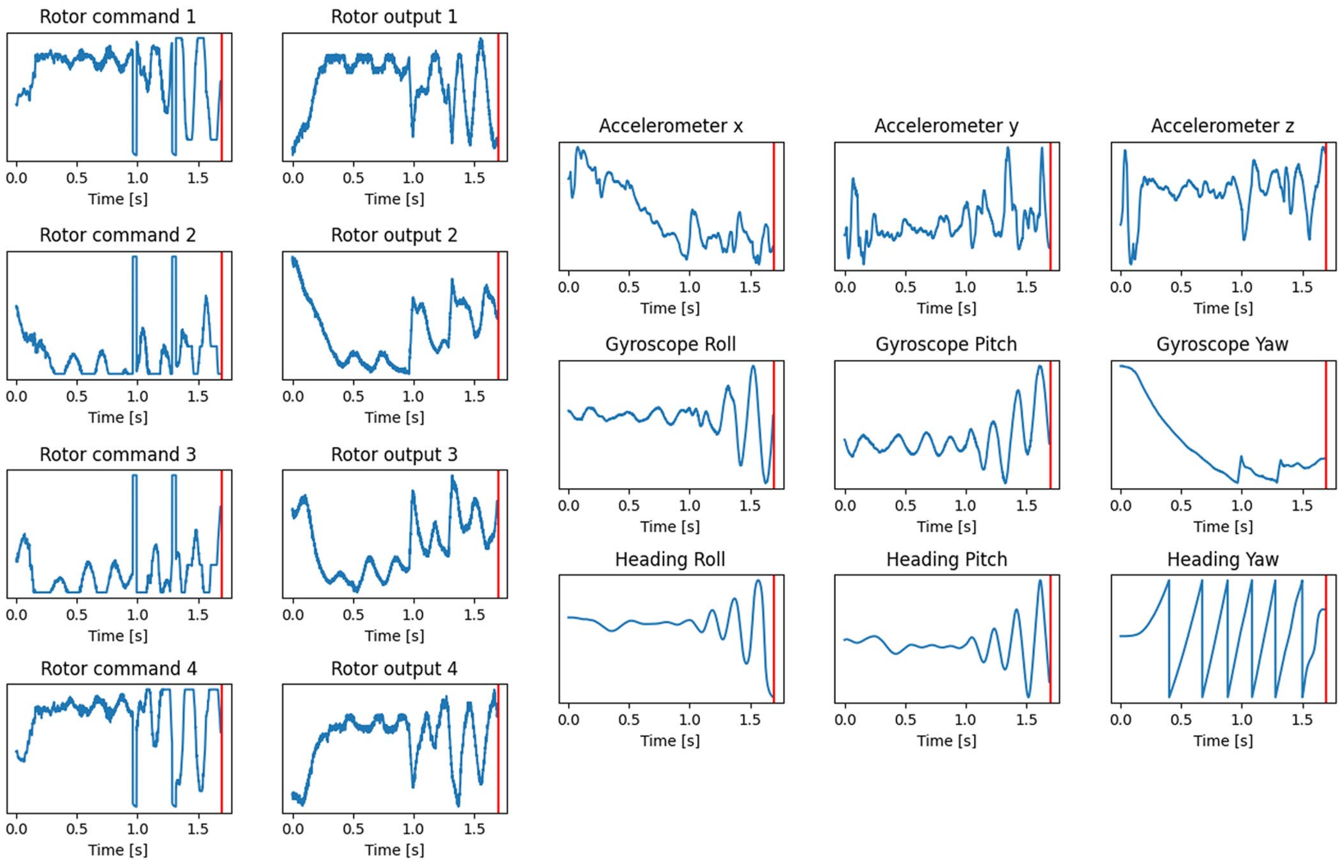


Fig. 4 Variables showing a trend before loss of control. The red line indicates the time where LOC starts.

signals for LOC, a variety of combinations is used during the training phase to find the most optimal mix of variables. In case of no clear preference for specific combinations, fewer variables are preferred to minimize processing and inference time.

Additionally, to mitigate the vanishing gradient problem [29], the input data are standardized using the Z Score, which is calculated according to

$$x_i' = \frac{x_i - \mu}{\sigma} \tag{2}$$

where μ represents the mean and σ the standard deviation. To ensure that the proposed algorithm can run in real time, these values are determined using all failure runs obtained during test 1.

Apart from using advanced memory cells, there is another way to stimulate the network to identify time-dependent features. The data are cut into chunks of window size M with a step size of one. For example, the first chunk contains all data points of all used variables from time $t = 1$ until $t = M$, whereas the second chunk contains all data points from time $t = 2$ until $t = M + 1$. The associated time to loss of control is defined as the time from the last data point present in the chunk until LOC. The window size is set to 20.

To further promote the prediction capacity of the networks, additional variables can be defined. It is evident from Fig. 4 that rotor saturation occurs before LOC for most flights. It is therefore hypothesized that including information about rotor saturation improves prediction performance. Thus, an additional feature, which is derived from the rotor commands, is designed that counts the number of saturated rotors at each moment in time. This is done as follows:

$$\mathrm{NumSat} = \sum_{i=1}^{4} z_i$$
$$z_i = \begin{cases} 0 & 165 < \omega i_{\mathrm{cmd}} < 1000 \\ 1 & \omega i_{\mathrm{cmd}} \leq 165 \vee \omega i_{\mathrm{cmd}} \geq 1000 \end{cases} \tag{3}$$

where the mentioned values are based upon the output range of the rotor commands, which lies between 118 and 1048. A value of 118 means 0% throttle, and a value of 1048 means 100% throttle. A margin is included in case the commands remain close to these values but are not exactly equal to them, leading to a saturated rotor for commands lower than 165 or higher than 1000.

In addition to preprocessing data, the results outputted by the networks are also postprocessed to improve performance. The detection branch outputs a value in the range $(-\infty, +\infty)$, where a negative value represents a nominal flight and a positive value represents a dangerous flight. This output is mapped to 0 or 1. By default, this value is mapped to 0. Once the output of this branch is a positive value for 20 consecutive time steps, equal to 20 ms, the output is mapped to 1. This is switched back to 0 if the output of the branch is a negative value for 20 successive time steps. These 20 ms are chosen such that it equals the window size M mentioned earlier.

The output of the prediction branch acts like a switch and is based upon the output of the detection branch according to

$$Y_{\mathrm{Pred}} = \begin{cases} -1 & Y_{\mathrm{Det}} = 0 \\ \mathrm{Prediction} & Y_{\mathrm{Det}} = 1 \end{cases} \tag{4}$$

where Y_{Pred} equals -1 in case nominal flight is detected and equals the output of the prediction branch otherwise.

IV. Results

All four network types are trained 10 times on 74 runs from test 1 flights using different combinations of parameters outlined in Sec. III.C and shown in Fig. 4, leading to 40 trained models in total. However, because random processes are involved in training, this is done three times, culminating in 120 models. The performance of each model is assessed using one validation failure run from test 1 that was not in the training set. During this validation run, the LOC event began 1.7 s after the dangerous maneuver initiated.

The performances of the models are grouped in three ways to compare them based on different characteristics. First, they are categorized by model type to identify which architecture performs best. Second, they are grouped by the parameter combination employed during training to investigate which variables are the most informative early warning signals for LOC. Lastly, the results of the generalization scenarios are presented to determine the generalization capabilities of different model architectures and variable combinations.

A. Model Performance

Figure 5 compares the RMSE, in seconds, achieved by each of the model architectures when applied to the validation LOC flight run.

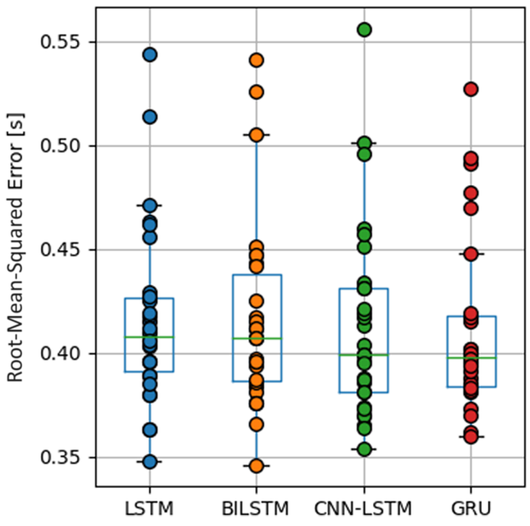


Fig. 5 RMSE values in seconds for different model types.

Each model type has 30 data points, due to 10 parameter combinations for three trials.

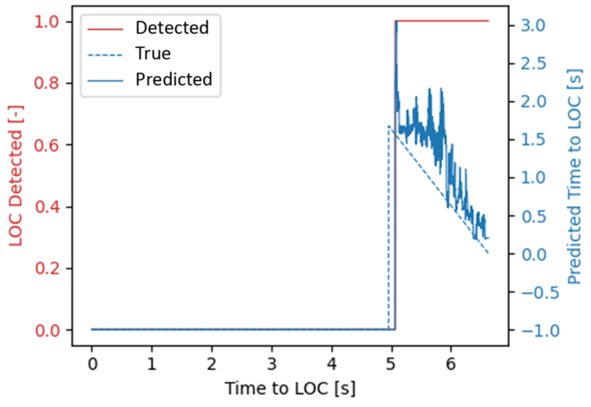
Each model shows a few high error outliers. These are all related to one specific variable combination, which will be addressed in the subsequent subsection. Besides this, the GRU model shows more outliers with high values in comparison to the other configurations. However, the associated RMSE values are comparable to the values within the whiskers of the other models. All other values are within the interquartile range or within the whiskers, which is 1.5 times the interquartile range.

Moreover, all the median values, denoted by the green lines in the boxplots, are similar and are all within each other's interquartile range. Furthermore, the spread around the median is comparable for different model types. Consequently, it is clear that the results are inconclusive on which model performs best with respect to the validation run. As a result, all types will be checked for their generalization capabilities to identify if one type is preferred above another.

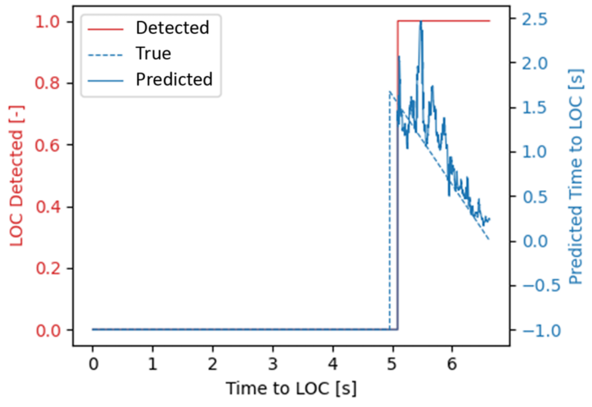
To get an impression of the average prediction performance, the run for which the RMSE value is closest to the median for the model is plotted in Fig. 6. The parameter combination used for these runs is also reported. Evidently, all model types are able to detect the maneuver leading to LOC correctly. Furthermore, although the time to LOC is overestimated at the beginning of the maneuver for all models, all predictions show a downward trend, indicating that they observe stronger failure features closer to the LOC event.

B. Variable Performance

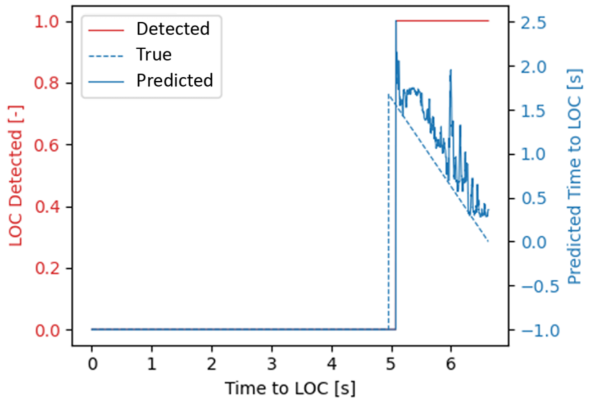
Reminiscent of the model architecture comparison, Fig. 7 depicts the differences in RMSEs for various variable combinations employed during training. Each combination has 12 data points, which are the RMSE values for the four model types and three random training processes. Note that these are the same data that are used in Fig. 5, only grouped differently.



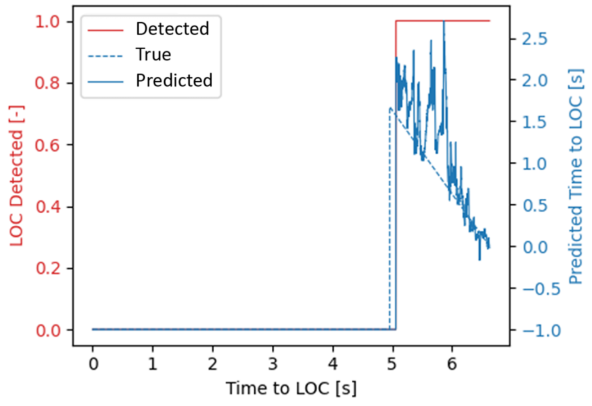
a) LSTM, 0.406 s ω_{cmd}



b) BiLSTM, 0.407 s ω_{cmd} and ω_{true} and Accelerations



c) CNN-LSTM, 0.396 s ω_{cmd} and Accelerations



d) GRU, 0.399 s ω_{cmd} and ω_{true} and Heading

Fig. 6 Best performance for each model on the validation run including RMSE value and used variables.

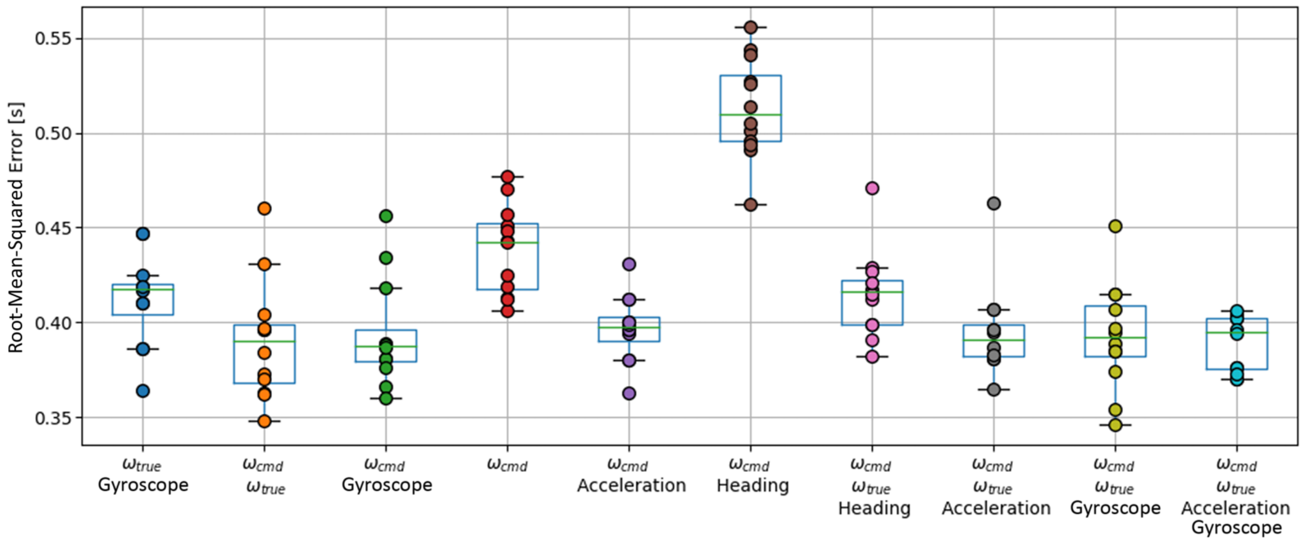


Fig. 7 RMSE values in seconds for different variable combinations.

Unlike the model type comparison, it is possible to observe preferences for specific parameters. From left to right, it is clear from the first three boxes that the inclusion of the commanded rotor speeds boost performance. However, using this variable alone is insufficient. Likewise, from the sixth box, the inclusion of the heading appears to be detrimental to performance, which is confirmed by the seventh box. In fact, these heading runs are responsible for the high RMSE outliers observed in Fig. 5.

Furthermore, the results shown in Fig. 7 suggest that combining more parameters does not necessarily improve performance. This leads to preference for models reliant on only two variables: one of which should be the commanded rotor values. The combination with the true rotor speed outputs is selected for the subsequent generalization performance analysis because the average RMSE value of this combination is the lowest.

C. Generalization Performance

Five different scenarios are tested where one condition is changed as compared to the flights from test 1. As mentioned earlier in this paper, only the generalization performance of the networks trained on commanded rotor values and true rotor outputs will be assessed in detail. However, because there are scenarios where there is a clear difference in generalization due to the used variables, these will be examined where relevant.

1. *Changing Mass*

The nominal mass of the Tiny Whoop including batteries is 56 g. Five flights are performed where mass is added to the quadcopter by attaching coins to the frame. Because it is assumed that the center of mass is in the middle of the quadcopter, attaching the coins here minimizes changes to the moment of inertia. The resulting RMSE values of the considered models applied to the different masses are presented in Fig. 8. Because each model is trained three times to cancel out the randomness of the training process, the presented RMSE values are the averages per model per mass. The average RMSE values achieved on the validation run for the nominal quadcopter of 56 g are depicted for reference.

It is expected that for increasing mass, the prediction will be less accurate due to a greater difference in the dynamic behavior of the quadcopter. This trend is observed for all models, except for the run with a mass of 76 g. This can be explained by looking at the time between the start of the dangerous maneuver and LOC. This is between 2.1 and 2.4 s for all runs, but is only 1.67 s for the run with a mass of 76 g. This lower time to LOC leads to a lower RMSE score.

Although Fig. 8 provides an overview of the performance of the models with respect to a mass change, it does not describe how the time to LOC evolves after the start of the dangerous maneuver.

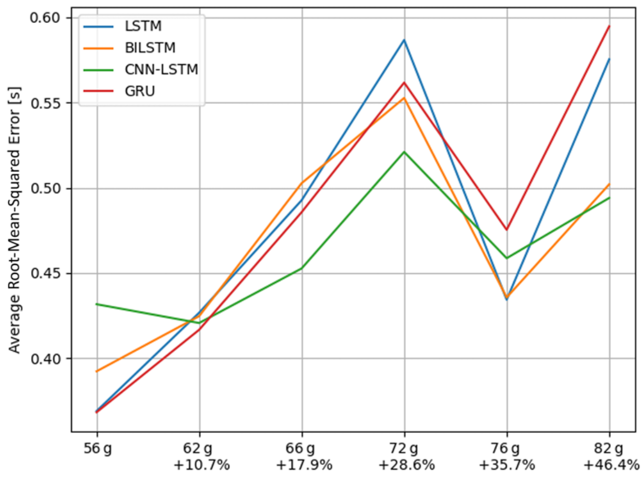


Fig. 8 Average RMSE values in seconds for different quadcopter masses.

Instead, it is necessary to look at the individual prediction plots. Figure 9 illustrates the prediction response of the BILSTM model for the heaviest mass case. The BILSTM model is chosen because the RMSE of 0.501 s is near the median performance between models. The detection of the dangerous maneuver still works as desired, and the prediction is lower than the true time to LOC for the majority of the run. Moreover, there still is a downward trend ending close to 0 s at the onset of the LOC event.

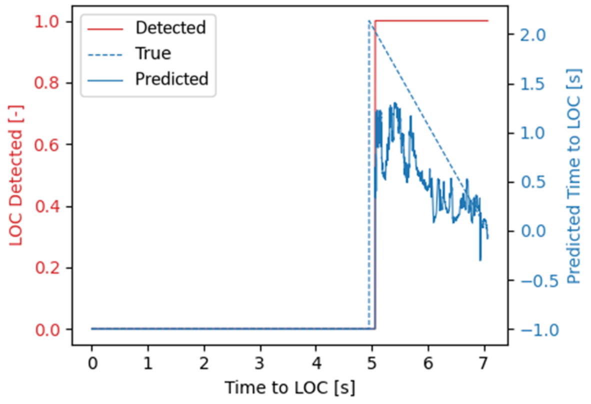


Fig. 9 Prediction of the BILSTM network on the heaviest quadcopter run.

2. *Different Propellers*

In the default configuration, the Tiny Whoop is equipped with 40 mm four-blade propellers. To investigate whether different propellers influence the LOC prediction capabilities, these were replaced by 30 mm three-blade propellers. The yawing moment authority of the smaller propellers is significantly lower, and hence the time to spin up to high yaw rates is longer, which extends the time to LOC. In total, six flights are performed with these propellers. Figure 10 visualizes the RMSE values for the first flight using the three-blade propellers for different variable combinations because there is a clear difference in RMSE values between them.

On average, models trained using the true rotor output experience higher RMSE values than those that do not. Therefore, to assess the suitability of the networks for LOC prediction with different propellers, the models trained on the commanded rotor values and true rotor outputs will not be used. Instead, models trained on commanded values only are considered because these show the best performance in Fig. 10. The average RMSE values for the different models trained only on the commanded rotor values across all the propeller generality flights are summarized in Table 4.

It is clear that there is a large variation in performance between the six flights. This can be explained by looking at the time between the start of the dangerous maneuver and LOC, summarized in Table 5. The longer the time to LOC, the higher the RMSE score. Furthermore, as expected, the time before the onset of the LOC event is longer with the new (smaller) propeller than the time to LOC in the validation run.

To investigate why there is such a variation in time to LOC, the prediction of one of the models on the second and worst propeller run is analyzed. Figure 11 depicts the prediction of the BILSTM model, which scored the median RMSE value of 1.66 s on this run. Once the dangerous maneuver is detected correctly, the time to LOC prediction instantiates at around 2 s. Subsequently, the characteristic downward prediction behavior is visible. The reason the time to LOC caps at around 2 s lies in the fact that the average time to LOC in the training set is 1.980 s. The trained networks are not familiar with runs that take much longer and can therefore fail to produce accurate prediction results for these runs. A longer time to LOC leads to a higher RMSE value because the difference between the prediction and the true time to LOC is higher when compared to shorter runs.

Table 5 Time to loss of control in seconds

	Time to LOC
Nominal	1.700
Propellor 1	4.007
Propellor 2	5.507
Propellor 3	2.142
Propellor 4	4.721
Propellor 5	2.645
Propellor 6	2.027

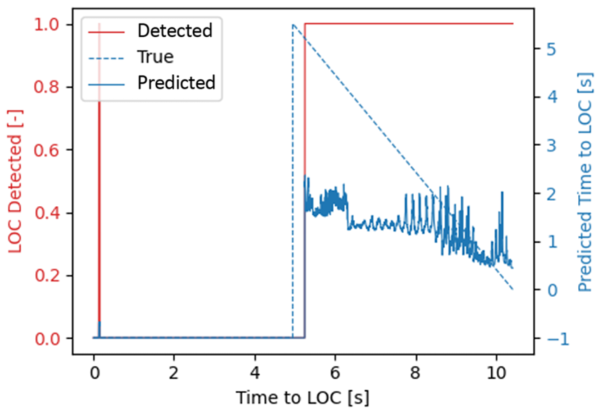


Fig. 11 Prediction of the BILSTM network on the second propeller run.

3. *LOC Due to High Pitch Rate*

Another evaluated scenario is whether the models can generalize to different LOC events. Instead of instigating LOC by demanding a high yaw rate, it is induced by demanding a high pitch rate. In this scenario, LOC is defined similarly as for the nominal scenario. Data

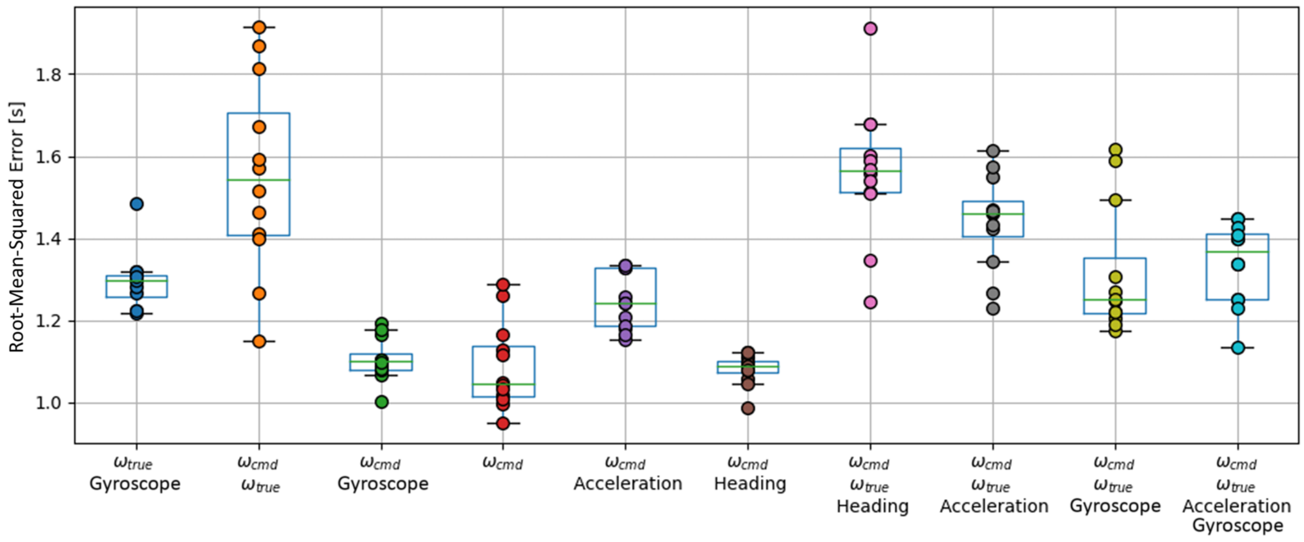


Fig. 10 RMSE values in seconds for different variable combinations on the first flight with three-blade propellers.

Table 4 Average RMSE values in seconds on different propeller flights using models trained on commanded rotor values (Props, propeller)

	Nominal	Props 1	Props 2	Props 3	Props 4	Props 5	Props 6
LSTM	0.369	1.043	1.637	0.466	1.260	0.560	0.567
BILSTM	0.392	1.044	1.670	0.475	1.262	0.549	0.576
CNN-LSTM	0.432	1.239	2.375	0.514	1.516	0.611	0.653
GRU	0.368	1.028	1.642	0.562	0.921	0.527	0.634

analysis shows that the quadcopter flips around its pitch axis for a few rounds, after which it starts to flip around its roll axis. LOC is defined as the moment in time where this flip over the roll axis initiates because this is the moment where it stops behaving the way it should. This happens faster than for the flights from test 1: on average, the time to LOC for the pitch runs is 0.470 s. To aid the prediction of the models, the two runs with the longest time to pitch-induced LOC are chosen, with a time to LOC of 1.1 s for both runs. Even though these favorable runs are used, all networks already fail in detecting the dangerous maneuver correctly.

4. *Flying in Wind*

To simulate the effects of wind, a Master DF 30P fan is used. This fan has a constant airflow of $10,200\ \mathrm{m}^3/\mathrm{h}$, resulting in an air velocity of $6.25\ \mathrm{m/s}$ given the fan diameter of 760 mm. Five flights are performed wherein wind is blowing on the quadcopter. The wind is coming from behind with respect to the quadcopter coordinate system (i.e. from the negative x axis direction towards the positive x axis direction), and at a small angle from beneath. During the first three flights, the quadcopter is flown into the stream before the maneuver starts, emulating constant wind conditions. During these flights, it is noted that the quadcopter is pushed away and upward by the stream. For the last two flights, the maneuver is started before it is flown in the stream, which can be seen as a wind gust acting on the quadcopter. Although the vehicle is again pushed away by the wind, it is influenced for a shorter period of time before the LOC event. It is expected that the wind gust influences the performance less because the wind speed is kept constant across both wind cases. The RMSE values obtained under the various wind conditions are visualized in Fig. 12, wherein runs 1–3 represent constant wind conditions and runs 4–5 denote wind gust conditions.

The prediction performances vary heavily and do not appear to depend on specific wind conditions. Consider, for example, the similar performances between runs 2 (constant wind) and 5 (wind gust). The limited number of tests in these conditions makes it challenging to draw any conclusions pertaining to specific wind conditions.

To explore the suitability of the models for LOC prediction in general windy conditions, the prediction response plots are assessed. The third flight is chosen, which has the highest RMSE value, and therefore represents the worst condition. Figure 13 illustrates the prediction of the GRU network on this flight, which scored the median RMSE value of 0.745 s. The detection shows a false positive after 2 s, which is likely caused by turbulent flight due to the presence of wind. Indeed, another false positive precedes the correct detection of the dangerous maneuver. Although the false positives are something to consider for practical implementations of the LOC predictors discussed here, developing a false positive mitigation strategy is out of the scope of the present work. Furthermore, the prediction branch is primarily overestimating the time to LOC and oscillates more as

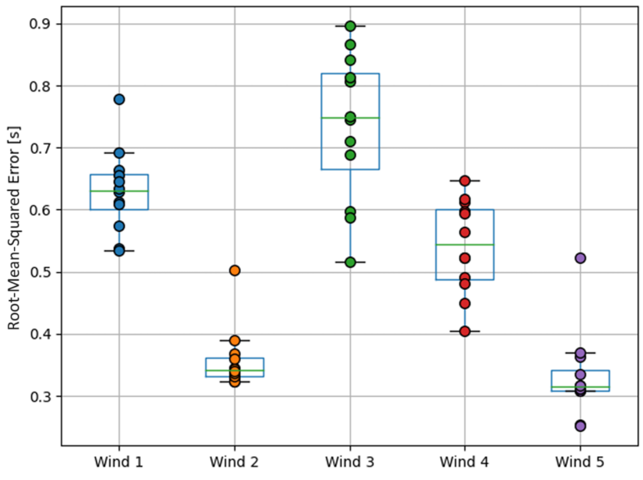


Fig. 12 RMSE values in seconds for different runs in wind. Runs 1, 2, and 3 represent constant wind; and runs 4 and 5 represent wind gusts.

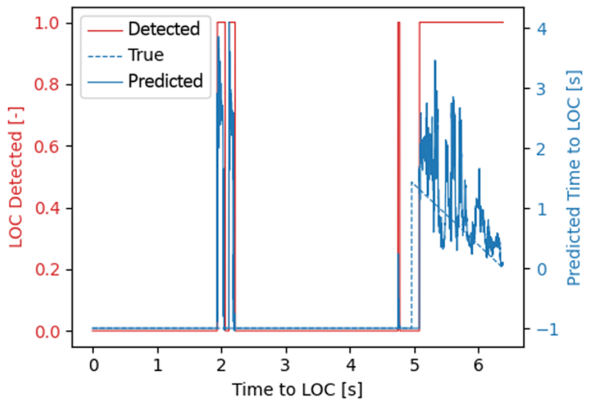


Fig. 13 Prediction of the GRU network on the third run in windy conditions.

compared to the nominal case. Nonetheless, the characteristic downward trend remains visible.

5. *Different Quadcopters*

Ideally, the developed LOC predictors may be applied to any quadcopter. As a data-driven approach, it is often undesirable to collect LOC data for new quadcopters. Hence, the final test aims to investigate the performance of the models trained on the Tiny Whoop when applied to other (larger) quadcopters. To this end, four yaw-induced LOC flights with the URUAV UZ85 quadcopter and 24 flights with the GEPRC CineGO quadcopter are evaluated. The UZ85 has a (diagonal) length of 85 mm and weighs 73 g at takeoff, whereas the CineGO has a (diagonal) length of 361 mm and weighs 265 g. Both quadcopters are induced into LOC similarly to the flights from test 1 described in Sec. III.A. It is interesting to note that even though both quadcopters are larger and up to five times heavier than the Tiny Whoop, they show the exact same LOC behavior. This reinforces the relevance of the experiments conducted with the Tiny Whoop, even for larger quadcopters. Nevertheless, there remains a key difference: the average time to LOC varies per quadcopter. For the Tiny Whoop, this is 1.98 s; whereas for the CineGO, this is 2.68 s. It is therefore expected that networks trained on data from the Tiny Whoop will underestimate the time to LOC for the other two quadcopters, resulting in larger RMSE values than on the original nominal test run. Next to this, another hypothesis is that the results will differ for different parameter combinations.

The reason for this is the standardization step that is taken during preprocessing. Namely, a mean and standard deviation are determined using data from the flights performed during test 1 of the Tiny Whoop. Because the range of true rotor outputs differs between the Tiny Whoop, UZ85, and CineGO, this standardization step will incorrectly scale the true rotor speeds of the UZ85 and CineGo. Indeed, we observe a poor performance of models that include the true rotor speed output as a variable when looking at the RMSE scores for all different variable combinations in Fig. 14. Another observation is that the RMSE values are higher for the CineGO than for the UZ85. This is in line with expectations due to its longer time to LOC in comparison to the UZ85. Therefore, the models tend to underestimate this time to LOC more for the CineGO than for the UZ85, leading to higher RMSE values overall.

To find out if these expectations are true, the prediction results on one flight of both drones will be compared. For the UZ85 quadcopter, a run with a model trained on commanded rotor values only will be chosen; whereas for the CineGO, one run with a model trained on both the commanded rotor values and gyroscope measurements is taken. These are shown in Figs. 15 and 16, respectively. These plots confirm that for the CineGO, the model underestimates the time to LOC, resulting in a high RMSE value. Furthermore, a comparison of both plots provides the insight that higher RMSE scores do not always mean that the performance is worse. Even though there is an underestimation for the CineGO, there is no false positive detection; and the prediction is more stable than for the UZ85.

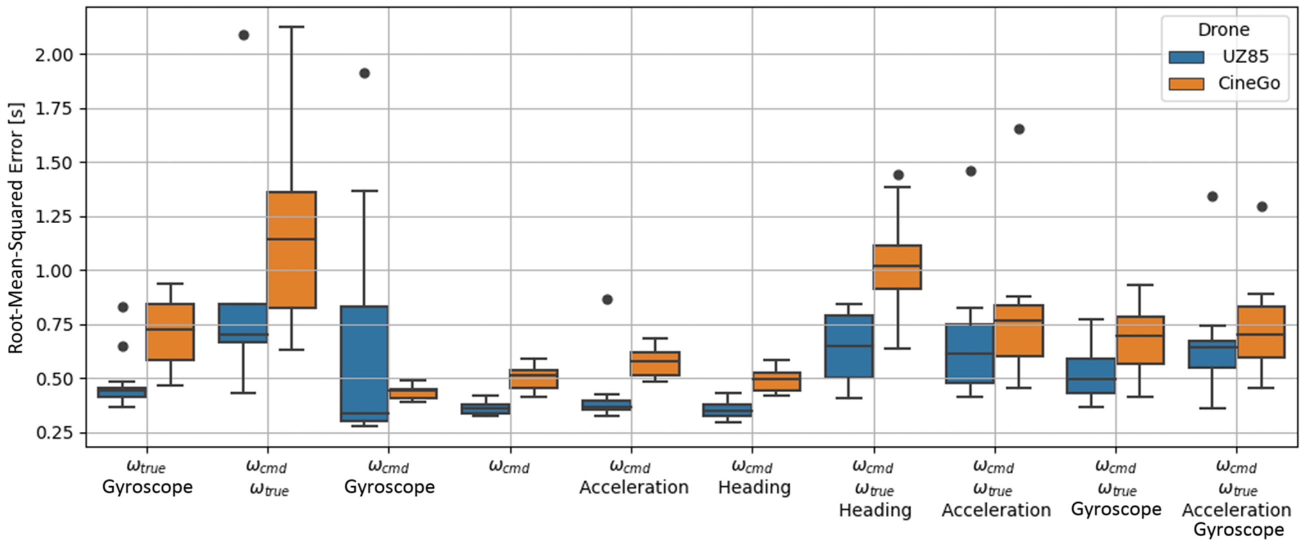


Fig. 14 Box plots created using the RMSE values of one run of the UZ85 and one run of the CineGO.

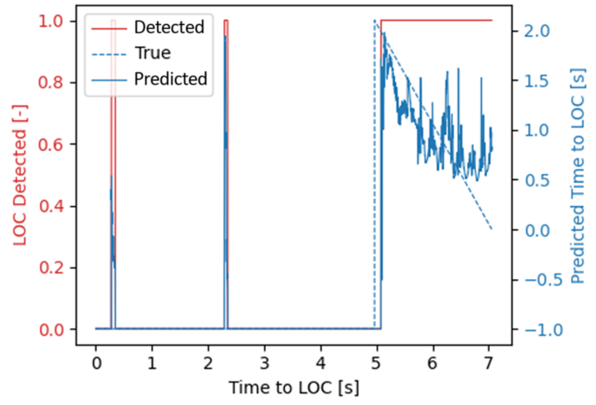


Fig. 15 Prediction of the GRU network on the second UZ85 run.

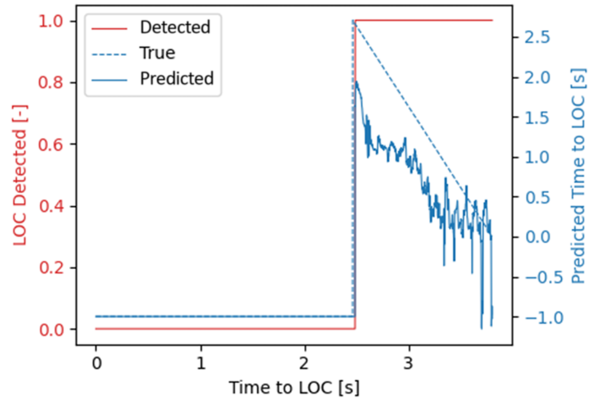


Fig. 16 Prediction of the GRU network on the 14th CineGO run.

V. Discussion

Combining the results outlined in the previous section leads to new findings about early warning signals, the suitability of RNN architectures for LOC prediction, and the application of these models.

A. Loss-of-Control Early Warning Signals

Of the considered early warning signals, the commanded rotor values are most informative in anticipating an impending LOC event, as noticed in Fig. 7. This is an interesting result because it implies that LOC is driven by trying to achieve a desired behavior instead of the true behavior of the quadcopter.

This can be explained by comparing the commanded rotor values and true rotor outputs in Fig. 4. The commanded rotor values experience saturation, whereas the true rotor outputs do not hit their minimum or maximum value. This validates the hypothesis stated in Sec. III.C that the rotor command saturation is an informative precursor for LOC. This signal is also recognized by the networks.

Apart from identifying a clear early warning signal, it is also possible to exclude certain variables. For the nominal case, including the heading reduces the prediction performance of all network types, as shown in Fig. 7, and is therefore unfavorable for LOC prediction. Furthermore, for the generality scenarios evaluating different propellers and different quadcopters, models trained on the true rotor outputs show worse performance than other models, as seen in Figs. 10 and 14, respectively. Although the combination of commanded rotor values and true rotor outputs performs best on the nominal run, other variable combinations follow closely, making this combination not necessarily superior to other combinations.

Following this analysis, two combinations turn out to be most useful for LOC prediction: the commanded rotor values together with either gyroscopic or acceleration measurements.

B. RNNs for Loss-of-Control Prediction

Based upon Fig. 6, it can be concluded that RNNs can detect dangerous maneuvers leading to LOC accurately for the scenarios it is trained on. Furthermore, the prediction branch tends to overestimate the time to LOC at the beginning of the dangerous maneuver, after which a clear downward trend can be observed in tandem with the slope of the true time to LOC. Additionally, all network architectures show an average prediction error close to 0.400 s. When using this RMSE value as an uncertainty margin for the output of the network, the prediction provides sufficient information for an (auto) pilot to understand that a LOC event is approaching and at what rate this is happening. This means that RNNs cannot only be used to detect a dangerous maneuver but also to predict a LOC event for the scenarios it is trained on.

Therefore, these results can be used for LOC prevention of quadcopters. The average time to LOC in the training set is almost 2 s. Human pilots need time to process a received warning signal and to take action to recover to safe flight. Two seconds may be too short to do this. Therefore, in further research, the usage of a trained model in a closed-loop control system should be investigated. Combining the outputs of both the detection and prediction branches is most useful to create an algorithm that can override pilot commands just in time.

However, conclusions on what model architecture is most viable for such a LOC prevention system are inconclusive. All model types show a median RMSE value close to 0.400 s on the validation run. Likewise, for the different generalization scenarios, the median values of the different models were also comparable to each other. Despite this, useful insights pertaining to the causes of LOC can be derived. Because both simple and complex RNNs perform similarly, the failure features are less time dependent than originally expected. This means that LOC is likely caused by the more short-term, perhaps instantaneous, behavior of the quadcopter. To confirm these findings, simple RNN structures or even traditional fully connected neural networks should be tested to see if they can still predict LOC correctly.

C. Applications

For both changing mass and different propellers, the detection and prediction behaviors of the models are similar. Even for the worst run encountered for both scenarios, the detection branch still detects the dangerous maneuver correctly, as depicted in Figs. 9 and 11 for changing mass and different propellers, respectively.

In both scenarios, the prediction branch underestimates the time to LOC. For changing mass, this can be explained through additional mass itself. To keep a heavier quadcopter in the air, higher thrust should be generated. The commanded and true rotor outputs will thus be higher, which for the nominal case happens closer to the LOC event, leading to an underestimation. For different propellers, this can be explained by looking at the time to LOC. As was already expected beforehand, the time between the dangerous maneuver and LOC is longer for the shorter 35 mm three-blade propellers. Because the average time to LOC in the training set is only 2 s, the outputted sensor values are associated with a shorter time to LOC by the trained models, leading to an underestimation.

Moreover, based upon the results from these two scenarios, it is clear that the time to LOC influences the RMSE values. For changing mass, the seemingly anomalous flight with a low RMSE despite an increased mass can be explained by a faster time to LOC of 0.5 s lower than the other flights. For different propellers, all flights had longer times to LOC than the ones from the training set, leading to higher RMSE values. These results imply that the application of the trained models is limited to failure runs with similar time-to-LOC as those used to train the models.

On the other hand, for the presented scenarios, the incorrect prediction behavior can be explained and therefore accounted for. When investigating the usage of these models in a closed-loop system, it is advised to incorporate information about the aerodynamic characteristics of the quadcopter and the associated expected deviations of the prediction to the true time to LOC. When these characteristics are changed, the output of the prediction branch can be adjusted according to the expected deviations, which would make these models suitable for LOC prediction under different aerodynamic characteristics.

When looking at the application of the trained models to different LOC scenarios, as is done for the pitch runs in this research, it can be concluded that the approach is limited to known LOC scenarios. This can be explained by looking at how different scenarios alter the sensor outputs of the quadcopter. Indeed, the other considered generalization scenarios do not lead to complete new sensor outputs; similar trends are observable: only with different values. A new LOC scenario shows completely new and potentially unknown behavior, and subsequently unknown sensor outputs. One option to overcome this is to create a model for each LOC event; but for the sake of available memory on the quadcopter, it is worth investigating if one model can be trained for different scenarios.

When flying in wind, the dynamic behavior of the quadcopter is changed more than for different aerodynamic characteristics. When looking at Fig. 13, false positives can be observed, as well as large differences between the predicted and true times to LOC. Furthermore, the predicted time oscillates with high amplitudes as compared to the other scenarios, culminating in unreliable predictions. This is due to turbulence caused by wind, which makes the movements of the quadcopter rather unpredictable. However, the dangerous maneuver can be detected correctly and the prediction still exhibits a downward trend. It is therefore expected that for low wind speeds, the prediction will fall within the error margin of the nominal predictions, which makes these models suitable for LOC prediction but only for limited wind conditions. To confirm this, more flights in windy conditions should be executed.

When flying different quadcopters, the detection branch can still identify the dangerous maneuver correctly, as shown in Figs. 15 and 16. For the UZ85 quadcopter, the prediction branch follows the downward trend clearly but appears to plateau towards the end. Likewise, for the CineGO quadcopter, the downward trend is present as well. However, there is a constant underestimation of the true time to LOC. The downward trend for all runs clearly shows that all quadcopters have similar failure behavior. However, due to a combination of the quadcopter specific time to LOC and the standardization step taken during data preparation, the prediction of the models is inaccurate. Nonetheless, two actions can be taken to overcome this problem. First, the same solution as proposed for changing the mass and propellers can be used. In a closed-loop system, information regarding the quadcopter characteristics can help to adjust for the expected deviations in the prediction branch. Second, it is recommended to look into modifying the standardization step. By using different mean and standard deviation values for different quadcopters, the sensor values might be mapped to the same range as for the first quadcopter. When this is possible, the same model may be suitable for correct detection and prediction of LOC with similar quadcopters.

In combination with the results derived from the windy conditions, some false positives are observable. It should be noted that within this research, there is no tradeoff included between the detection time, false positives, and false negatives. This becomes a relevant topic when such a prediction system is incorporated into a closed-loop system that actively attempts to prevent LOC. It is therefore recommended to further investigate the optimal tradeoff to avoid false positives but still reliably anticipate LOC. Such a closed-loop system would not only benefit the detection branch but also the prediction branch. Moreover, a filter could be included to make the prediction time less noisy.

Aggregating the generality scenarios considered here, it becomes apparent that the prediction branch is biased toward a 2 s prediction. It should be noted that this is not explicitly learned by the network because the time to LOC is not an input. However, it seems that the models are capturing some of the dynamics specifically associated with the Tiny Whoop. Quadcopters in a similar weight class, like the UZ85, show a similar time to LOC in their conventional configurations; however, heavier quadcopters like the CineGO result in a different time to LOC. This has had a negative influence on the generalization capabilities of the neural networks for larger quadcopters. To determine if a more varied dataset results in better generalization performance, it is worth finding failure maneuvers that are less biased toward one specific time to LOC.

Finally, similar algorithms can be created for other (autonomous) vehicles suffering from LOC, where one important requirement is that sufficient failure runs can be performed such that the networks can learn failure features. To reduce the number of necessary crashes, the use of data from simulators or generative models that are capable of mimicking failure data should be investigated. Only when these techniques can create sufficient failure data will it be possible to apply the proposed method on aircraft to make aviation safer.

VI. Conclusions

Loss of control remains the primary cause of crashes for both aircraft and UAVs. To reduce the amount of LOC events, onboard LOC prevention systems should be designed. In this work, it was demonstrated that recurrent neural networks can be used for LOC prediction in quadcopters using only onboard sensor measurements. Training and testing these networks by using data from 172 real-world LOC events showed that the commanded rotor values provide the clearest early warning signals for LOC. The RNNs can predict LOC 2 s before the actual event occurs using these values in combination with either the gyroscopic or accelerometer measurements.

However, the application is currently limited to scenarios that show similar behavior as that used by the training set in terms of the LOC scenario and time to LOC. It was found that for varying aerodynamic and mass or inertia characteristics, the prediction can still be used if the expected deviation in prediction behavior is compensated for. To draw conclusions about the usability of the model in wind conditions, different quadcopters, and for different types of LOC, additional research is necessary.

References

[1] Belcastro, C. M., Newman, R. L., Evans, J. K., Klyde, D. H., Barr, L. C., and Ancel, E., "Hazards Identification and Analysis for Unmanned Aircraft System Operations," *17th AIAA Aviation Technology, Integration, and Operations Conference*, AIAA Paper 2017-3269, 2017. https://doi.org/10.2514/6.2017-3269

[2] *Statistical Summary of Commercial Jet Airplane Accidents, Worldwide Operations | 1959–2020*, Boeing Commercial Airplanes, 2021, http://www.boeing.com/news/techissues/pdf/statsum.pdf [retrieved Nov. 2022].

[3] Belcastro, C. M., and Jacobson, S. R., "Future Integrated Systems Concept for Preventing Aircraft Loss-of-Control Accidents," *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2010-8142, 2010. https://doi.org/10.2514/6.2010-8142

[4] Di Donato, P. F., Balachandran, S., McDonough, K., Atkins, E., and Kolmanovsky, I., "Envelope-Aware Flight Management for Loss of Control Prevention Given Rudder Jam," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 4, 2017, pp. 1027–1041. https://doi.org/10.2514/1.G000252

[5] Balachandran, S., and Atkins, E., "Markov Decision Process Framework for Flight Safety Assessment and Management," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 4, 2017, pp. 817–830. https://doi.org/10.2514/1.G001743

[6] Balachandran, S., and Atkins, E. M., "Flight Safety Assessment and Management to Prevent Loss of Control due to In-Flight Icing," *AIAA Guidance, Navigation, and Control Conference*, AIAA Paper 2016-0094, 2016. https://doi.org/10.2514/6.2016-0094

[7] Rohith, G., "An Investigation into Aircraft Loss of Control and Recovery Solutions," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 233, No. 12, 2019, pp. 4509–4522. https://doi.org/10.1177/0954410019825942

[8] Zhao, Y., and Zhu, J. J., "Automatic Aircraft Loss-of-Control Prevention by Bandwidth Adaptation," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 4, 2017, pp. 878–889. https://doi.org/10.2514/1.G001835

[9] Edwards, C., Lombaerts, T., and Smaili, H., *Fault Tolerant Flight Control: A Benchmark Challenge*, Springer–Verlag, Berlin, 2010.

[10] Schuet, S., Lombaerts, T. J. J., Acosta, D., Kaneshige, J., Wheeler, K., and Shish, K., "Autonomous Flight Envelope Estimation for Loss-of-Control," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 4, 2017, pp. 847–862. https://doi.org/10.2514/1.G001729

[11] Tekles, N., Chongvisal, J., Xargay, E., Choe, R., Talleur, D. A., Hovakimyan, N., and Belcastro, C. M., "Design of a Flight Envelope Protection System for NASA's Transport Class Model," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 4, 2017, pp. 863–877. https://doi.org/10.2514/1.G001728

[12] McDonough, K., and Kolmanovsky, I., "Fast Computable Recoverable Sets and their Use for Aircraft Loss-of-Control Handling," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 4, 2017, pp. 934–947. https://doi.org/10.2514/1.G001747

[13] Stepanyan, V., Krishnakumar, K., Dorais, G., Reardon, S., Barlow, J., Lampton, A., and Hardy, G., "Loss-of-Control Mitigation via Predictive Cuing," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 4, 2017, pp. 831–846. https://doi.org/10.2514/1.G001731

[14] Zhang, Y., de Visser, C. C., and Chu, Q. P., "Database Building and Interpolation for an Online Safe Flight Envelope Prediction System," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 5, 2019, pp. 1166–1174. https://doi.org/10.2514/1.G003834

[15] Zogopoulos-Papaliakos, G., Karras, G. C., and Kyriakopoulos, K. J., "A Fault-Tolerant Control Scheme for Fixed-Wing UAVs with Flight Envelope Awareness," *Journal of Intelligent and Robotic Systems*, Vol. 102, No. 2, 2021, Paper 46. https://doi.org/10.1007/s10846-021-01393-3

[16] Campbell, N. H., Grauer, J. A., and Gregory, I. M., "Loss of Control Detection for Commercial Transport Aircraft Using Conditional Variational Autoencoders," *AIAA SciTech 2021 Forum*, AIAA Paper 2021-0778, 2021, pp. 1–30. https://doi.org/10.2514/6.2021-0778

[17] Kirkendoll, Z., "Automatic Ground Collision Avoidance and Loss of Control Prevention Systems for General Aviation Using Run-time Assurance," *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*, IEEE, New York, 2022, pp. 1–10. https://doi.org/10.1109/DASC55683.2022.9925833

[18] Sun, S., and de Visser, C. C., "Quadrotor Safe Flight Envelope Prediction in the High-Speed Regime: A Monte-Carlo Approach," *AIAA SciTech 2019 Forum*, AIAA Paper 2016-0094, 2019. https://doi.org/10.2514/6.2019-0948

[19] Zhang, Y., Huang, Y., Chu, Q., and de Visser, C., "Database-Driven Safe Flight-Envelope Protection for Impaired Aircraft," *Journal of Aerospace Information Systems*, Vol. 18, No. 1, 2021, pp. 14–25. https://doi.org/10.2514/1.I010846

[20] Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., and Gao, R. X., "Deep Learning and Its Applications to Machine Health Monitoring," *Mechanical Systems and Signal Processing*, Vol. 115, Jan. 2019, pp. 213–237. https://doi.org/10.1016/j.ymssp.2018.05.050

[21] Gers, F., Schmidhuber, J., and Cummins, F., "Learning to Forget: Continual Prediction with LSTM," *Neural Computation*, Vol. 12, No. 10, 2000, pp. 2451–2471. https://doi.org/10.1162/089976600300015015

[22] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y., "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Assoc. for Computational Linguistics, Doha, Qatar, 2014, pp. 1724–1734. https://doi.org/10.3115/v1/D14-1179

[23] Rezaeianjouybari, B., and Shang, Y., "Deep Learning for Prognostics and Health Management: State of the Art, Challenges, and Opportunities," *Measurement*, Vol. 163, Oct. 2020, Paper 107929. https://doi.org/10.1016/j.measurement.2020.107929

[24] Zhang, L., Lin, J., Liu, B., Zhang, Z., Yan, X., and Wei, M., "Review on Deep Learning Applications in Prognostics and Health Management," *IEEE Access*, Vol. 7, No. 1, 2019, pp. 162415–162438. https://doi.org/10.1109/ACCESS.2019.2950985

[25] Fink, O., Wang, Q., Svensén, M., Dersin, P., Lee, W. J., and Ducoffe, M., "Potential, Challenges and Future Directions for Deep Learning in Prognostics and Health Management Applications," *Engineering Applications of Artificial Intelligence*, Vol. 92, June 2020, Paper 103678. https://doi.org/10.1016/j.engappai.2020.103678

[26] Sadhu, V., Zonouz, S., and Pompili, D., "On-Board Deep-Learning-Based Unmanned Aerial Vehicle Fault Cause Detection and Identification," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, New York, 2020, pp. 5255–5261. https://doi.org/10.1109/ICRA40945.2020.9197071

[27] Wang, B., Liu, D., Peng, Y., and Peng, X., "Multivariate Regression-Based Fault Detection and Recovery of UAV Flight Data," *IEEE Transactions on Instrumentation and Measurement*, Vol. 69, No. 6, 2020, pp. 3527–3537. https://doi.org/10.1109/TIM.2019.2935576

[28] Nabi, H., Lombaerts, T., Zhang, Y., Van Kampen, E., Chu, Q., and De Visser, C., "Effects of Structural Failure on the Safe Flight Envelope of Aircraft," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 6, 2018, pp. 1257–1275. https://doi.org/10.2514/1.G003184

[29] Géron, A., *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed., O'Reilly Media, Sebastopol, CA, 2019.

E. Atkins
Associate Editor