2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

Model-Free Unsupervised Anomaly Detection of a General Robotic System Using a Stacked LSTM and Its Application to a Fixed-Wing Unmanned Aerial Vehicle

Jae-Hyeon Park, Soham Shanbhag, and Dong Eui Chang*

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9981950

***Abstract*— With the growing application of various robots in real life, the need for an automatic anomaly detection system for robots is necessary for safety. In this paper, we develop an anomaly detection method using a stacked LSTM that can be applied to any robot controlled by a feedback control. Our method does not need installation of additional sensors. Our method is model-free and unsupervised because it does not require the analytical model of the system and the training data does not require faulty operation conditions. We validate our method on real fixed-wing unmanned aerial vehicle flight data containing control surface failure scenarios. We demonstrate the superiority of the proposed algorithm over existing anomaly detection methods in the literature. Our code is available at https://github.com/superhumangod/Model-free-unsupervised-anomaly-detection.**

I. INTRODUCTION

With the ever growing use of various robots like unmanned aerial vehicle (UAV) [1], unmanned ground vehicle (UGV) [2], and robotic arm [3], ensuring a safe operation is increasing in importance for maintaining robots in good condition and avoiding possible risks. However, various unintended anomalies can happen during robot operations like control surface failures, sensor failures, collisions, etc. Especially, when robots are operating in a remote area or in case of a swarm of robots, it would be better if the ground control station automatically informs the user when robots failed. There are two major types of autonomous anomaly detection methods: model-based and data-driven [4].

Model-based methods use model-based estimation algorithms like Kalman filters to generate residuals and use these residuals to detect anomalies. Bu et al. [5] used a particle filter to generate residuals, and applied fuzzy rules on the residuals to detect anomalies. Sun et al. [6] improved upon [5] for faster and robust anomaly detection, by using a Kalman filter and a neural network. Aboutalebi et al. [7] used a nonlinear observer to generate residuals and applied a neural network on the residuals to obtain an added fault signal. Venkataraman et al. [8] compared one parity-space based method and two types of observer-based methods for detecting control surface failures of the UAV. One downside of model-based methods is that they require the knowledge of an exact analytical model of the system and precise model parameters. However, finding an accurate model can be cumbersome, the problem which can be solved by data-driven methods.

Data-driven methods assume that data contain an underlying model, and use data-based methods like machine learning to detect deviations from the normal condition. These methods do not require the model and model parameters, since the machine learning algorithms learn those from the data. Some data-driven methods assume a linear relationship between variables. He et al. [9] used online subspace tracking to project the robot operation multidimensional data to a low-dimensional subspace, and detected anomaly if that subspace deviates severely from the normal condition. Birnbaum et al. [10] used a recursive parameter estimation to learn the model and detect anomalies, but they verified their algorithm only in simulation. Keipour et al. [11] shared some basic ideas with [10], and verified their algorithm in real UAV flight data. Park et al. [12] used multivariate linear analysis techniques like principal component analysis and partial least squares regression, to detect anomalies given robot operation data. However, robots are nonlinear systems which do not fit squarely into linear models.

Considering the nonlinearity of robot dynamics is one of the key ideas of accurate anomaly detection. Recent anomaly detection studies take into account the nonlinearity in robot dynamics. These studies use machine learning algorithms that learn the inherent nonlinearities in the system, and detect anomalies. Ouadine et al. [13] used a Hammerstein-Wiener model to learn the nonlinear model and detect anomalies. They tested their method using only simulation. Bronz et al. [14] used a nonlinear support vector machine to detect anomalies. Park et al. [15] trained an autoencoder neural network to reconstruct measurements during the normal operation. This algorithm detects anomaly if the reconstruction loss is greater than the threshold. This algorithm learns the nonlinearity by using a neural network, but does not take into account temporal correlations in the data.

Recently, a long short term memory (LSTM) neural network is being used in the field of anomaly detection, because it has the ability to learn long-term dependencies in a time series and solves the vanishing gradient problem of conventional recurrent neural networks [16]. LSTM is applied to anomaly detection of mechanical equipment [17],

* Corresponding author
The authors are with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), 291 Daehak-ro, Yuseong-gu, Daejeon 34141, South Korea. (alchemiclove@kaist.ac.kr; sshanbhag@kaist.ac.kr; dechang@kaist.ac.kr)

This work was conducted by Center for Applied Research in Artificial Intelligence(CARAI) grant funded by Defense Acquisition Program Administration(DAPA) and Agency for Defense Development(ADD) (UD190031RD).

big data [18], stock price [19], and feeding robot [20]. Wang et al. [21] made a UAV anomaly detection algorithm using an LSTM. This method learned the spatial-temporal relations between the variables using an LSTM, but predicted a single variable given the past variables as inputs, and used the prediction of this single variable to detect anomalies. Wang et al.[22] improved upon [21] for faster computation. The model [22] has the similar stacked LSTM structure with [21], but is computationally faster than [21] by reducing dimension of the input data with PCA and by using the special hardware for accelerated computation. Park and Chang [23] used multilayer perceptron and LSTM to detect faults in UAVs, but the algorithm could only detect trained faults. Although previous data-driven anomaly detection researches give promising results, there are no general rules for choosing what robot variables to use as inputs and outputs in the algorithm. We may insert additional sensors and use all possible measurements as inputs and outputs like autoencoder based anomaly detection [15], but there is a possibility that fewer variables are enough for promising performance. It is necessary to develop a general rule for selecting the variables to use as inputs and outputs for anomaly detection.

In our paper, we devise a model-free unsupervised anomaly detection method for a general robotic system based on a stacked LSTM. Our method is model-free because it does not require the analytical model of the system. Our method is unsupervised because it uses only normal operation data to train the stacked LSTM, thus making the data collection process much simpler as compared to methods requiring faulty data for training. Our method gives the general rule for choosing which variables to use as inputs and outputs for anomaly detection of any robot with a feedback control. Our method only uses the measured states and the reference states as inputs and outputs of the stacked LSTM. These variables are generally used for controlling the robot, so our algorithm does not need additional sensors and can be applied to any robot with a feedback control. We compare our method with the cutting-edge data-driven methods that use LSTM [21] or linear model approximation [11]. Our method gives the highest accuracy among the compared algorithms.

Our paper is organized as follows. Section II develops a methodology of anomaly detection for general robotic systems. Section III illustrates how we apply our method to a fixed-wing UAV anomaly detection. Section IV shows the performance of our method by comparing with other methods. Finally, Section V concludes the paper.

II. METHODOLOGY

In this section, we develop a general methodology of anomaly detection for robotic systems based on a stacked LSTM. Our method is classified as an LSTM based approach, which detects anomalies by utilizing the deviation of measured system outputs from predicted outputs of an LSTM, according to the survey by Lindemann et al [16]. A continuous-time dynamics of a general robotic system can be written in an abstract form as below:

$$\dot{S} = f_c(S, U), \tag{1}$$

where c stands for continuous-time, S is the state, and U is the control signal to the actuators. We assume that the sensor measurements and the control signals have the same frequency, with a time step of h. In most robotic systems, the state S can be measured by various sensors like GPS, IMU, magnetometer, or visual odometry [24] in discrete time as follows:

$$S_{m,k} = S_k + N, \tag{2}$$

where m stands for measurement, k is the epoch, and N is the stochastic noise. Many current robotic systems adopt reference point tracking by feedback control [25], [26], [27]. For these systems, control signal U is given as a function of the past measured states and the reference states as follows:

$$U_k = g(S_{m,k-1}, \ldots, S_{m,1}, S_{r,k-1}, \ldots, S_{r,1}), \tag{3}$$

where r stands for reference point. It is important to note that not all the variables in S are used in the reference point. From equations (1), (2) and (3), the measured states $S_{m,k}$ at a time kh can be approximately written as:

$$\begin{aligned} S_{m,k} &= f_d(S_{k-1}, \ldots, S_1, U_k) + N, \\ U_k &= g(S_{m,k-1}, \ldots, S_{m,1}, S_{r,k-1}, \ldots, S_{r,1}), \end{aligned}$$

where f_d is the discrete-time update equation corresponding to f_c. If we assume that the noise N has a small constant covariance, we may approximate the states S_i $(i = 1, \ldots, k-1)$ with the measured states $S_{m,i}$ $(i = 1, \ldots, k-1)$. Then, $S_{m,k}$ becomes a function of $(S_{m,k-1}, \ldots, S_{m,1}, S_{r,k-1}, \ldots, S_{r,1})$ as below:

$$S_{m,k} = f_{d2}(S_{m,k-1}, \ldots, S_{m,1}, S_{r,k-1}, \ldots, S_{r,1}) + N.$$

During any robot operation, we get a time series of measured states $S_{m,k}$ and reference states $S_{r,k}$ since $k = 0$. We assume that only previous L time steps affect the current state, and the states before L time steps have negligible effect on the current state. Then $S_{m,k}$ becomes a function of $(S_{m,k-1}, \ldots, S_{m,k-L}, S_{r,k-1}, \ldots, S_{r,k-L})$ as below:

$$S_{m,k} = f_{d3}(S_{m,k-1}, \ldots, S_{m,k-L}, S_{r,k-1}, \ldots, S_{r,k-L}) + N.$$

We use a stacked LSTM to learn f_{d3} using the data from a normal operation; refer to Malhotra et al [28] for more info on a stacked LSTM. We may devise the stacked LSTM structure with various complexity depending on the robot's system. After the stacked LSTM learns the function, it predicts the measured state $\hat{S}_{m,k}$ given the past measured states and the reference states. We calculate the residuals $r_k = \|\hat{S}_{m,k} - S_{m,k}\|$ at each time step, where $\|\cdot\|$ is the Euclidean norm for vectors. The residuals r_k will be small in a normal operation, and large in an operation with some anomalies in the sensors or actuators that affect the robotic system severely. Hence, we can detect anomalies with an appropriate threshold on the residuals r_k. Our method uses only the measured states and the reference states as inputs

and outputs of the stacked LSTM, so our method can be applied to any robot that uses a feedback control. Our method does not need any other measurements like an audio signal [29], a battery temperature [15], an actuator input signal [12] and does not predict all the variables like an autoencoder [15]. Thanks to the stacked LSTM, our method can learn the nonlinear functions of the robotic systems. Our method is model-free because it does not need the analytical system model. Although we learn the system model from the data, our algorithm is still classified as model-free because we do not use any information from the analytical system model on detecting faults. This nomenclature of 'model-free fault detection' is also adopted in [30], [31].

For training the stacked LSTM and finding the threshold for the residuals, we divide the robot operation data into a training set, a validation set, and a test set. The training set and the validation set are only composed of normal operation data. The validation set is used for choosing hyperparameters that minimize mean square loss between the predicted states and the measured states, and choosing the threshold value for anomaly detection. Our method is an unsupervised method because we do not need normal/fault labels in the training set and the validation set. This means we need that all data be absence of fault during training. This nomenclature of 'unsupervised fault detection' is also adopted in the following paper [15]. We use normal/fault labels only on the test set to assess the performance of our method. Our method trains the stacked LSTM using the training set to predict $S_{m,k}$ given $(S_{m,i}, S_{r,i})$ $(i = k-1, \ldots, k-L)$. For optimization, we use the mean squared error loss function and Adam optimizer. We train the stacked LSTM and select the model having the lowest loss in the validation set. For choosing the best input series length L, we start L from 3 and increase by 1. For each L, we initialize a new stacked LSTM, train it for many epochs, and save the model having the lowest validation loss. Among the stacked LSTM with different L, we choose the model with the lowest lowest validation loss.

After the training, we predict $S_{m,k}$ using the trained stacked LSTM, and calculate the residuals $r_k = \|\hat{S}_{m,k} - S_{m,k}\|$. We calculate the detection threshold of the residuals using the validation set. Due to fluctuation of residuals caused by noises in the measurements, we use a low-pass infinite impulse response (IIR) filter from [21] to smooth the residuals, which is defined as follows:

$$\begin{cases} r_{smooth,0} = r_0, \\ r_{smooth,k} = \beta r_{k-1} + (1-\beta) r_k, \ k \geq 2, \end{cases}$$

where β is the filter coefficient varying from 0 to 1. After smoothing, we calculate the mean $\mu_{val} \in \mathbb{R}$ and the standard deviation $\sigma_{val} \in \mathbb{R}$ of the smoothed residuals of the validation set. Then, we calculate the threshold $T = \mu_{val} + 3.291\sigma_{val}$ where 3.291 is chosen for 99.9% confidence interval. The choice of T is such that it is greater than all the residuals of the validation set, but not too large so that it is unable to detect faults in application. If T is too large, there will be many false negatives, and if T is too small, there will be many false positives, where positive implies an anomaly. We use the residuals of the validation set as the standard for choosing a magnitude of the proper threshold T. The filter coefficient β plays the important role in calculating T. If β is small, residuals become greater than T due to noise, and vice versa. By increasing β, we ensure that the residuals from the validation set become lower than T while the maximum residual is not much lower than T. We choose β such that $|T - \max r_{smooth}|$ attains it's minimum value. The pseudocode for training can be seen in Algorithm 1. We apply our algorithm in the test set using the value of β and T obtained from the validation set. Although we adopt the IIR idea from Wang et al. [21], we have given thorough explanation on choosing the filter coefficient β and the threshold T which [21] did not.

Algorithm 1 Training part of the fault detection algorithm

Require: Normal operation data of a robot

1: Split the data into a training set and a validation set
2: Calculate the mean μ and the standard deviation σ from the training set
3: Normalize all data by the z-score method with μ and σ
4: **for** $L \leftarrow L_{min}$ to L_{max} **do** ▷ In our case: $L_{min} = 3$, $L_{max} = 24$
5: Initialize the stacked LSTM that learns f_{d3}
6: Train the stacked LSTM using the training set for many epochs
7: Save the model with the lowest validation loss
8: **end for**
9: Choose the model with L that have the lowest lowest validation loss
10: Use the model to calculate residuals $r_k = \|\hat{S}_{m,k} - S_{m,k}\|$ of the validation set
11: **for** $\beta \leftarrow \beta_{min}$ to β_{max} **do** ▷ In our case: $\beta_{min} = 0.7$, $\beta_{max} = 0.97$
12: Use the low-pass IIR filter to smooth the residuals
13: Calculate μ_{val} and σ_{val} of the smoothed residuals
14: $T \leftarrow \mu_{val} + 3.291\sigma_{val}$
15: Save T and $|T - \max r_{smooth}|$ for each β
16: **end for**
17: Choose T and β with the lowest $|T - \max r_{smooth}|$ value

III. EXPERIMENT

In this section, we illustrate how we apply our method in Section II to an anomaly detection of a fixed-wing UAV. We implement our method using Python, Keras, and Tensorflow. We use the public dataset called AirLab Failure and Anomaly (ALFA) dataset made by Keipour et al. [32]. ALFA dataset presents a fixed-wing UAV flight data with several types of faults in control surfaces for use in anomaly detection researches. ALFA dataset includes processed data for 47 autonomous flights with 10 normal flight scenarios and 37 fault scenarios composed of engine, rudder, elevator, and aileron failures. Total time for the normal flight scenarios

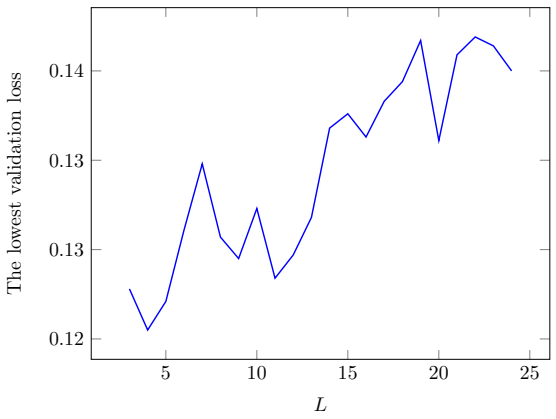

Fig. 1. The lowest validation loss of the stacked LSTMs with different L

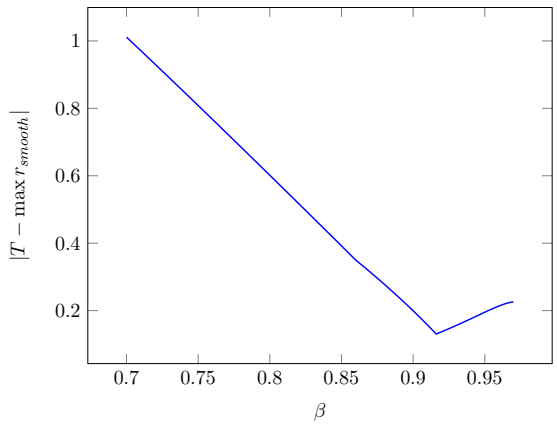

Fig. 2. A graph of $|T - \max r_{smooth}|$ with different β for $L = 4$

is 558 [s]. Fault scenarios consist of a normal flight followed by a sudden failure of a control surface. Fault scenarios have 3380 [s] of normal flight time and 777 [s] of post-fault flight time. Each flight scenario gives time series of various measurements at different frequencies. Among the 47 flight scenarios, the first scenario has no ground truth normal/fault labels, so we only use 46 flight scenarios.

As explained in Section II, we only need sequences of the measured states S_m and the reference states S_r. A state of a fixed-wing UAV can be written as $S = (P, V, \Theta, \Omega)$, where P is the position, V is the velocity, Θ is the attitude, and Ω is the angular velocity; refer to [33] for a fixed-wing UAV dynamics. Regarding the effect of position P on the dynamics, the dataset is collected at the same altitude and external conditions, and hence the effects due to P are negligible. Moreover, the dataset has reference states for velocity V and attitude Θ only. Hence we do not use P as inputs to the model. We show in Section IV that the model still performs well without P. The input variables for our stacked LSTM are the measured velocity $V_m \in \mathbb{R}^3$ in $[\mathrm{m\,s^{-1}}]$, the measured attitude Euler angles $\Theta_m \in \mathbb{R}^3$ in $[^\circ]$, the measured angular velocity $\Omega_m \in \mathbb{R}^3$ in $[^\circ\,\mathrm{s^{-1}}]$, the reference velocity $V_r \in \mathbb{R}^3$ in $[\mathrm{m\,s^{-1}}]$, and the reference attitude Euler angles $\Theta_r \in \mathbb{R}^3$ in $[^\circ]$. We obtain sequences of V_m, Θ_m, V_r, Θ_r from the mavros/nav_info topic, and Ω_m from the mavros/local_position topic in the given dataset; refer to [34] for more info on topics. We transform Θ_m and Θ_r into a unit quaternion, so they are capped between $[-1, 1]$ for the network stability. The measurements of the variables are available in varying frequencies: V_m, Θ_m, V_r, Θ_r in $20 - 25$ [Hz], and Ω_m in 4 [Hz]. So we sample those values at a frequency of 4 [Hz], which is a time step of 0.25 [s]. Also, we use the last values measured or commanded before each epoch, because the signals are not measured exactly at 0.25 [s] time steps. Finally, we obtain 46 flight scenarios which are time series of $X_k = (V_{m,k}, \Theta_{m,k}, \Omega_{m,k}, V_{r,k}, \Theta_{r,k})$ at a frequency of 4 [Hz].

To train the stacked LSTM, we need normal flight data for the training set and the validation set. To prepare training data, we include the 10 normal flight scenarios in the dataset, along with the pre-fault parts in 23 random fault flight scenarios. Remaining 13 fault flight scenarios are for the test set. Accordingly, we have 25 normal flight scenarios for the training set, 8 normal flight scenarios for the validation set, and 13 fault flight scenarios for the test set.

We use 25 normal flight scenarios in the training set for the flight data normalization. We calculate the mean $\mu \in \mathbb{R}^{17}$ and the standard deviation $\sigma \in \mathbb{R}^{17}$ from the training set. Then, we normalize all flight data including the validation set and the test set using the z-score method as $X_z = (X - \mu)/\sigma$. We do the normalization so that all the input variables to the stacked LSTM have a similar scale.

Now, we explain our stacked LSTM structure and optimization specifications. Our method trains the stacked LSTM in normal flight to predict $(V_{m,k}, \Theta_{m,k}, \Omega_{m,k}) \in \mathbb{R}^{10}$ given $(V_{m,i}, \Theta_{m,i}, \Omega_{m,i}, V_{r,i}, \Theta_{r,i}) \in \mathbb{R}^{17}$ $(i = k-1, \ldots, k-L)$. Our stacked LSTM structure is shown in Table I. Return sequences column denotes whether the output is a sequence or not. If it is true, the layer outputs the full sequence. It it is false, the layer outputs the last value of the output sequence. We use the training set to train our stacked LSTM. We use the mean squared error loss function, Adam optimizer, and the learning rate $\epsilon(n) = \epsilon_0 \times 0.5^{\lfloor n/22 \rfloor}$, where n is

TABLE I
THE STRUCTURE OF OUR STACKED LSTM

Layer	Type	Output Dim	Output Series Length	Return sequences
1	input	17	L	True
2, 3, 4	LSTM	64	L	True
5	LSTM	32	L	True
6	LSTM	32	1	False
7	Dense	32	1	-
8	ReLU	32	1	-
9	Dropout(0.1)	32	1	-
10	Dense	32	1	-
11	ReLU	32	1	-
12	Dropout(0.1)	32	1	-
13	Dense	10	1	-

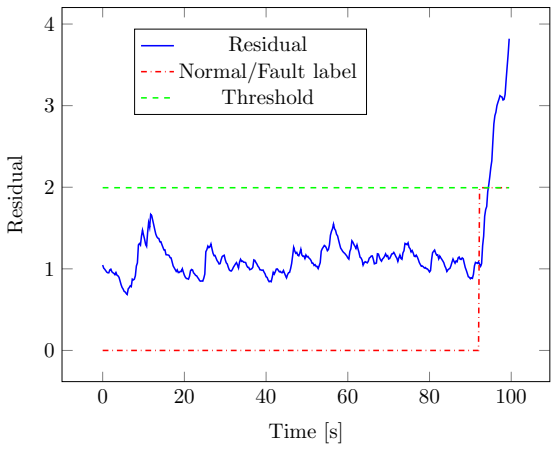

Fig. 3. Residual plot of a random fault scenario using our method. An example of a true negative and a true positive.

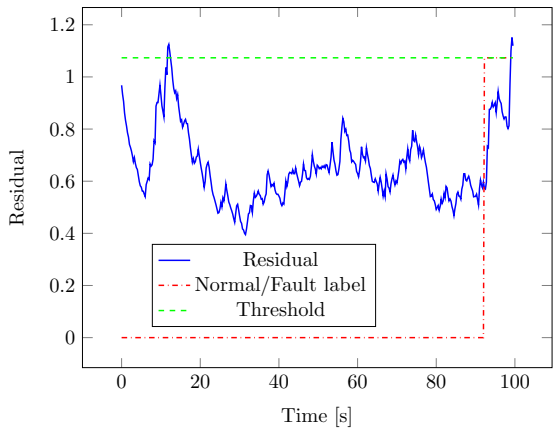

Fig. 4. Residual plot of the fault scenario showing a false positive using the network that predicts only roll rate. An example of a false positive and a true positive.

the epoch number, $\epsilon_0 = 0.01$ is the initial learning rate, and $[d]$ returns the element-wise largest integer no greater than d. The detailed explanation of neural network can be found in [35]. We choose L in the following way. We change L from 3 to 24 with a step size 1. For each L, we process the training set and the validation set as samples of series with length L, train the stacked LSTM in Table I for 100 epochs, and choose the model with the lowest validation loss. We plot the lowest validation loss of the stacked LSTMs with different L in Fig. 1. Then, we choose L with the lowest lowest validation loss. In this case, L is 4. We apply the trained stacked LSTM with $L = 4$ on the validation set and calculate residuals. To find β, we change β from 0.7 to 0.97 with a step size 0.001. For each β, we smooth the residuals using the low-pass IIR filter, calculate the threshold T, and calculate $|T - \max r_{smooth}|$. We plot $|T - \max r_{smooth}|$ with different β for $L = 4$ in Fig 2. We choose β with the minimum value of $|T - \max r_{smooth}|$, which is $\beta = 0.916$ and $T = 1.9941$.

IV. RESULTS

As in Section III, we use 13 fault flight scenarios as a test set to show the performance of our algorithm. These scenarios are time series data of a normal flight followed by a sudden control surface failure. Given the z-score normalized fault scenarios, we predict the states using the stacked LSTM trained in Section III, and calculate the residuals r_k. We smooth the residuals with the low-pass IIR filter using β calculated from the validation set and obtain the smoothed residuals $r_{smooth,k}$. Finally, anomaly detection is realized by the following: the UAV is in a normal state if $r_{smooth,k} < T$, and an anomalous state if $r_{smooth,k} \geq T$. We determine that there is an anomaly once $r_{smooth,k} \geq T$ happens.

For comparison, we choose the method of Wang et al. [21] and the method of Keipour et al. [11]. Wang et al. [21] used LSTM to make a UAV anomaly detection algorithm. This algorithm inputs past measurements to predict the next single variable, roll rate in the paper. Then, this algorithm makes residuals by subtracting predicted roll rates from the measured roll rates, and detects anomalies if the absolute value of the residual is greater than the threshold. Wang et al. did not provide a general rule for choosing which variables to use as inputs. For fair comparison, we use the same stacked LSTM structure, training set, and the validation set we use in Section III, but the network has only one output, which means this network predicts a single variable. After the prediction, we calculate residuals, do the low-pass IIR filter smoothing, and choose a threshold. We make individual networks that predict x-velocity, y-velocity, z-velocity, roll rate, pitch rate, yaw rate, separately. The values of β and T are optimized for each predicted variable, so they are different for each predicted variable. Keipour et al. [11] used 22 scenarios of ALFA dataset to validate their anomaly detection algorithm. Keipour et al.'s algorithm assumes a linear relationship between the measured signal and the reference signal of the single variable. This algorithm predicts the single variable given the past measured and reference values of that single variable using the recursive least squares method. Then, the algorithm subtracts the predicted values from the measured values to generate residuals, and detects anomalies if the absolute values of the residuals are greater than the threshold. Keipour et al's algorithm has some factors for engineering like the criterion for the algorithm to be considered stable, the number of the past time steps for the inputs to the algorithm, etc. We present two cases of Keipour et al's algorithm, a high true negative case and a high true positive case.

We show the residual plot of a random fault scenario using our method in Fig. 3. The blue solid line indicates the residuals, the red dash-dotted line indicates normal/fault labels in which 0 means a normal condition, and the green dashed line indicates the threshold. We define performance measures of the anomaly detection methods considering practical UAV flights, which are an average response time, a number of true positives, a number of false positives, a number of true negatives, and a number of false negatives.

TABLE II

PERFORMANCE MEASURES OF THE ANOMALY DETECTION METHODS. TN, FP, TP, FN REFER TO NUMBER OF TRUE NEGATIVES, FALSE POSITIVES, TRUE POSITIVES, AND FALSE NEGATIVES, RESPECTIVELY.

Method	TN	FP	TP	FN	Accuracy	Precision	Recall	F1 score	Average response time [s]
Our stacked LSTM	13	0	12	1	0.9615	1	0.9231	0.9600	8.38
x-velocity [21]	12	1	6	7	0.6923	0.8571	0.4615	0.6000	6.00
y-velocity [21]	13	0	1	12	0.5385	1	0.0769	0.1429	6.25
z-velocity [21]	11	2	4	9	0.5769	0.6667	0.3077	0.4211	5.81
roll rate [21]	11	2	3	10	0.5385	0.6000	0.2308	0.3333	12.33
pitch rate [21]	11	2	11	2	0.8462	0.8462	0.8462	0.8462	8.02
yaw rate [21]	11	2	3	10	0.5385	0.6000	0.2308	0.3333	13.58
linear model [11]	7	6	1	12	0.3077	0.1429	0.0769	0.1000	13.00
linear model [11]	4	9	5	8	0.3462	0.3571	0.3846	0.3704	10.55

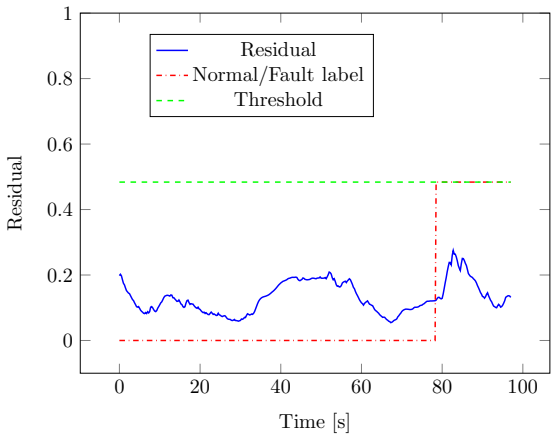

Fig. 5. Residual plot of the fault scenario showing a false negative using the network that predicts only y-velocity. An example of a true negative and a false negative.

In practical UAV flight, the user stops the UAV as soon as the user finds out an occurrence of a fault, so the user prevents severe situations like totally losing the UAV, and repairs it. Therefore, a response time of an anomaly detection algorithm, which is the time it takes for the algorithm to detect a fault since the start of the fault, is important. We define a response time as the time it takes from the start of the fault until the residuals exceed the threshold. Also, it is important for an anomaly detection algorithm to correctly detect faults. During the fault condition case of the fault scenarios, we increase the number of true positives by 1 if the residuals are greater than the threshold at least once, and increase the number of false negatives by 1 if the residuals are always below the threshold. Also, an anomaly detection algorithm should not give false positives in normal flight, because the user will keep stopping the UAV to check its condition which is bothersome. During the normal condition case of the fault scenarios, we increase the number of true negatives by 1 if the residuals are always lower than the threshold, and increase the number of false positives by 1 if the residuals are greater than the threshold at least once. Fig. 3 is an example of a true negative and a true positive. Fig. 4 is the residual plot of the fault scenario using the network that predicts only roll rate, and is an example of a false positive and a true positive. Fig. 5 is the residual plot of the fault scenario using the network that predicts only y-velocity, and is an example of a true negative and a false negative.

The performances of our stacked LSTM, Wang et al. [21], and Keipour et al. [11] are in Table II. Our method shows the best performance among the compared methods. Our method and Wang et al.'s method have the same stacked LSTM structure but different number of outputs. Our method predicts the whole state and combine the effects using the Euclidean norm, while Wang et al.'s method predicts one variable of the state at a time. Our method shows better performance than Wang et al.'s method because some variables might be more or less sensitive to the certain fault types giving a different detection performance to each fault type. Also, combining Wang et al.'s methods by using 'OR' or 'AND' operation at the end does not give better performance than ours. Using the Euclidean norm to combine the effect of the variables at the residual making step gives the best performance in our dataset. Keipour et al. [11] assumes a linear relation between the signals which is actually nonlinear. Our method extracts spatial-temporal features from the flight data, and learns a nonlinear relation between the variables thanks to the stacked LSTM. Therefore, our method gives the best accuracy. Although we validate our method in control surface failure scenarios, our method can be applied to detection of various faults that leads to a deviation of the system from the non-faulty system like a structural damage or constant external disturbances. We add UAV animations made by MATLAB code from [36] in our supplementary video.

In addition, we check the real-time performance of our algorithm. We use the following computation platform to do simulations: CPU - Intel(R) Core(TM) i7-8700 @ 3.20GHz, GPU - NVIDIA GeForce RTX 2080 Ti, and memory - 16GB RAM. We read the measurements in 4 [Hz], do preprocessing, and input the measurements to our stacked LSTM to detect anomalies. Total computation time for one step is just $6.21[\mu s]$ which is negligible compared to 250 [µs]. Hence, our algorithm allows for real time detection.

V. CONCLUSIONS

In this paper, we develop a model-free unsupervised anomaly detection method of a general robotic system using a stacked long short-term memory (LSTM) network. Our

method is model-free in the sense that it does not need an analytical model of a robotic system. This method is unsupervised because we only need normal operation data with no normal/fault labels during the training. We use the stacked LSTM to predict the current measured states using the past measured states and reference states as inputs, calculate residuals, do smooth filtering on the residuals, and detect anomalies with the threshold. For validation of the proposed method, we use AirLab Failure and Anomaly dataset, which is a public dataset containing 47 flight scenarios of a fixed-wing unmanned aerial vehicle (UAV) with normal and fault conditions. The proposed method shows the highest performance compared to the other cutting-edge anomaly detection methods. We have assumed availability of full state measurement. A future area of research would be adapting this method for partially observable states.

REFERENCES

[1] A. Tahir, J. Böling, M.-H. Haghbayan, H. T. Toivonen, and J. Plosila, "Swarms of unmanned aerial vehicles—a survey," *Journal of Industrial Information Integration*, vol. 16, p. 100106, 2019.

[2] K. Asadi, A. K. Suresh, A. Ender, S. Gotad, S. Maniyar, S. Anand, M. Noghabaei, K. Han, E. Lobaton, and T. Wu, "An integrated UGV-UAV system for construction site data collection," *Automation in Construction*, vol. 112, p. 103068, 2020.

[3] X. Xing and D. E. Chang, "Deep reinforcement learning based robot arm manipulation with efficient training data through simulation," in *2019 19th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2019, pp. 112–116.

[4] L. H. Chiang, E. L. Russell, and R. D. Braatz, *Fault detection and diagnosis in industrial systems*. London, United Kingdom: Springer Science & Business Media, 2000.

[5] J. Bu, R. Sun, H. Bai, R. Xu, F. Xie, Y. Zhang, and W. Y. Ochieng, "Integrated method for the UAV navigation sensor anomaly detection," *IET Radar, Sonar & Navigation*, vol. 11, no. 5, pp. 847–853, 2017.

[6] R. Sun, Q. Cheng, G. Wang, and W. Y. Ochieng, "A novel online data-driven algorithm for detecting UAV navigation sensor faults," *Sensors*, vol. 17, no. 10, p. 2243, 2017.

[7] P. Aboutalebi, A. Abbaspour, P. Forouzannezhad, and A. Sargolzaei, "A novel sensor fault detection in an unmanned quadrotor based on adaptive neural observer," *Journal of intelligent & robotic systems*, vol. 90, no. 3, pp. 473–484, 2018.

[8] R. Venkataraman, P. Bauer, P. Seiler, and B. Vanek, "Comparison of fault detection and isolation methods for a small unmanned aircraft," *Control Engineering Practice*, vol. 84, pp. 365–376, 2019.

[9] Y. He, Y. Peng, S. Wang, and D. Liu, "ADMOST: UAV flight data anomaly detection and mitigation via online subspace tracking," *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 4, pp. 1035–1044, 2018.

[10] Z. Birnbaum, A. Dolgikh, V. Skormin, E. O'Brien, D. Muller, and C. Stracquodaine, "Unmanned aerial vehicle security using recursive parameter estimation," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1, pp. 107–120, 2016.

[11] A. Keipour, M. Mousaei, and S. Scherer, "Automatic real-time anomaly detection for autonomous aerial vehicles," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5679–5685.

[12] J.-H. Park, C.-Y. Jun, J.-Y. Jeong, and D. E. Chang, "Real-time quadrotor actuator fault detection and isolation using multivariate statistical analysis techniques with sensor measurements," in *2020 20th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2020, pp. 33–37.

[13] A. Y. Ouadine, M. Mjahed, H. Ayad, and A. El Kari, "UAV quadrotor fault detection and isolation using artificial neural network and Hammerstein-Wiener model," *Stud Inform Control*, vol. 29, no. 3, pp. 317–328, 2020.

[14] M. Bronz, E. Baskaya, D. Delahaye, and S. Puechmore, "Real-time fault detection on small fixed-wing UAVs using machine learning," in *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. IEEE, 2020, pp. 1–10.

[15] K. H. Park, E. Park, and H. K. Kim, "Unsupervised fault detection on unmanned aerial vehicles: encoding and thresholding approach," *Sensors*, vol. 21, no. 6, p. 2208, 2021.

[16] B. Lindemann, B. Maschler, N. Sahlab, and M. Weyrich, "A survey on anomaly detection for technical systems using LSTM networks," *Computers in Industry*, vol. 131, p. 103498, 2021.

[17] Z. Li, J. Li, Y. Wang, and K. Wang, "A deep learning approach for anomaly detection based on SAE and LSTM in mechanical equipment." *International Journal of Advanced Manufacturing Technology*, vol. 103, 2019.

[18] X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, "Variational LSTM enhanced anomaly detection for industrial big data," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3469–3477, 2020.

[19] T. Ergen and S. S. Kozat, "Unsupervised anomaly detection with LSTM neural networks," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 8, pp. 3127–3141, 2019.

[20] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1544–1551, 2018.

[21] B. Wang, D. Liu, Y. Peng, and X. Peng, "Multivariate regression-based fault detection and recovery of UAV flight data," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 6, pp. 3527–3537, 2019.

[22] B. Wang, X. Peng, M. Jiang, and D. Liu, "Real-time fault detection for UAV based on model acceleration engine," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 12, pp. 9505–9516, 2020.

[23] J.-H. Park and D. E. Chang, "Data-driven fault detection and isolation of system with only state measurements and control inputs using neural networks," in *2021 21st International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2021, pp. 108–112.

[24] V. Usenko, J. Engel, J. Stückler, and D. Cremers, "Direct visual-inertial odometry with stereo cameras," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1885–1892.

[25] I. Matraji, A. Al-Durra, A. Haryono, K. Al-Wahedi, and M. Abou-Khousa, "Trajectory tracking control of skid-steered mobile robot based on adaptive second order sliding mode control," *Control Engineering Practice*, vol. 72, pp. 167–176, 2018.

[26] A. Andreev and O. Peregudova, "Trajectory tracking control for robot manipulators using only position measurements," *International Journal of Control*, vol. 92, no. 7, pp. 1490–1496, 2019.

[27] D. E. Chang and Y. Eun, "Global chartwise feedback linearization of the quadcopter with a thrust positivity preserving dynamic extension," *IEEE Transactions on Automatic Control*, vol. 62, no. 9, pp. 4747–4752, 2017.

[28] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, vol. 89, 2015, pp. 89–94.

[29] A. Altinors, F. Yol, and O. Yaman, "A sound based method for fault detection with statistical feature extraction in UAV motors," *Applied Acoustics*, vol. 183, p. 108325, 2021.

[30] C. Alippi, S. Ntalampiras, and M. Roveri, "Model-free fault detection and isolation in large-scale cyber-physical systems," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 1, pp. 61–71, 2016.

[31] P. Zhang and S. X. Ding, "A model-free approach to fault detection of continuous-time systems based on time domain data," *International Journal of Automation and Computing*, vol. 4, no. 2, pp. 189–194, 2007.

[32] A. Keipour, M. Mousaei, and S. Scherer, "ALFA: A dataset for UAV fault and anomaly detection," *The International Journal of Robotics Research*, vol. 40, no. 2-3, pp. 515–520, 2021.

[33] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*. New Jersey, USA: Princeton university press, 2012.

[34] Open Robotics. (2021) ROS. [Online]. Available: https://www.ros.org/

[35] A. L. Caterini and D. E. Chang, *Deep Neural Networks in a Mathematical Framework*. Cham, Switzerland: Springer, 2018.

[36] W. Bużantowicz, "Matlab script for 3D visualization of missile and air target trajectories," *International Journal of Computer and Information Technology*, vol. 5, no. 5, pp. 419–422, 2016.