Deep Learning Based Framework for Multicopter Sensor Data Imputation

Gaurav Makkar
PhD Student

Farhan Gandhi
Redfern Professor and Director

Center for Mobility with Vertical Lift (MOVE)
Rensselaer Polytechnic Institute, Troy, NY

ABSTRACT

The study presented in this paper introduces a novel approach for imputing missing sensor data in multicopters, which enables enhanced safety and reliability by leveraging the multitude of sensors on these aerial vehicles. The proposed approach is based on two deep learning techniques, namely Autoencoders (AE) and Long Short-Term Memory (LSTM) networks. The effectiveness of this approach is evaluated using flight test data from a 2.5 kg hexacopter, and three different scenarios of missing data are considered. To validate the performance of the proposed approach, it is compared against two commonly used imputation techniques: k-Nearest Neighbor (KNN) imputation and Random Forest imputation. The results indicate that the proposed approach outperforms both KNN and Random Forest in terms of the accuracy of imputation. The network has an error of less than 10% when processing signals with six missing sensor readings for a duration of 10 seconds. In contrast, KNN and Random Forest algorithms have an average error of 18% and 26%, respectively. Moreover, the trained model can handle missing data with varying degrees of sparsity, which makes it a more robust and flexible solution. The study also investigates the impact of using initial estimates provided by the Kalman Filter for training the deep learning models. It is observed that incorporating these estimates does not result in any improvement in the imputation accuracy. This suggests that the proposed approach is able to learn the underlying patterns in the data without the need for additional information from the Kalman Filter. Overall, the results of this study demonstrate the potential of deep learning techniques for imputing missing sensor data in multicopters. The proposed approach offers a more accurate and efficient solution than the traditional imputation techniques, and can handle varying degrees of data sparsity. The findings of this study have important implications for the design and operation of multicopters, and could result in enhanced performance and operational effectiveness of these aerial vehicles.

NOTATION

h	altitude
ϕ	roll attitude
θ	pitch attitude
ψ	yaw attitude
a_x	acceleration along x axis
a_y	acceleration along y axis
a_z	acceleration along z axis
p	roll rate
q	pitch rate
r	yaw rate
V_0	collective control
V_{1c}	lateral control
V_{1s}	longitudinal control
V_{0d}	yaw control

INTRODUCTION

Unmanned Aircraft System (UAS) operations have grown significantly over the last decade, largely driven by the emergence of new commercial and military applications, and this in turn has led to increased interest in developing trusted autonomy. Autonomous UAS are intended to operate in close proximity to obstacles and potentially human bystanders, necessitating strict performance requirements on guidance systems in terms of accuracy and integrity. High-quality data from sensors plays a pivotal role in meeting these requirements and allowing for decisions to be made with high confidence in these vehicles when flown autonomously. Issues such as power supply interruption, temperature-dependent variation, sensor damage, transmission failure, etc. can cause sensor data loss or packet loss, resulting in missing values that will prevent the system from receiving critical information (Refs. 1–5). This can negatively affect the behavior of the aircraft mid-flight and potentially risk the safety of the vehicle or bystanders.

Even though conventional data-fusion algorithms like Extended Kalman Filter (EKF) have been enhanced so as to improve accuracy and robustness, their performance can degrade with increasing non-linearity in system dynamics (Ref. 6). The field of machine learning and artificial intelligence (AI) has seen major advancements in the last decade. This along with low form factor processor boards, capable of parallel computations and development of advanced optimization al-

Presented at the Vertical Flight Society's 79th Annual Forum & Technology Display, West Palm Beach, FL, USA, May 16–18, 2023.

gorithms, have opened up avenues to employ machine learning techniques to enhance autonomous capabilities, including imputing missing data (Refs. 5, 7–9).

Some research groups have looked at the problem of missing sensor values in a statistical sense (Refs. 10–18). These statistical methods are simple to use, and their performance is acceptable when the rate of missing data is low. However, with increasingly complex datasets, the performance of statistic-based methods declines, since these methods are based on linear assumptions which is not true for nonlinear, real-world situations. Addressing this issue, multiple groups have applied machine learning to fill in or forecast the missing values. Amiri et al. (Ref. 19) compared different fuzzy-rough nearest neighbors-based methods for missing data imputation. Garnier et al. (Ref. 20) implemented Artificial Neural Network (ANN) for missing values estimation. Another research group compared different decision tree-based methods for predictions with incomplete data on different datasets (Ref. 21).

Despite these efforts, there are still some problems that the algorithms mentioned above fail to address. First, they perform poorly if multiple time series have data missing at the same time. Second, most of the studies examined situations where the missing rate is small, which is not always true. Addressing these issues, the goal of this study is to develop a deep learning-based framework for multiple time-series missing value imputation in the case of a multicopter sensor malfunction/failure. This methodology uses temporal information and also explores correlations between different variables by combining two deep learning methods: Long Short-Term Memory (LSTM) networks, and Autoencoders (AE), to comprehensively handle different situations of missing data.

MODELING AND ANALYSIS

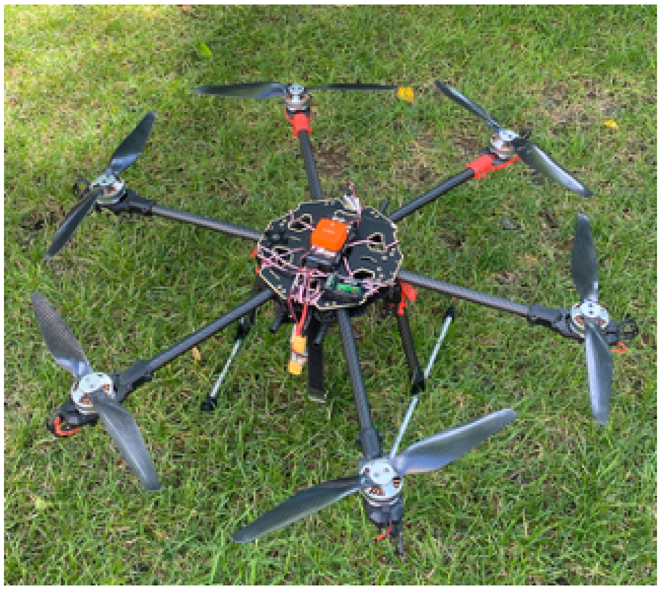
Figure 1. RPI's Hexacopter

The platform used for the study is RPI's hexacopter (Fig. 1). It is based on Tarot 680 Pro frame and features the Cube Orange flight controller (Ref. 22). Detailed specifications of the aircraft are provided in Table 1. The aircraft is flown in stabilize mode for data collection and it allows the pilot to fly the vehicle manually.

Cube orange features a barometer, magnetometer, and an inertial measurement unit (IMU), that employs an accelerometer, and a gyroscope. All the sensors are connected via a serial peripheral interface for quick transmission of data. Pixhawk also logs the commanded controls along with all the sensor outputs, which can be retrieved after the flight for analysis. This study uses four controls (PWM signals: V_0, V_{1c}, V_{1s}, and V_{0d}), accelerations (a_x, a_y, and a_z), angular rates(p, q, and r), attitudes (ϕ, θ, ψ) and barometer recordings (h) for the algorithm development.

The linear accelerations and angular rates are recorded by the accelerometer and gyroscope respectively. The attitudes are then calculated based on the accelerometer, and gyroscope readings using an Attitude Heading Reference System (AHRS). Thus, if either accelerometer or gyroscope readings are affected due to sensor malfunctioning, the attitude calculation will be erroneous. With that in consideration, this study considers the cases listed in Table 2. Thus, the algorithm should be capable of handling cases with 1, 3, or 6 time series with missing data.

A good imputation model should be capable of exploring correlations among data and learning temporal information to handle different missing patterns (systematic missing, random missing, etc.). This study uses Autoencoders (AE) for exploring correlations and Long Short Term Memory (LSTM) networks to handle the temporal dependencies in data. AE are multilayered NN consisting of two blocks: encoder and decoder. The feature extraction capability of AE can be used to learn the correlations between different signals. LSTM is a special kind of recurrent neural network capable of learning both long and short-term dependencies. These two machine learning approaches are discussed in detail in the following sections. For the present study, the Matlab® Deep Learning toolbox is used for implementing both deep learning algorithms.

Table 1. Hexacopter Specifications

Aircraft	
Weight, with battery	2.5kg
Boom Length	340mm
Motor Weight	93gm
Motor Kv Rating	380RPM/V
Rotor diameter	330mm

Autoencoders

Autoencoder (AE) is a multi-layer neural network architecture that consists of encoding and decoding layers connected by a bottleneck layer (Refs. 23–25). The primary focus of

Table 2. Possible Missing Data Scenarios

	Variable affected	Number of time series with missing data
Barometer	h	1
AHRS	ϕ, θ, ψ	3
Accelerometer	a_x, a_y, a_z, and ϕ, θ, ψ	6
Gyroscope	p, q, r, and ϕ, θ, ψ	6

an AE is to learn a lower-dimensional representation, known as encoding, for higher-dimensional data. In the context of the current application, where the sensor readings are related by physics-based equations, AE can capture the features that correlate the different data streams. This is accomplished by training the network to minimize the difference between the input and the reconstructed output using a loss function such as mean squared error.

Autoencoders are a versatile class of neural networks that can be used for various tasks such as data compression, denoising, and anomaly detection. In data compression, the objective is to learn a compressed representation of the data that can be used to reconstruct the original data with minimal loss of information. In denoising, the objective is to learn a representation of the data that is robust to noise and can be used to remove noise from the input. Finally, in anomaly detection, the autoencoder learns a representation of the data that can identify instances that deviate from the normal patterns in the data.

Autoencoders are a powerful tool for unsupervised learning tasks such as data compression, denoising, and anomaly detection. They work by learning a compressed representation of the input data that can be used to reconstruct the original input with minimal loss of information. Variants such as the Variational Autoencoder (VAE) can also learn a probabilistic representation of the data, which is useful for tasks such as generative modeling and image synthesis.

The autoencoder architecture (Fig. 2) consists of two main parts: the encoder and the decoder. The encoder takes the input data and maps it to a lower-dimensional representation, called the encoding or latent representation. The decoder then takes this encoding and maps it back to the original input space. Both the encoder and decoder are typically implemented as neural networks, with the encoding and decoding functions learned during the training process.

The bottleneck layer is designed to restrict the flow of information, allowing only the essential information to pass through. The size of the bottleneck layer determines the features that are extracted from the information passed from the encoder layer. Since the features are embedded in the bottleneck layer, missing data can be imputed using the correlation information stored in the bottleneck. Additionally, since the bottleneck represents the compressed representation of the input, it lowers the risk of overfitting.

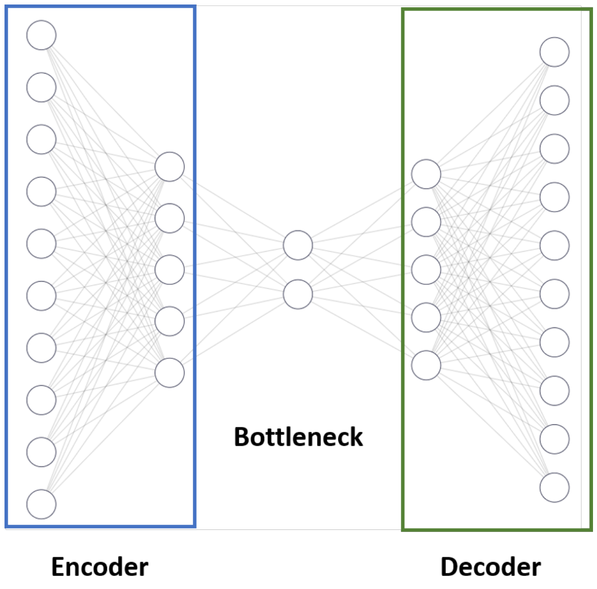

Figure 2. Autoencoder Architechture

Long short-term memory (LSTM)

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture that was first proposed by Hochreiter and Schmidhuber in 1997 (Ref. 26). The purpose of LSTMs is to overcome the vanishing gradient problem, which is a common issue encountered in traditional RNNs during training on long sequences. The vanishing gradient problem arises when the gradients that are propagated back through time in the network become very small, making it difficult for the model to learn long-term dependencies.

LSTMs have been extensively verified in the machine learning and AI community on various problems, such as speech recognition and fault detection (Refs. 27, 28). Their neuron structure is particularly well-suited for learning long and short-term impacts from past time-series, resulting in superior performance compared to traditional nonlinear machine learning algorithms like Support Vector Machines (SVM) or Artificial Neural Networks (ANN).

Fig. 3 compares a single ANN unit with LSTM unit. It can be observed from the figure that the main difference is that LSTM units have value (h_t) and state communication (C_t) in between the units. So, in addition to input at the current time step, LSTM also uses the output from the previous time step to make predictions.

LSTMs address the vanishing gradient problem by incorporating memory cells and gates into the RNN architecture. The memory cells are capable of storing information for extended periods, while the gates can control the flow of information into and out of the memory cells. There are three primary types of gates used in LSTMs: the input gate, the forget gate, and the output gate (Fig. 4). The input gate controls the flow of information into the memory cell, while the forget gate controls the flow of information out of the memory cell. Finally,

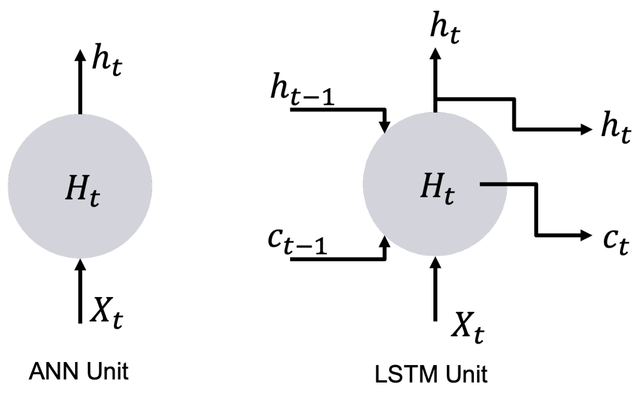

Figure 3. ANN unit vs LSTM unit (Ref. 27)

the output gate determines which information should be output from the memory cell.

First, the forget gate decides which information to keep from historical blocks, and is given by the Eq. 1.

$$f_t = \sigma(W_f.[h_{t-1}, X_t] + b_f) \tag{1}$$

where σ represents the sigmoid function, W_f and b_f are the weights and bias. For each training iteration, the weights and biases are tuned automatically during the process to improve the predictions.

The input gate decides which new information in X_t should be stored in the cell state. It is accomplished using Eq. 2 and 3.

$$i_t = \sigma(W_i.[h_{t-1}, X_t] + b_i) \tag{2}$$

$$\tilde{C}_t = tanh(W_c[h_{t-1}, X_t] + b_c) \tag{3}$$

The old state is multiplied by f_t to forget what is unnecessary. The new C_t is given as follows

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{4}$$

Eventually, based on C_t, h_{t-1}, and X_t, output gate provides h_t using Eq. 5, and 6.

$$o_t = \sigma(W_o.[h_{t-1}, X_t] + b_o) \tag{5}$$

$$h_t = o_t \times tanh(C_t) \tag{6}$$

The units are stacked in series as shown in Fig. 4. All the other calculations resemble the processes in a simple ANN such as back-propagation, adaptive learning rate, and batch gradient descent.

In summary, LSTMs are a powerful class of neural networks that can handle the vanishing gradient problem and model long-term dependencies in sequential data. They achieve this by incorporating memory cells and gates into the RNN architecture, allowing the model to selectively retain or discard information over long sequences.

Training Algorithm

The architecture of the proposed algorithm is shown in Figure 5. The encoding layers of the AE compress the sensor data by nonlinear transformation and the decoding layers use the compressed features to recover the data. $X_T = X_1, X_2, ...X_t$ represents the complete historical data with T time slots. It is a multidimensional time-series with $X_1 = [V_0\ V_{1c}\ V_{1s}\ V_{d_1}\ \phi\ \theta\ \psi\ p\ q\ r\ a_x\ a_y\ a_z\ h]_1$. The AE tries to learn a function $g_{w,b}(X)$ that satisfies $g_{w,b}(X) \approx X$, where w and b are the hyperparameters that control the learning process. The objective function for AE is given by Eq. 7.

$$L(g_{w,b}, X) = \frac{1}{2}||g_{w,b}(X) - X||^2 \tag{7}$$

where $g_{w,b}$ is the reconstructed input as given by the first simple autoencoder.

$$g_{w,b}(X) = W_1 f(W_1 x + b) + b' \tag{8}$$

The forward calculation of the $(l+1)$layers in AE is

$$\alpha^{(l+1)} = f(W^{(l)} a^{(l)} + b^{(l)}) \tag{9}$$

where f is the activation function and $W^{(l)}$, $\alpha^{(l)}$, and $b^{(l)}$ are the weights, activation value, and bias of the layer l respectively. Other details of the AE structure are discussed in the text (Ref. 23).

The weights and bias can be updated using

$$W_i^l = W_i^l - \alpha a_j^l \delta_i^{l+1} \tag{10}$$

$$b_i^l = b_i^l - \alpha \delta_i^{l+1} \tag{11}$$

To model the temporal relations, the regular neurons are replaced by LSTM cells (Figure 5). The readings at adjacent time stamps would not change abruptly, but over a longer range the differences might be pronounced. Also, since the data is collected from flight tests, it is inherently noisy, and therefore the model needs to be regularized. The regularization penalizes large deviations.

$$L'(g_{W,b}, X) = \frac{1}{2}||g_{W,b}(X_T) - X_T||^2 + \tag{12}$$

$$\lambda||g, X'|| \tag{13}$$

λ is the regularization parameter. X_T' is the prediction at the target time stamp which has weights shared from it's adjacent time stamps (Fig. 6). This can be thought of as approximating maximum likelihood training for the generative model (Ref. 23)

$$log p_{model} x = log \sum_h p_{model}(h, x) \tag{14}$$

To enable the network to capture long-term and short-term dependencies a sweep of varying window sizes is executed.

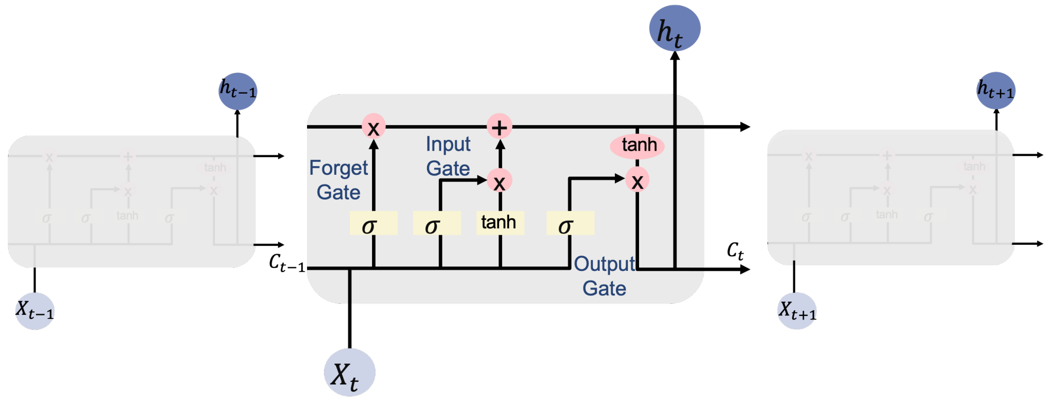

Figure 4. The repeating module in an LSTM contains four interacting layers (Ref. 29)

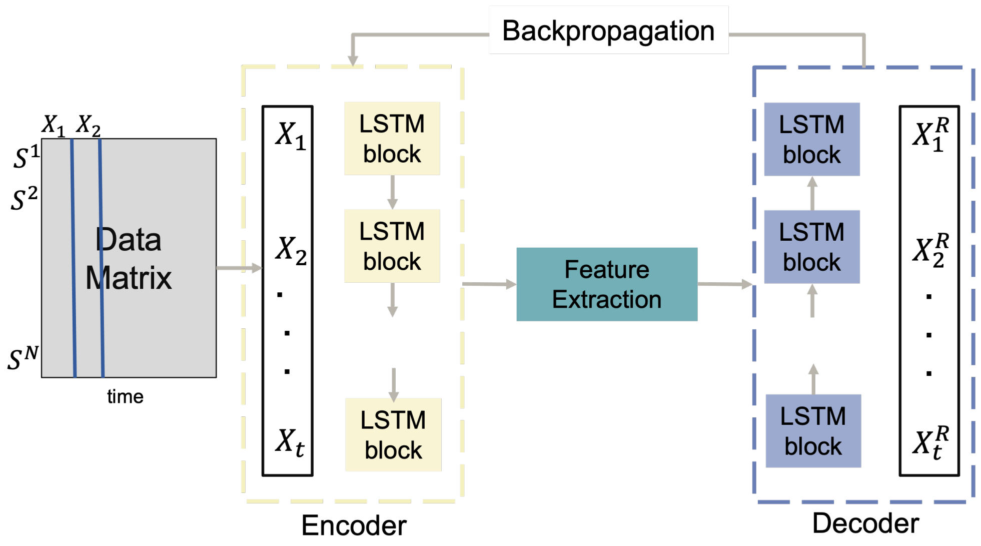

Figure 5. LSTM and AE based data imputation algorithm

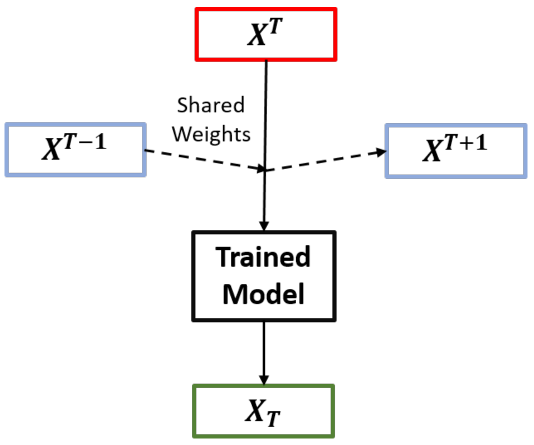

Figure 6. Overall model of training spatiotemporal auxiliary features with regularization.

For this study, a total of 3200 seconds of flight test data was used. 2400 seconds (75% of the total data set) of data was used for training and validation, and the remaining is used for testing the trained network. In addition to hover, the flight test data contains data corresponding to various maneuvers. Two examples are shown in Fig. 7, and 8. The orange arrows represent the direction of flight and grey circles represent the markers. It is important to note that the distances x_1, x_2, and height h are varied ($x_1, x_2 \approx 12m - 15m$). Moreover, the tests are done at various airspeeds. Thus, the training data consists of variations of multiple flight variables. This ensures that the network is trained on all variations and generalizes well. During the training process, the data is passed in blocks of 10 seconds. The flag for missing data for the intended flight variable is raised and the network tries to regenerate the data for that variable. This value is then compared with the true values to find the RMSE errors and train for the next iteration. The training process is initialized with arbitrary weights and these are tuned until the reconstruction error $|X_1 - X_{1R}|$ for all

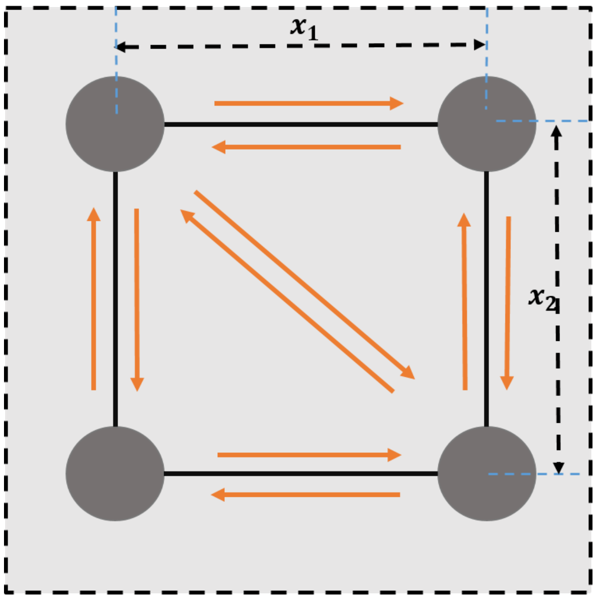

Figure 7. Example: Top view of the maneuver including in the data

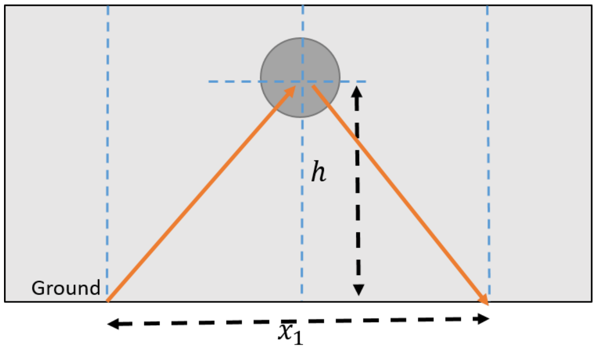

Figure 8. Example: Climb and descent data included in the data

the variables in the validation set stops decreasing (Ref. 23). The training process can be summarized as follows:

- Split the dataset into training and testing dataset.
- Initialize AE-LSTM
- For each X_i in the training set, output the corresponding reconstruction.
- Calculate reconstruction error: $|X_1 - X_{1R}|$
- Calculate regularization error: $\lambda||(g,X)||^2$
- Backpropagate the error to update the weights and biases: $W_i^l = W_i^l - \alpha a_j^l \delta_i^{l+1}$

RESULTS

Once the training is complete, the model can be tested on the reserved test data set. To predict the missing values in real-time, the test data set is configured to be passed in series and the missing values are flagged. As soon as the model sees the missing flag, it runs the dataset through the trained model and starts making the predictions. The length of missing time is increased from 1 second to 10 seconds in increments of 1 second to check the performance of the trained model with varying lengths of missing data.

Barometer Readings Missing - 1 time series

First, the case with missing barometer readings is considered. In Fig. 9, five seconds of barometer data is missing every 15 seconds (represented by the gray regions). The true barometer reading is shown in blue and the predicted values, using the trained network, are represented in orange. Visually comparing the two curves, it is clear that the trained LSTM-AE network is able to predict the aircraft altitude well when data is missing from one time series. To quantify the performance, the root mean square error (RMSE) is evaluated for the entire test dataset using a similar procedure. The evaluated RMSE is 0.12 m. The reason behind the good predictions is that the AE structure is able to capture the correlations between barometer readings and other variables, and LSTM learns the temporal relationship in the data. With data only missing from one-time series, there is sufficient information for imputation since the prediction is made using the information about the correlation with the other 13 variables (controls, accelerations, attitudes, and angular rates).

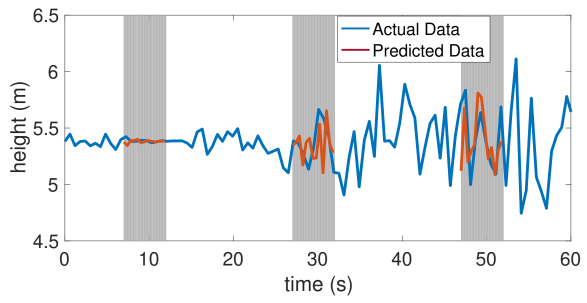

Figure 9. Predictions with barometer data missing

Next, the performance of the LSTM is compared with two established data imputation algorithms, namely K-Nearest Neighbors (KNN) and Random Forest Imputation (Fig. 10). KNN is designed to find K nearest neighbors from all complete instances in a given dataset and then fill in the missing values with the mean of the neighbors (Ref. 30). The Random Forest algorithm works by creating multiple decision trees and aggregating their predictions. Each decision tree is built on a random subset of the data and a random subset of the features. The algorithm then averages the predictions of all the decision trees to make a final prediction (Refs. 31, 32). The predicted values are then used to fill in the missing values in the dataset. RMSE is computed for all three algorithms for length of missing data ranging from 1 to 10 seconds as depicted in Fig. 10. As anticipated, the RMSE increases with the duration of missing data for all three algorithms (LSTM-AE in blue, KNN in orange, and Random Forest in yellow),

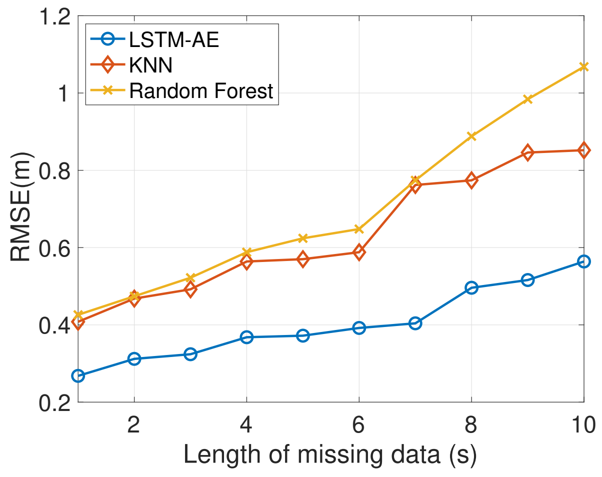

Figure 10. Performance comparison of different imputation methods

indicating that temporal information from previous time steps becomes weaker as the algorithm progresses. Furthermore, the RMSE values of LSTM-AE are (50% to 75%) lower than those of KNN and Random Forest, highlighting its superior predictive performance.

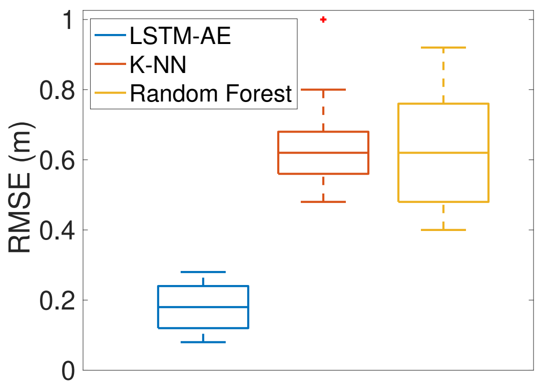

Figure 11. Box Plot for testing data using sliding window

The method's generalization capability can be evaluated by implementing a sliding window on the test dataset and computing the root mean square error (RMSE) for each method. This is done by using a window of fixed size and moving it along the time axis and making predictions for each situation. This method enables the visualization of the error variance across the entire dataset and enables a quantitative assessment of the developed methods' performance across the entire dataset, providing a robust evaluation of its generalization capability. The box plot of the testing dataset using a sliding window of 5 seconds is presented in Fig. 11. The error spread for LSTM-AE and KNN is smaller than the spread observed using the Random Forest approach. Notably, the mean value predicted by LSTM-AE is 0.4m lower than the mean error obtained by the other two methods.

AHRS Reading Missing - 3 time series

Given that the predictive ability of the network has been established in the scenario where data is missing from one time series, the next case to be considered is when the Attitude and Heading Reference System (AHRS) is malfunctioning. In this scenario, the attitude calculations are either incorrect or missing (only sensor noise). This presents a significant challenge as accurate attitude calculations are critical for the safe operation of many systems, particularly those involving navigation and control. This would mean that data is missing from 3-time series (ϕ, θ, and ψ) simultaneously. The predictions for attitudes are shown in Fig. 12. Again, the grey regions represent five seconds of time when the data is missing. The same trained model makes accurate predictions even when 3-time series have data missing simultaneously, and shows good agreement with the actual data over all three missing data segments.

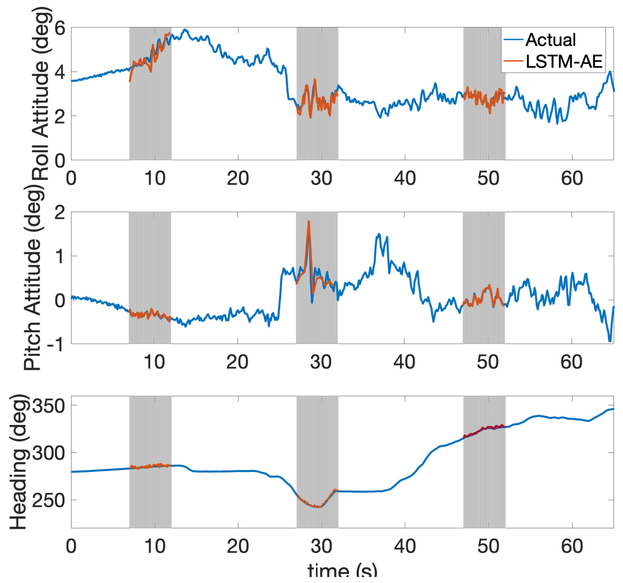

Figure 12. Predictions with AHRS data missing

The overall RMSE value for attitudes with the length of missing time increasing from 1-second to 10 seconds is shown in Figure 13. The RMSE increases for all three attitudes as the length of missing time increases since the temporal information for each attitude gets weaker. The LSTM-AE-based algorithm performs 75% - 100% better than KNN and Random Forest for all the lengths of missing data. This is due to the strong correlation between the accelerations/angular accelerations and attitudes.

To assess the generalization capabilities of the algorithm, a sliding window approach is adopted to assess its performance across multiple datasets. The resulting RMSE error distributions for each dataset are depicted in Fig.14 The predicted mean values for roll and pitch attitude obtained using the LSTM-AE method are found to be 3.5 degrees lower than those obtained from the other two methods. Similarly, for yaw, the mean value difference is 1.8 degrees. Notably, the

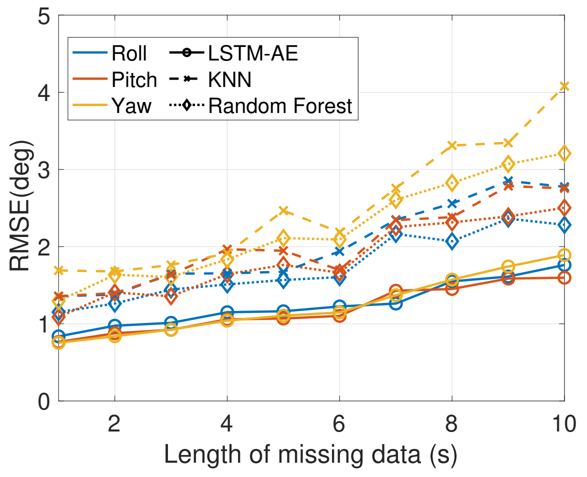

Figure 13. Performance comparison of LSTM-AE with KNN and Random Forest

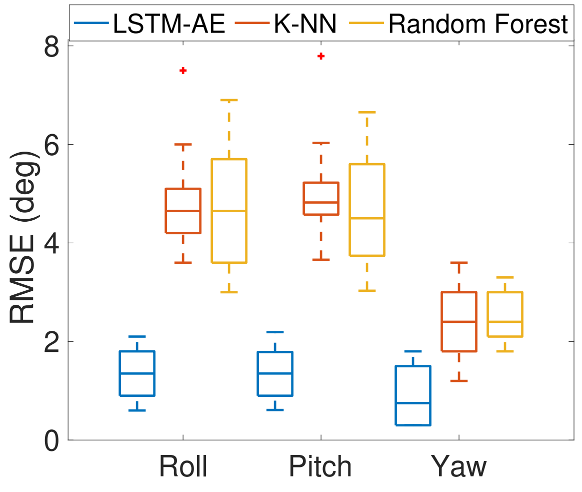

Figure 14. Box plot for data missing from 3 time series

variance of errors for roll and pitch is relatively small for LSTM-AE and KNN predictions, while Random Forest produced errors with larger variances, sometimes exceeding 6 degrees. This can be attributed to the fact that Random Forest does not take into account the correlation between different variables, resulting in poorer performance compared to the other methods. In cases where AHRS calculations are compromised, the proposed deep learning approach outperforms existing imputation models.

Accelerometer/Gyroscope Readings Missing - 6 time series

In the event of missing accelerometer readings, the resulting inaccurate AHRS calculations can generate erroneous/compromised information across 6-time series. This may lead to spurious or unreliable data, potentially affecting the overall accuracy and reliability of the system. Therefore, it is essential to ensure that accelerometer readings are consistently captured to minimize the risk of inaccuracies in AHRS calculations and subsequent data analysis. Missing acceleration data also flags the attitudes so that readings produced by AHRS are not used for decision-making. Similar to the previous cases, the length of missing data is increased from 1 second to 10 seconds.

The predicted attitudes and accelerations for the case where data is missing for 5 seconds are shown in Fig. 15 and 16 respectively. The blue and orange plots represent the actual and predicted values, respectively, for both attitudes and accelerations. It is evident that the predicted values closely match the true values, owing to the network's ability to correlate controls, angular rates, and barometer readings with the required variables. Fig. 17 and 18 present the overall Root Mean Square Error (RMSE) values for attitudes and accelerations across all three axes, respectively, as the length of missing data increases. As expected, the RMSE error increases with an increasing length of missing data for all axes. Notably, when using the LSTM-AE approach, the error in accelerations remains relatively constant $\left(0.3\frac{m}{s^2}\right)$, while it grows to $3\frac{m}{s^2}$ for the other methods when the length of missing data increases to 10 seconds.

Furthermore, Fig. 19 and 20 depict the box plots representing the variance of RMSE errors for attitudes and accelerations with a 5-second sliding window. LSTM-AE predictions outperform the other two algorithms, exhibiting a generally lower mean and a comparable variance for all three axes. Overall, these results demonstrate the effectiveness of the LSTM-AE approach in accurately predicting attitudes and accelerations, even in the presence of missing data for six variables.

Similarly, the situation where gyroscope readings are inaccurate or missing is also investigated. This also represents the scenario where data is missing from 6-time series simultaneously. The results are shown in Fig. 21 - 24 and the observations are generally similar to those reported earlier in this section for missing acceleration and attitude data.

Training using initial estimate from Kalman Filter

ArduPilot employs a combination of sensor data and advanced algorithms to calculate the vehicle's attitudes, i.e., its orientation in space. The specific algorithms utilized depend on the available sensors and the vehicle's mode of operation. Typically, ArduPilot uses a complementary filter algorithm that combines data from the gyroscopes and accelerometers to estimate the vehicle's orientation. The gyroscopes provide a continuous measure of the vehicle's orientation change rate, while the accelerometers offer a reference for the direction of gravity, thereby determining the vehicle's orientation.

Besides the complementary filter, autopilots can also leverage magnetometer data to estimate the vehicle's heading or direction. The magnetometer measures the strength and direction of the Earth's magnetic field, which serves as a reference for determining the vehicle's attitudes. Moreover, GPS data is incorporated if available, to enhance the accuracy of the attitude estimates. GPS provides the vehicle's position and velocity, which can be used to refine the orientation and heading estimates. Thus, the attitude calculation process involves the

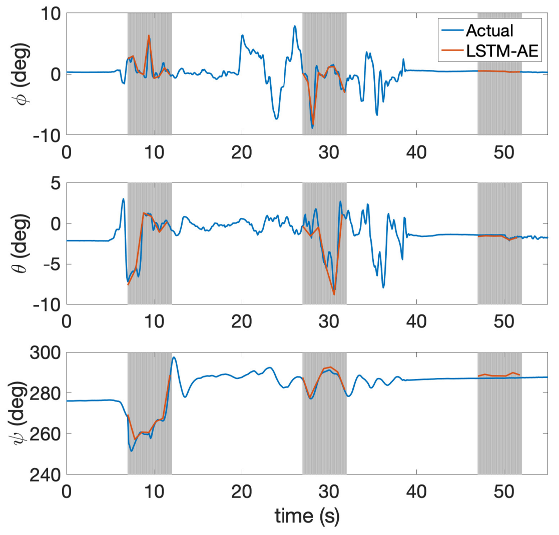

Figure 15. Predictions with acceleration data missing

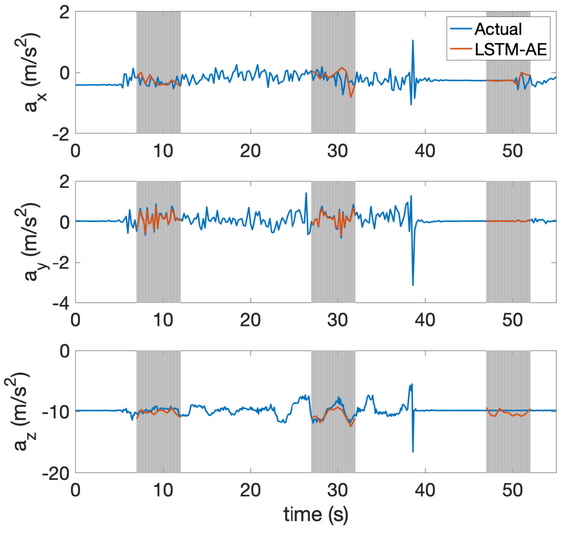

Figure 16. Attitudes predictions with accelerations data missing

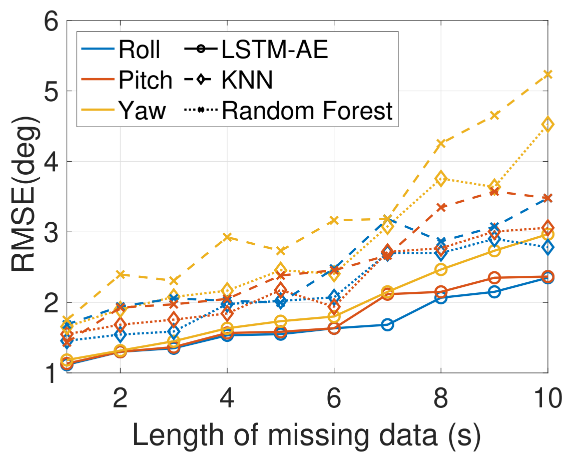

Figure 17. Predictions with data acceleration data missing

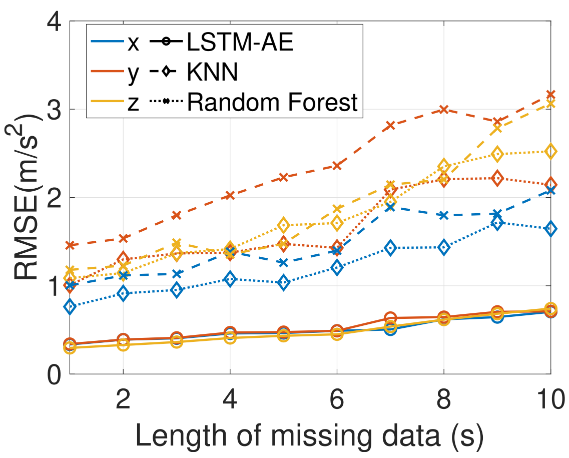

Figure 18. Attitudes predictions with accelerations data missing

fusion of data from multiple sensors using sophisticated algorithms to estimate the vehicle's orientation and heading in space.

Estimated Kalman Filter (EKF) is a mathematical algorithm widely used in robotics, aerospace, and other fields for state estimation and sensor fusion (Refs. 33–35). It is an extension of the traditional Kalman filter and is designed to handle nonlinear systems. The EKF algorithm combines sensor measurements with a mathematical model of the system to estimate the state of the system in the presence of noise and uncertainty. Most of the autopilots have EKF integrated in the system that uses a fault detection and exclusion (FDE) algorithm, which identifies and excludes faulty sensors from the estimation process.

The Extended Kalman Filter (EKF) can offer a reliable initial estimate of the states, which can subsequently enhance the training process. In this study, we consider the scenario where attitude data is missing. During the training phase, the EKF-predicted initial attitude values are utilized, followed by the model's standard training procedure. Fig. 25 displays the predicted attitudes using both the LSTM-AE and the new network, where the initial guess is based on the EKF-predicted values. However, no significant improvement in prediction accuracy is observed, as evidenced by the closely-following curves. The same can be validated using the RMSE values plotted for both methods for varying lengths of missing data (Fig. 26).

CONCLUSIONS

This study proposes a deep learning-based approach for imputing missing sensor data in multicopters, a critical task

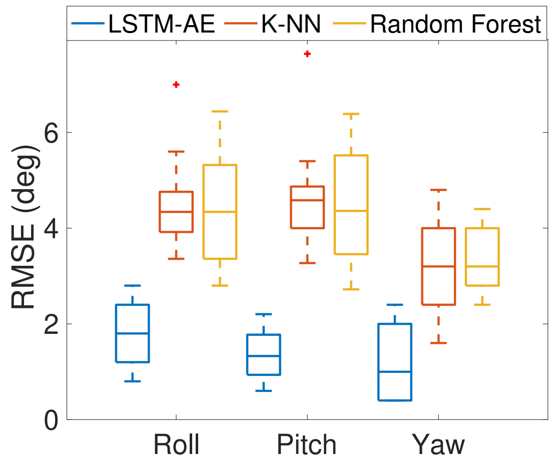

Figure 19. RMSE for attitudes with acceleration data missing

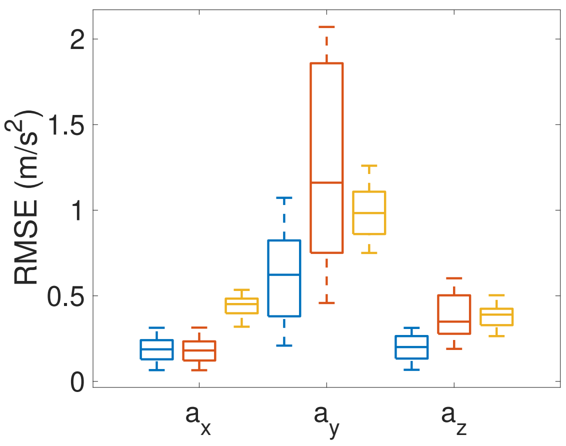

Figure 20. RMSE for accelerations data missing

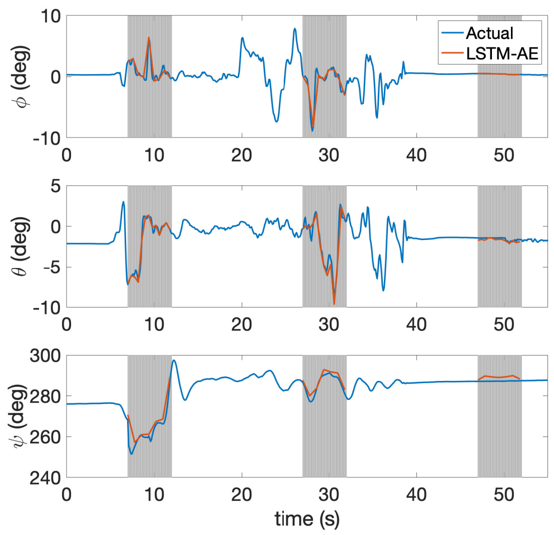

Figure 21. Predictions with angular rates data missing

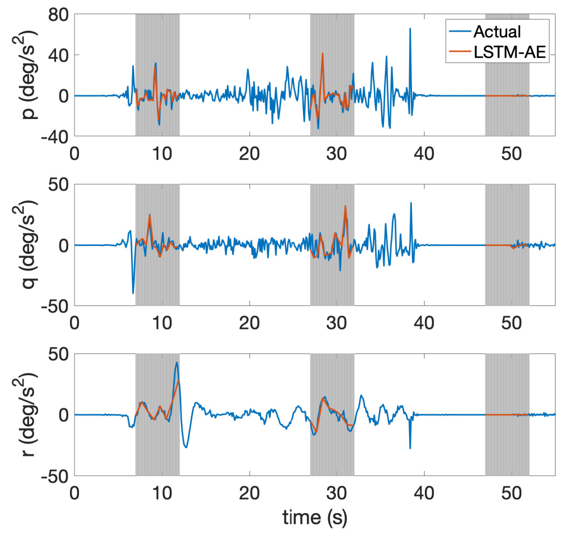

Figure 22. Attitute predictions with angular rates data missing

for safe and efficient operation of aerial vehicles. The proposed approach is based on two deep learning techniques, Autoencoders (AE) and Long Short-Term Memory (LSTM) networks. The neurons in AE architecture are replaced with LSTM units to integrate the two methods. The algorithm is evaluated using flight test data from a 2.5 kg hexacopter, and three different scenarios of missing data are considered. This allows testing the developed algorithm under realistic conditions and assessing its ability to handle different levels of missing data.

The algorithm is compared to two commonly used imputation techniques: k-Nearest Neighbor (KNN) imputation and Random Forest imputation. This comparison provides a benchmark for the performance of the proposed approach and demonstrates its superiority over traditional methods.

The results of the study indicate that the LSTM-AE trained network outperforms both KNN and Random Forest in terms of the accuracy of imputation. The network can handle missing data with varying degrees of sparsity, which makes it a more robust and flexible solution.

The results show that the LSTM-AE network outperforms the other two models by 50%-75% in cases where data is missing from a single time series. In situations where the AHRS malfunctions and data is missing for attitudes, the network trained using LSTM-AE performs 47%-82% better than the alternatives. Furthermore, the predictions for all the testing data using LSTM-AE have a lower mean, and lower variance, indicating few outliers. Similar findings are observed when accelerometer/gyroscope readings are missing.

The study also examined the effect of utilizing initial estimates obtained from the Kalman Filter to train the deep learning models. However, incorporating these estimates did not

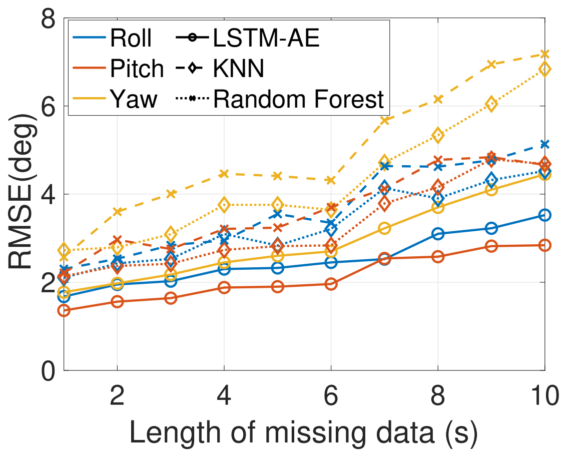

Figure 23. RMSE for attitudes with acceleration data missing

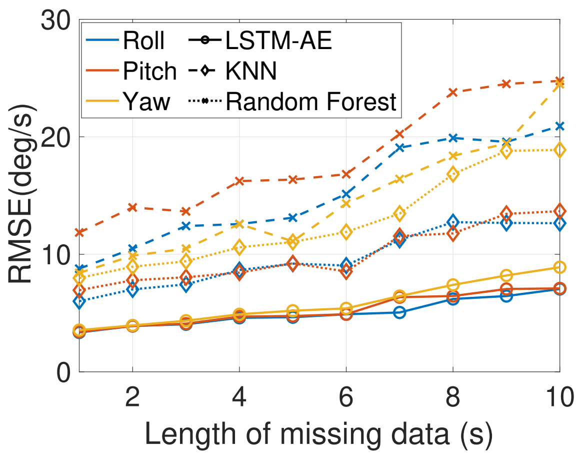

Figure 24. RMSE for accelerations data missing

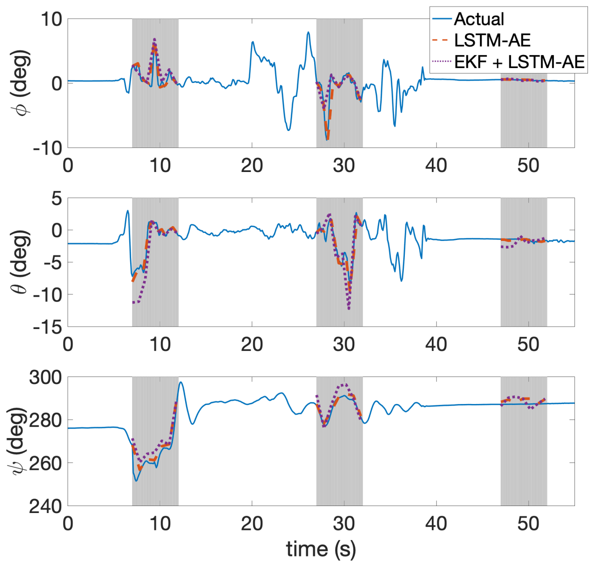

Figure 25. Predictions using LSTM-AE and EKF+LSTM-AE

lead to any enhancement in the accuracy of imputation. This indicates that the proposed method can discern the inherent patterns in the data without requiring supplementary information from the Kalman Filter

The research highlights the efficacy of deep learning methods in filling in missing sensor data in multicopters. The suggested approach surpasses traditional imputation methods with respect to accuracy and efficiency and can manage different levels of data sparsity making it a reliable and adaptable solution.

The outcomes of this study hold significant implications for the development and functioning of multicopters, emphasizing the need to address missing data to ensure secure and optimal operation of these airborne devices. The presented methodology holds promise for enhancing the safety and reliability of multicopters and has the potential to be implemented in other areas grappling with the issue of missing data.

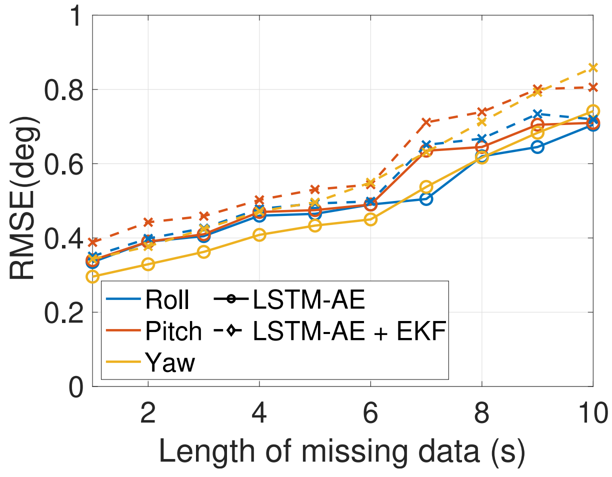

Figure 26. RMSE comparison for LSTM-AE method with EKF+LSTM-AE

REFERENCES

1. Khalastchi, E., Kalech, M., and Rokach, L., “Sensor fault detection and diagnosis for autonomous systems,” Proceedings of the 2013 international conference on autonomous agents and multi-agent systems, 2013.

2. Heredia, G., and Ollero, A., “Sensor fault detection in small autonomous helicopters using observer/Kalman filter identification,” 2009 IEEE International Conference on Mechatronics, 2009.

3. Khalastchi, E., Kaminka, G. A., Kalech, M., and Lin, R., “Online anomaly detection in unmanned vehicles,” The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, 2011.

4. Hashimoto, M., Kawashima, H., and Oba, F., "A multi-model based fault detection and diagnosis of internal sensors for mobile robot," Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453), Vol. 4, 2003.

5. Napolitano, M. R., Neppach, C., Casdorph, V., Naylor, S., Innocenti, M., and Silvestri, G., "Neural-network-based scheme for sensor failure detection, identification, and accommodation," *Journal of Guidance, Control, and Dynamics*, Vol. 18, (6), 1995, pp. 1280–1286.

6. Bonnabel, S., "Left-invariant extended Kalman filter and attitude estimation," 2007 46th IEEE Conference on Decision and Control, 2007.

7. Kendoul, F., "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Journal of Field Robotics*, Vol. 29, (2), 2012, pp. 315–378.

8. Smolyanskiy, N., Kamenev, A., Smith, J., and Birchfield, S., "Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness," 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017.

9. Jia, B., Feng, W., and Zhu, M., "Obstacle detection in single images with deep neural networks," *Signal, Image and Video Processing*, Vol. 10, (6), 2016, pp. 1033–1040.

10. Yi, X., Zheng, Y., Zhang, J., and Li, T., "ST-MVL: Filling missing values in geo-sensory time series data," Proceedings of the 25th International Joint Conference on Artificial Intelligence, 2016.

11. Shao, C., Fang, F., Bai, F., and Wang, B., "An interpolation method combining Snurbs with window interpolation adjustment," 2014 4th IEEE International Conference on Information Science and Technology, 2014.

12. Narayanan, S., Marks, R., Vian, J. L., Choi, J., El-Sharkawi, M., and Thompson, B. B., "Set constraint discovery: missing sensor data restoration using autoassociative regression machines," Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290), Vol. 3, 2002.

13. Abdella, M., and Marwala, T., "The use of genetic algorithms and neural networks to approximate missing data in database," IEEE 3rd International Conference on Computational Cybernetics, 2005. ICCC 2005., 2005.

14. Che, Z., Purushotham, S., Cho, K., Sontag, D., and Liu, Y., "Recurrent neural networks for multivariate time series with missing values," *Scientific reports*, Vol. 8, (1), 2018, pp. 6085.

15. Pratama, I., Permanasari, A. E., Ardiyanto, I., and Indrayani, R., "A review of missing values handling methods on time-series data," 2016 international conference on information technology systems and innovation (ICITSI), 2016.

16. Bańbura, M., and Modugno, M., "Maximum likelihood estimation of factor models on datasets with arbitrary pattern of missing data," *Journal of applied econometrics*, Vol. 29, (1), 2014, pp. 133–160.

17. Dhevi, A. S., "Imputing missing values using Inverse Distance Weighted Interpolation for time series data," 2014 Sixth international conference on advanced computing (ICoAC), 2014.

18. Sitaram, D., Dalwani, A., Narang, A., Das, M., and Auradkar, P., "A measure of similarity of time series containing missing data using the mahalanobis distance," 2015 second international conference on advances in computing and communication engineering, 2015.

19. Amiri, M., and Jensen, R., "Missing data imputation using fuzzy-rough methods," *Neurocomputing*, Vol. 205, 2016, pp. 152–164.

20. Garnier, A., Eynard, J., Caussanel, M., and Grieu, S., "Missing data estimation for energy resources management in tertiary buildings," CCCA12, 2012.

21. Valdiviezo, H. C., and Van Aelst, S., "Tree-based prediction on incomplete data using imputation or surrogate decisions," *Information Sciences*, Vol. 311, 2015, pp. 163–181.

22. "PX4 User Guide," https://docs.px4.io/main/en/flight_controller/cubepilot_cube_orange.html, [Online; accessed 17 October 2022].

23. Goodfellow, I. J., Bengio, Y., and Courville, A., *Deep Learning*, MIT Press, Cambridge, MA, USA, http://www.deeplearningbook.org, 2016.

24. Tschannen, M., Bachem, O., and Lucic, M., "Recent advances in autoencoder-based representation learning," *arXiv preprint arXiv:1812.05069*, 2018.

25. Meng, Q., Catchpoole, D., Skillicom, D., and Kennedy, P. J., "Relational autoencoder for feature extraction," 2017 International joint conference on neural networks (IJCNN), 2017.

26. Hochreiter, S., and Schmidhuber, J., "Long short-term memory," *Neural computation*, Vol. 9, (8), 1997, pp. 1735–1780.

27. Ma, J., Ding, Y., Cheng, J. C., Tan, Y., Gan, V. J., and Zhang, J., "Analyzing the leading causes of traffic fatalities using XGBoost and grid-based analysis: a city management perspective," *IEEE Access*, Vol. 7, 2019, pp. 148059–148072.

28. Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., and Woo, W.-c., "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," *Advances in neural information processing systems*, Vol. 28, 2015.

29. Olah, C., "Understanding LSTM Networks," , 2015.

30. Finley, A. O., McRoberts, R. E., and Ek, A. R., "Applying an efficient k-nearest neighbor search to forest attribute imputation," *Forest Science*, Vol. 52, (2), 2006, pp. 130–135.

31. Pantanowitz, A., and Marwala, T., "Missing data imputation through the use of the random forest algorithm," Advances in computational intelligence, 2009.

32. Kokla, M., Virtanen, J., Kolehmainen, M., Paananen, J., and Hanhineva, K., "Random forest-based imputation outperforms other methods for imputing LC-MS metabolomics data: a comparative study," *BMC bioinformatics*, Vol. 20, (1), 2019, pp. 1–11.

33. Maybeck, P. S., "The Kalman filter: An introduction to concepts," *Autonomous robot vehicles*, 1990, pp. 194–204.

34. Simon, D., "Kalman filtering," *Embedded systems programming*, Vol. 14, (6), 2001, pp. 72–79.

35. Catlin, D. E., *Estimation, control, and the discrete Kalman filter*, Vol. 71, Springer Science & Business Media, 2012.