UAV degradation identification for pilot notification using machine learning techniques

Anush Manukyan[†*], Miguel A. Olivares-Mendez[*], Tegawendé F. Bissyandé[*], Holger Voos[*], Yves Le Traon[*]
[*] University of Luxembourg, Luxembourg
forename.lastname@uni.lu
[†] forename.lastname.001@student.uni.lu

***Abstract*—Unmanned Aerial Vehicles are currently investigated as an important sub-domain of robotics, a fast growing and truly multidisciplinary research field. UAVs are increasingly deployed in real-world settings for missions in dangerous environments or in environments which are challenging to access. Combined with autonomous flying capabilities, many new possibilities, but also challenges, open up. To overcome the challenge of early identification of degradation, machine learning based on flight features is a promising direction. Existing approaches build classifiers that consider their features to be correlated. This prevents a fine-grained detection of degradation for the different hardware components. This work presents an approach where the data is considered uncorrelated and, using machine learning techniques, allows the precise identification of UAV's damages.**

I. Introduction

Robotics is a fast growing and truly multidisciplinary research field. It is at the crossing of the latest technologies in the areas of mechanical engineering, electrical engineering, as well as computer science.
There exist many types of robots, such as humanoid robots, robotic arms or unmanned vehicles, to name a few. Each brings its own set of unique challenges and constraints.
The purpose of Unmanned Vehicles, in particular Unmanned Aerial Vehicles (UAV), is to be deployed in dangerous environments and perform specific tasks or missions that are highly dangerous and life-threatening for humans. Such critical missions include, but are not limited to, defusing bombs, find survivors in ruins and exploring unstable mines and shipwrecks [1], [16].
UAVs can be remote controlled or autonomous (meaning that no human interaction is required). Flying autonomously brings up many new challenges and obstacles, because the UAV operates in an open space, which is typically unknown. Thus, the UAV would need to learn about the environment or have somehow outside help in order to avoid collisions during autonomous navigation, take-off and landing [2], [11], [17].
By letting the UAV gain awareness about its status and overall health of its different body parts (propeller, battery, protection hull, sensors, . . .), it can identify and understand failures and thus, make better decisions for a particular mission.
One of the best ways to solve these issues is to predict the UAV's behaviour based on the flight data, gathered by on board or external sensors, and use those predictions as notifications.
This work proposes an approach for UAV's hardware damage identification based on well-known machine learning techniques. At the core, the idea consists of analysing deviations between the UAV's actual flight data and its desired flight path and identify whether or not a degradation or damage is affecting the drone's performance.
The proposed machine learning approach uses k Nearest Neighbour (kNN) algorithm along with Dynamic Time Warping (DTW). The choice of kNN is motivated by the fact that it is simple to implement, achieves great performance and is flexible enough to support different types of data and distance measure functions. In this case, the data is flight paths, which are a time series of sensor values and coordinates. Therefore, DTW is a great choice, since it can compute the difference between two time series of different lengths. Besides, most existing approaches focus on correlated data [1], [2], but here, the individual features, such as latitude, longitude and altitude, are handled as uncorrelated. This has the advantage of allowing to detect the exact source of the failure or degradations.
To prove the viability of our approach, extensive experiments have been performed using a commercial UAV in order to test and measure the consequences of damaging propellers.

The paper is organised as follows: Section 2 covers the state-of-the-art research. Section 3 presents the core approach and describes the general model, based on machine learning best-practices, along with how the degradation identification process is designed. Then, Section 4 showcases the experimental setup with the different types of experiments, covers the observations and finally presents all the results and performances of the algorithm. At last, section 5 concludes the work achieved and describes the potential future directions.

II. Related Work

Anomaly detection of UVs recently became a major trend and investigation topic for researchers. This chapter presents the review of the current state of the art focuses on the work and the research that have been done on the area of anomaly detection of UVs and pattern recognition in time series using machine learning techniques.

A. Anomaly detection in Unmanned Vehicles

Raz Lin et al. [1] presented a novel approach for detecting anomalies in unmanned autonomous vehicles, based on their internal and external sensor readings using the Mahalanobis distance[13]. First is the pre-processing phase, they search for dependencies, between different internal sensors on the vehicles. It uses an efficient search method to identify subgroups of variables that are statistically dependent [1]. Thus, they find several distinct groups of variables, each of much smaller dimension than the initial set. Then, they identify abnormal values in each of the smaller-dimensional groups of variables using Mahalanobis distance [1]. Khalastchi et al. [15] presented another approach, which is the improvement of their previous technique [1]. In this work they present an online data-driven anomaly detection approach, based on sliding window technique, which allows mining frequent patterns over data streams. First, they filter the input to reduce noise, after they split the data into sets of correlated attributes to reduce the dimension of a data. And finally, they calculate the Mahalanobis distance for each set in order to return the degree of difference between sets of data being an outlier and by beforehand-defined threshold they declare anomalies. Das et al. [2] have pursued another research direction about anomaly detection in flight recorder data, based on dynamic data-driven approach. They offer a novel approach, standing on a feature extraction technique, called symbolic dynamic filtering, which extends the iOrca algorithm [2], making its performance faster and computationally less expensive. They divide time-series data into general symbol sequences, and build probabilistic finite state automata for using as features for pattern classification. They use this approach for anomaly detection and behaviour identification of mobile robots.

B. Machine learning approaches for classification of Time Series

As the data provided from UAVs are continuous time series data, we overview previous work that have been done on identifying and determining abnormal patterns in time series using machine learning approaches.
Spiegel et al. [4] worked on Pattern Recognition and Classification for Multivariate Time Series. Their approach starts by splitting a time series into segments, and then clustering the recognized segments into groups with similar contexts [4]. Lin et al. introduce a novel approach of symbolic representation of time series, that is suitable for streaming algorithms [3]. They propose an algorithm for dimensional and numerosity reduction of time series. Their approach is called Symbolic Aggregate ApproXimation (SAX). On the first step of the algorithm the data is transformed into the Piecewise Aggregate Approximation (PAA) representation and then symbolized into a discrete string. Basically, dimensional reduction, via PAA, reduces time series of n dimensions into w dimensions, (n¿w), by dividing the data into w equal sized frames. The calculated mean value of the data falling within a frame, where data-reduced representation is the vector of these values. At the last step, they use a distance measure function which is based on Euclidean distance and gives the minimum distance between the original time series of two words. Recently Li et al.[5] suggested a novel approach that uses n-gram language modelling techniques [5]. The proposed method, called Domain Series Corpus (DSCo), is based on previous work that has been mentioned in the state of the art [3]. It works by first building pre-class language models and then uses those models for segmenting the time series samples. Essentially, the algorithm classifier a time series by looking at what language model it belongs the most.

After investigation of existing research and work, it is clear that many different approaches and solutions have been proposed. However, most of them were focused on anomaly detection of autonomous robots where the altitude, longitude and latitude are correlated, which means that they can successfully identify anomalies, but do not detect the exact source of the failure. Thus, this work considers that the altitude, longitude and latitude are not correlated. This has the advantage of allowing to isolate each feature and to pinpoint the exact issue.

III. Approach

This section presents a new approach for identification of degradations.

A. Model

A typical scenario consists of a mission given to a drone, which is a pre-determined path, that the UAV follows autonomously in an arbitrary environment. During this mission, flight data is being recorded by internal and/or external sensors or other movement tracking systems. The captured data is in a raw format and, thus, must be pre-processed by performing various feature extraction techniques, such as reducing the dimension in order to facilitate the next computational steps in the model. After those first steps, the processed data should be stored in a convenient file format or database in order to easily perform further analysis.

Raw Data

As previously stated, UAVs have on-board sensors or external ones that are being employed for tracking and that record all kinds of data. Usually, this data is in raw format since it is unprocessed. Also, the recorded features or attributes are programmatically or manually defined beforehand.

Feature Extraction

Following general machine learning techniques, the gathered raw data should go through a feature extraction phase, also called pre-processing. In case of UAVs, a typical batch of data consists of several flights and each flight contains at least timestamped coordinates and other sensor specific data, such as battery level, speed, or even temperature. The raw data can contain a lot of data points, especially if they have been recorded via an external, high performance, sensor. Thus, dimensionality reduction techniques can be applied without any loss of quality or detail. Additionally, certain attributes, such as speed and heat, can be removed altogether, effectively focusing on location based data and reducing the size of the data in storage.

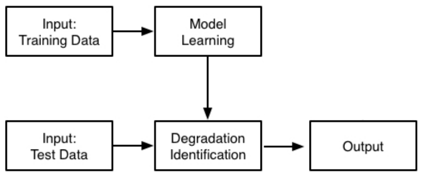

Fig. 1. General flow of the degradation identification process.

General Inputs

After the feature extraction is done, the next step consists of actually using the gathered knowledge for training the algorithm. It is crucial that when new test data is being collected, it must first go trough the exact same feature extraction process, which has been previously used for training the model, and then into the degradation identification process. Otherwise, many errors and inconsistencies can occur due to feature or format differences. The machine learning algorithm used in this approach, has to be trained once and then can predict the label of each incoming test objects. However, it is preferred to continuously try to improve the algorithm by extending or refining the training set or algorithm parameters as more data is collected over time.

Learning Model

The initial set of raw data needs to be labelled in order to train the machine learning algorithm. This is called model learning. In this case, a supervised learning approach is taken. In other words, the original data samples, representing each UAV flight sessions, are being labelled manually.

Degradation Identification

The last step comprises the identification of degradations of new UAV's flight data. This data is also referred to as the input test set. The goal is to yield an estimation about how good or bad a flight is, based on the learned knowledge from the training set. And in case of a bad flight, the cause, such as defect in the motors or a broken battery should be identified.

Outputs and Results

The final returned result presents information about the level of anomalous behaviour and points out the exact source of the issue.

B. Degradation Identification Process

The main objective of the degradation identification process is to categorize the output based on the learned samples. In short, it is a classification job. Thus, the model learning procedure can be supervised, unsupervised or semi-supervised. For solving the particular task presented in this work can be applied any of those three types of learning approaches. However, the later showcased proof of concept uses a supervised learning technique. The main task of this type of learning is to map a category to an observed input [6].

There exist many different machine learning algorithms, but the focus is on classification methods for time series. Still, many options are available such as k Nearest Neighbour [7], Support Vector Machines [8], Single Value Decomposition [10], Symbolic Aggregate approximation [2], and many more, for this purpose.

In this work, k Nearest Neighbour (kNN) classification has been chosen, due to its known simplicity, intuitiveness and ability to produce high performance results. Besides, kNN can handle time series effectively and can be easily extended to support other distance measures as it will be presented later in this section.
kNN, after being trained with labelled time series data, takes unlabelled time series as an input, along with a value for how many nearest neighbours will be retained and a function to compute the distance measure between pairs of objects. As a result, the algorithm finds the closest candidates relative to the training dataset and assigns the most frequent label to the given input test data.
Regarding the distance measure of the kNN classifier, it is required to have a function able to compare two time series of different sizes or lengths. Thus, the Dynamic Time Warping (DTW) has been selected, because of its specific feature of being insensitive to the difference of length of two time series when computing the distance between them. DTW basically finds the distance between two data sets by calculating the possible alignment warp, also called warping path, using a dynamic programming approach and selecting the one having the minimum value.

DTW has the disadvantage of not being very efficient when working with large data sets as it is computationally intensive. Fortunately, the algorithm can be improved by making use of a smaller warping window, effectively speeding up the algorithm by reducing the number of computations.

In the state of the art, it has been shown that most existing works have been focusing on correlated data. On opposite, the approach presented in this paper focuses on uncorrelated data, in order to achieve more detailed and isolated analyses. This means that x, y and z, respectively, latitude, longitude and altitude are studied individually.

IV. Assessment

This section presents in great detail each step of the experiments. The entire process from the initial experiment setup and data gathering phase to the final model training and performance evaluation.

A. Experimental Setup

The Automation and Robotics Research Group of the University of Luxembourg was having a dedicated space

at the Mudam museum in Luxembourg City, where they demonstrated via a show the possible collaboration of different kinds of robots, such as a humanoid robot and a drone. Since, it was running in a closed and relatively small environment, safety nets surrounded the scene to avoid the drone from flying into the public. In addition, a large table (2.5 x 2.5m) was used as a scene for the robots and a motion tracking system (OptiTrack) has been installed. Figure 5.1 showcases the entire setup as seen by the public.
For all shows, and thus, all our experiments, the commercially available AR.Drone [9] has been used. This UAV was modified by gluing 6 small motion capture markers on it, which allows OptiTrack [12] to follow the moving object and record every coordinate. Figure 5.2 shows the AR.Drone with the markers installed.
The public show or exhibition was about a humanoid robot communicating with the drone and interacting with the public. The complete scenario takes about 12 minutes and for the entire duration, the robots were performing autonomously.
Since, this paper focuses on the drone data, the scenes related to the humanoid robot were fully omitted from the context. This made experiments faster, effectively reducing the duration to 1 minute and 40 seconds. In addition, this had the effect of skipping the parts where the drone waits on the table for several minutes without flying.

The entire show consists of four short parts. Each part starts with a take-off and ends with a landing. For simplicity, we refer to one each show or run as one flight. Here are the different parts:

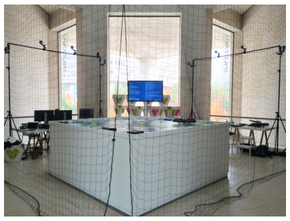
Fig. 2. The setup at the musem as seen from the public

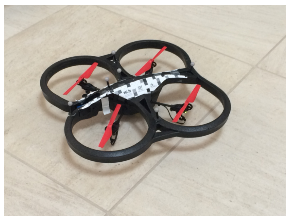
Fig. 3. AR.Drone with the markers installed

- Part 1: The drone takes off from the table and flies upwards before hovering a few seconds. Then it goes back downward and lands on it's initial position.

- Part 2: The drone takes off from the table and flies upwards. Then, it flies to the right in an aggressive fashion and goes back to the left more gently. Finally, it goes downward and lands also on it's initial position.

- Part 3: The drone takes off from the table and flies upwards. Here, it starts flying to the right and makes a semi circle. At the end, it goes downward and lands on the table at a different location.

- Part 4: The drone takes off from the table and flies upwards. It performs a complete circular flight over the table and finished by landing on his initial position.

It is implemented into the show's computer, which controls the drone, that no exact speed parameter is given to the drone. Instead, a time duration is given for each step. Thus, when the drone is instructed to fly to a certain coordinate in a certain time, the controller computes the manoeuvre and speed required.

B. Experiments

Since, the focus of this work is towards identification of degradations, the goal of the experiments are to find out how the AR.Drone behaves with different kind of damage. The idea is to introduce damage on one part at a time in order to destabilize the drone.

Using the environment at the museum, several experiments have been setup and each one has been performed multiple times in order to record enough data to build the training, validation and test data needed for the proof of concept. Bellow is presented the various experiments performed, along with some descriptions of what has been observed for each case.

We have chosen to mainly play with the different levels of damage on propellers, but the upcoming analysis and the approach, presented here, can also be adapted for motors, battery and their associated time series.

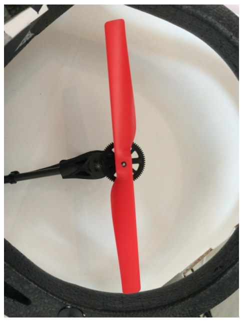
Fig. 4. New propellers

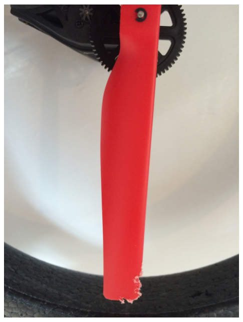
Fig. 5. Slightly damaged left propeller

The 1^{st} stage of the experiments used four completely new factory propellers as shown on figure 4. As expected, the drone had a smooth and stable flight and followed the scenario's path very closely. Thus, this flight has been considered as the desired and best one. This information is important, because it will be used later for correctly labelling the initial data.
The 2^{nd} step consisted of slightly damaging the left propeller on the back side, by scraping away a small amount of plastic as seen on figure 5. The change resulted in a small decrease in the flight quality. A noticeable degradation was the fact that the UAV did not fly up smoothly all the time and had a very small loss of speed.

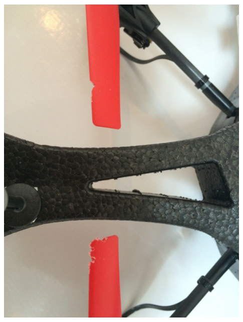
Fig. 6. Front-left damaged propeller

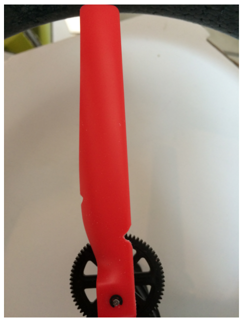
Fig. 7. Scarped front-right propeller

For the 3^{rd} experiment, the left propeller on the front side has been damaged in a similar way as the one on the left back side. Surprisingly, this did not make much of a difference, except that landing position was not equal to the take off position by a few centimetres. Figure 6 shows the damaged propeller.
Then, for the 4^{th} experiment, the right propeller on the front side has been scarped on both sides of the blade, as shown on figure 7. Again, the resulting behaviour was quite similar to the previous experiments, but there were a little more instabilities and, thus, more corrections are done by the drone's controller.

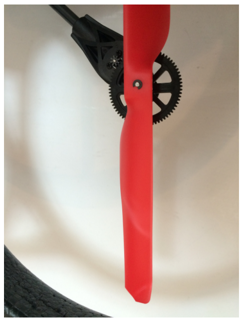
Fig. 8. Front-right propeller cut at the edge

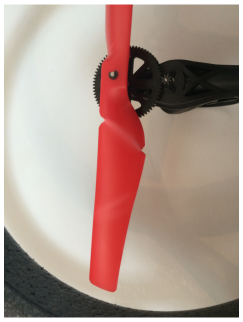
Fig. 9. Front-right bended propeller

The last three experiments did not affect to UAV's flight significantly. As a result, it became obvious that if a propeller loses some small part of plastic, it can still fly smoothly. Therefore, for the 5^{th} experiment, the decision was made to cut and bend the propeller which would soften the plastic and lead to increasing degradations in the flight. Figure 8 presents the front right propeller, which is cut at the edge and bended on the middle.
It turns out that bending propellers does, indeed, create many disturbances. Already during the take off, the UAV didn't fly straight up but went off course and flew over the table's border. At the end, it also landed, quite brutally, many centimetres farer away from the desired landing position.
The reason for this sudden worsening is that scarped or slightly cut propellers does not create enough disturbances. Thus, the drone's controller is able to self-balance and counteract the issue easily. Only a slight decrease in acceleration and smoothness is felt. However, the act of bending propellers completely changes the natural movement and interaction with the air while rotating. Therefore, the controller gets confused, since it is not programmed to handle such a level of damage.
For the 6^{th} experiment, the right propeller in the front has been bended, which had been already damaged a first time during the third experiment. Figure 9 shows the damage done. With two bend propellers and two other slightly scraped, the controller's job was getting even harder. Within the first seconds of flight, the degradations were clearly visible. The most significant bad behaviour was that the UAV started to whisk horizontally, as it was struggling to keep hovering in one place and need to constantly make small, but strong corrections.

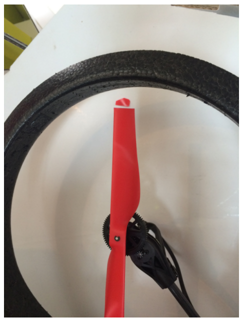
Fig. 10. Cut propeller

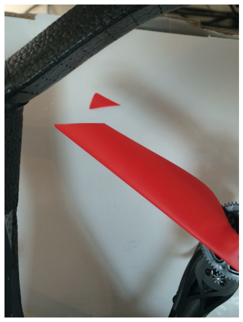
Fig. 11. Cut and bended propeller

Since the attempt to degrade the flying capabilities of the drone was looking successful by bending propellers, the 7^{th} experiment consisted of cutting a bigger chunk off the propeller as shown on figure 10. If the drone behaves with the same level of instability it did during the fifth experiment, then it validates the fact that cutting does not influence the drone much, at least less than bending.
Last but not least, for the 8^{th} experiment, an even bigger piece of the propeller has been cut and multiple bends has been added as it can be seen on figure 11. The resulting show was, by far, the worst one. Clearly, the UAV had a lot of trouble taking off and following the desired path. At many occasions, it was sliding off course, sometimes even for one to two full seconds, effectively crashing into the safety nets and never being precisely where it should be. It constantly lost altitude, forcing the controller to adjust aggressively. While landing, the drone could not stabilize and was rotating to the right. This was most probably due one propeller producing less power and thus naturally pushing the drone in one direction. At last, it has been previously stated that all experiments were run multiple times.

To conclude the experiments, it has been observed and recorded what the different kinds of damages, or degradations, affects the drone's flying capabilities. Next step is to gather raw data and evaluate it using the presented approach.

C. Evaluation and Results

Before the experiments can be evaluated and the results interpreted, the collected data samples' structure needs to be first understood in order to effectively build a pre-processing phase. Note that all further analysis has been performed using offline techniques. Once processed, the initial labelling method of the data will be presented, which will lead to the actual implementation of the kNN classifier with DTW.

A total of 60 experiments has been performed, resulting in 60 rosbags files, which were later split into 120 CSV files (1 GB), because AR.Drone and OptiTrack message data were contained in the same Bag file. This leads us to the actual content of the CSV files, before any pre-processing.
On one hand, one message, generated from the AR.Drone, is referred as "navdata" and contains about 36 different attributes about the current status of the drone at any time.
On the other hand, the message data, recorded from OptiTrack, contains only position and orientation information about the tracked rigid bodies, in this case the AR.Drone only. About 10 different attributes are saved.

For this work, the OptiTrack data is the most interesting and useful, since it describes the position and orientation of the drone at a rate of 240 recordings per seconds. This high rate is needed for the system to control the show in real-time. However, for analytical and data mining purposes, it generates more data than is actually needed. Additionally, only the position and timestamp will be kept for the degradation identification process.
Therefore, pre-processing techniques are applied to reduce and clean the data in a first place. Here, reduction consists of diminishing the number of recordings per seconds and still keep the same quality. This has been done by simply removing duplicate coordinates for each timestamp. Two values were considered the same if there is not more difference than 0.01, which is equivalent to 1 centimetre.
Furthermore, through cleaning, the unused attributes are being purged, effectively keeping only header_stamp_secs, pose_position_x, pose_position_y, pose_position_z attributes.

For the proof of concept, a web service has been implemented in order to perform the pre-processing steps presented above. The web service is developed using Spring Boot, a Java web framework. It provides a user interface that allows the upload of CSV files. The service then processes it directly and saves the result as a new CSV file on the server. Generally, for this case, each CSV has been reduced by 80% in size.

As an alternative, the web service can also be used as an API, which allows to build simple automation scripts that, for instance, upload multiple CSV files. It has also the advantage of being the first step toward building an online detection tool and can be extended easily to support more functionalities.

Once the pre-processing is achieved, the next step is to build training data. As stated in section 3, the presented approach uses a supervised learning technique, which requires manual labelling of the training samples.

In section 4.1, it was described that the first experiment, with all four new propellers in perfect conditions, has been considered as the best flight. Thus, the recorded path of this flight will be the desired path and the main reference point for evaluating the quality of the other flights. Since, the classification algorithm used for this work uses DTW as a distance measure, it is safe to assume that using DTW to label all samples is a reliable approach. Essentially, the distance between the best flight and each other flight is computed and ordered by increasing value. So, the first element is the second best flight and the last flight is the worst, compared to the desired one. This ordered list can then be split into three parts, one for each of the following class labels: "Good", "Bad", "Worst".

The result of computing the DTW of the 51 full flights resulted in some interesting insight. The second best flight, which is the first element in the list, has a DTW value of 60. On the contrary, the last or worst flight has a value of 616. The average DTW distance is at 208. The data set actually comprises many average (bad) flight and only a few good and worst ones. This is not an optimal data set. The best is to have completely uniformly distributed samples.

Figure 12 shows the graphical representation of the DTW values (x-axis), relative to the best flight, of all 51 flights (y-axis), ordered in from smallest to biggest difference. Figure 13 presents the distribution of the experiments grouped by their DTW value. It is obvious form that graph that most flights are average, with DTW values between 150 and 180.

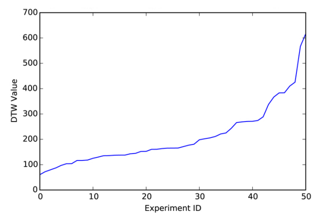

Fig. 12. DTW value between first show and each other show.

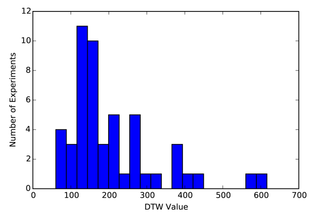

Fig. 13. Distribution of experiments grouped by DTW value.

Based on those values and on the observed drone behaviour during the experiments, we have chosen to assign a range of DTW values to a specific class label as shown on table I:

TABLE I
DTW VALUE LIMIT FOR EACH LABEL.

Min DTW	Max DTW	Class Label	Flight quantity
0	150	Good	19
150	350	Bad	25
350	∞	Worst	7

Looking at how the table showing that most flights are being labelled as "Bad", highlights the fact that for half of the experiments, the UAV's on-board controller could adapt to the degradations and keep flying. However, once the damage reached a more critical level, the DTW values also increases more dramatically.

D. Classification

With the labelled data built and ready to use, the classification process can be performed. In this work and proof of concept, the classifier used is k Nearest Neighbour with Dynamic Time Warping as a distance measure, as it was previously stated. The goal of this classification is to predict the level of damage of propeller of new flight paths. The classifier will take the unlabelled input flight and return the label associated to the closest flight that has been learned during training.

The first step, as for most machine learning algorithm, is to select the model and then evaluate its performance. Model selection consists of choosing the best values for all free variables in the algorithm, which are in this case the number of neighbours, k, and the size of the warping window, w. Multiple models are created and their performances are measured based on their accuracy. As a reminder, accuracy is the ratio of correct predictions.
To achieve this, the initial and complete data set was randomly split into two disjoint subsets: 90% has been used for training and 10% for testing. Then, the training data has been again randomly divided into two subsets: 70% has been used for the actual training and the remaining 30% for validation. In essence, the training set is used for finding the best parameters for the model and the validation set is used to tune those parameters. At last, using the best value for the model's parameters, the entire training data is used to train the classifier and the performance of the algorithm is estimated on the test data.

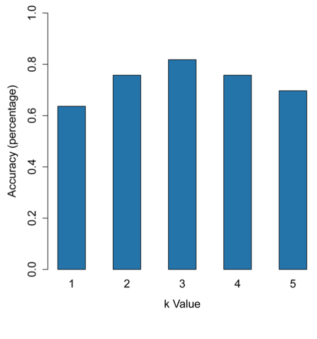

Fig. 14. Model comparison for position z.

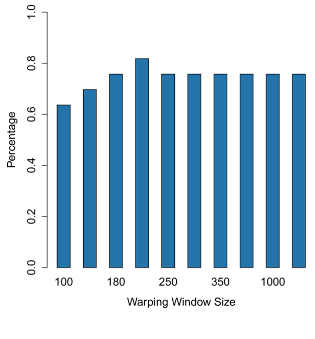

Fig. 15. Model comparison.

The strategy for finding the best parameters for the proof of concept has been to first find the best value for k, among the values 1, 2, 3, 4 and 5, given a very large warping window w of 4000. As was mentioned before, the warping window limits the number of calculation done. Therefore, a high number makes sure that nearly no calculations are skipped. However, the computation takes much more time and is thus only interesting in this case, for first finding the best k.

Starting with position z, figure 14 shows that the best value for k is 3, achieving 81% accuracy. For this model, the warping window has been set to 4000.

With the best k of 3 defined, the next step consists of tuning, or more precisely, reducing the maximum warping window value, such that the overall computation is quicker, for the same result. Thus, 9 new models have been trained and tested for the following values for w: 100, 150, 180, 200, 250, 300, 350, 500, 1000, 4000. Figure 15 presents the results of these 9 modes trained with $k = 3$ value. It is obvious that when the warping window size is 200 algorithm has the best performance. However, it is interesting to note that once a certain value reached, the warping window doesn't affect the accuracy anymore. A warping window of 250 yields the same performance as for 4000, yet needs less than half the time to be computed. The warping window size of 200 is only chosen once and is used for all following analysis, because as it will be presented shortly, it turns out that it achieves similar great result for position x and y.

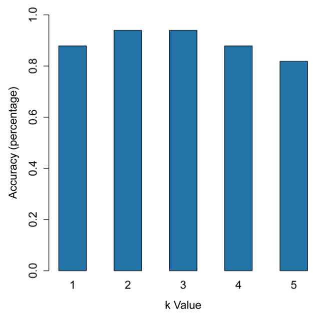

Fig. 16. Model comparison for position x.

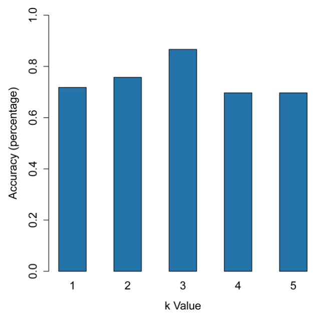

Fig. 17. Model comparison for position y.

Figure 16, shows that for position x, the best value for k, using a warping window of 200, is 2 and 3. Both models achieve an impressive 93% accuracy at most and 81% for the worst case of k equal to 5. Finally, for position y, presented on figure 17, the result looks more similar too position z. There, the best value for k is 3 and provides an accuracy of 86%. The lowest value is 69% for k equals to 4. Recall that the data is uncorrelated, so it is perfectly normal to have different best ks for each feature.
At this stage, the optimal parameters have been defined using the training and validation sets. The final step consists of training the model using the entire 90% training data and evaluate it against the remaining 10% of test data, which will provide the real performance of the model. Table II summarizes the final results on the test data for each position x, y and z, with their corresponding parameters. The proposed model achieves an accuracy between 85% and 91%. Those results have been produced by the proof of concept, that implements the presented machine learning

TABLE II
THE FINAL RESULTS ON THE TEST DATA FOR EACH POSITION X, Y, Z.

Feature	Accuracy	K	Max Warping Window
X	0.9075%	2	200
Y	0.8575%	3	200
Z	0.8866%	3	200

algorithms presented here, that is kNN classifier with DTW. The implementation is written in Python and made available on GitHub [14]. The choice for Python is motivated by the numerous libraries available for handling large data sets, such as numpy. Additionally, the language itself allows for short yet powerful code. The implementation allows to be installed locally and be used as a command line tool. In its basic usage, it takes as input parameters k and w, mentioned above, along with the training data and the test data, whose labels will be predicted. Other features are included, such as finding the best k or w and printing the confusion matrix. Note that, the Python classes implementing those functionalities, can be easily ported into a web service, allowing for remote usage or for running on a more performant server.

As final words about the validation of the models, it has been proven that well-known algorithms, in this case kNN and DTW, can be applied on new areas, such as UAV flight data. Those simple algorithms are relatively straightforward to implement and yield great results and performance. Thus, they are good candidates for a first approach to a solution. In addition, this proposed approach can be further used for any other kind of time series data, like battery or motor data. The only important point is that before being able to predict the new data, the model needs to be optimized and trained for it as it has been done for the position data with the propeller related experiments.

V. CONCLUSION AND FUTURE WORK

This paper focused on the issue, that fully autonomous drones are facing numerous challenges, such as navigation and environment awareness and need to have the capability to react to errors, anomalies and damages or degradations over time. It presented an approach that identifies degradations of UAVs, using well-proven machine learning techniques on the flight data. Presented approach is different from existing solutions in a sense that it handles x, y and z as uncorrelated. Thus, it is assumed that a degradation of some kind does not necessary affect all directions or capabilities of the drone. Which means that it is not only successfully identifies the anomalies but also pinpoints the exact source of the failure.

The model is based on kNN with DTW as a distance measure to achieve classification of time series of different lengths.
Despite the fact that this work is based on offline processing, an online solution is possible and is a logical extension as a future work. The idea being that the classification of new incoming flight data is done during the UAV's flight. It can be real-time or work with batches of data points. One approach consists of using an expanding window, where the data is buffered as it arrives and the classification is performed on the entire buffer. This approach is, however, limited by the total flight time, since the buffer grows as time passes. Furthermore, as the algorithm runs in short regular intervals, it slows down as the data sets grows.
Another possible extension is to use other classification algorithms for handling time series effectively and efficiently. A recent approach suggests a novel approach that uses n-gram language modelling techniques [5]. The proposed method, called Domain Series Corpus (DSCo) [18], is based on previous work that has been mentioned in the state of the art [3]. It works by first building pre-class language models and then uses those models for segmenting the time series samples.

To conclude, it has been shown that the level of degradation of a UAV can be predicted based on the analysis of flight coordinates using different approaches.

REFERENCES

[1] Lin Raz, Eliyahu Khalastchi, and Gal Kaminka. "Detecting anomalies in unmanned vehicles using the mahalanobis distance." Robotics and Automation (ICRA), 2010 IEEE International Conference on.

[2] Das, Santanu, et al. "Anomaly detection in flight recorder data: A dynamic data-driven approach." American Control Conference (ACC), 2013.

[3] Lin, Jessica, et al. "A symbolic representation of time series, with implications for streaming algorithms." ACM 2003.

[4] Spiegel, Stephan, et al. "Pattern recognition and classification for multivariate time series." Proceedings of the fifth international workshop on knowledge discovery from sensor data. ACM, 2011.

[5] Li, Daoyuan; Bissyandé, Tegawendé F.; Kubler, Sylvain; Klein, Jacques; Le Traon, Yves. "Profiling household appliance electricity usage with n-gram language modeling." The 2016 IEEE International Conference on Industrial Technology.

[6] Lin, Jessica, et al. "Pattern recognition in time series." Advances in Machine Learning and Data Mining for Astronomy 1 (2012)

[7] Gou, Jianping, Taisong Xiong, and Yin Kuang. "A novel weighted voting for k-nearest neighbor rule." Journal of Computers 6.5 (2011)

[8] Hearst, Marti A., et al. "Support vector machines." Intelligent Systems and their Applications, IEEE 13.4 (1998): 18-28.

[9] `http://ardrone2.parrot.com`, (November,2015)

[10] De Lathauwer, Lieven, Bart De Moor, and Joos Vandewalle. "A multilinear singular value decomposition." SIAM journal on Matrix Analysis and Applications 21.4 (2000): 1253-1278.

[11] Olivares Mendez, Miguel Angel, and Pascual Campoy. "Vision Based Fuzzy Control Approaches for Unmanned Aerial Vehicles." 16th IFSA/9th EUSFLAT. 2015.

[12] OptiTrack Wiki
`http://wiki.optitrack.com`, (December, 2015)

[13] De Maesschalck, Roy, Delphine Jouan-Rimbaud, and Dsir L. Massart. "The mahalanobis distance." Chemometrics and intelligent laboratory systems 50.1 (2000): 1-18.

[14] GitHub repository of the proof of concept
`https://github.com/anumanu/uav-degradation-identifivation.git`, (February, 2016)

[15] Khalastchi, Eliahu, et al. "Online data-driven anomaly detection in autonomous robots." Knowledge and Information Systems 43.3 (2015): 657-688. APA

[16] Renckens, I. R. "Automatic Detection of Suspicious Behaviour." (2014).

[17] Olivares-Mendez, Miguel A., Somasundar Kannan, and Holger Voos. "Vision based fuzzy control autonomous landing with UAVs: From V-REP to real experiments." Control and Automation (MED), 2015 23th Mediterranean Conference on. IEEE, 2015.

[18] Li, Daoyuan; Bissyandé, Tegawendé F.; Klein, Jacques; Le Traon, Yves. "DSCo: A Language Modeling Approach for Time Series Classification." The 12th International Conference on Machine Learning and Data Mining (MLDM 2016).