Anomaly Detection in Motion Timeseries using the Bosch XDK and Dynamic Time Warping

Julián Rico Mejía[1], Oscar Aguilar Aguila Isaías[2], Priyanka Paschapur[3]

Abstract: This paper presents the development of an anomaly detector for robotic movements using the dynamic time warping (DTW) algorithm and its implementation in Matlab. Data was collected by mounting the Bosch Cross-Domain Development Kit (XDK) sensor on a collaborative robot arm (Cobot), aiming at industrial applications in need for motion anomaly detection during repetitive tasks. The paper discusses practical issues like parameter tuning as well as algorithmic variants such as decoupling accelerometer and gyroscope data.

Keywords: Machine Learning, Dynamic Time Warping, Anomaly Detection, Bosch XDK, Collaborative Robot.

1 Introduction

Anomaly detection refers to the problem of finding patterns in data which do not conform to expected behaviour. It finds extensive use in a wide variety of applications such as fraud detection for credit cards, insurance or health care, intrusion detection for cyber-security, fault detection in safety-critical systems, and military surveillance for enemy activities [CBK09].

Anomalous data detection has been studied within statistics as early as the late 19th century. Nowadays, anomaly detection algorithms have been automated and systemized in many fields, such as computer science, especially in statistics applications, machine learning, data mining and information theory [BJ15].

The specific characteristics of an anomaly detection problem are determined by several factors. Important aspects of any anomaly detection technique are the nature of the input and output data, the type of anomaly, and the availability of labeled data. The input data is generally a collection of data instances and each data instance can be described by a set of attributes. The output data defines how the anomaly should be reported, by labels such as normal/anomalous or by scores, where each instance is assigned an anomaly score. The

[1] Hochschule Darmstadt, Fachbereich Elektrotechnik und Informationstechnik, Schöfferstraße 3, 64295, Darmstadt, julian.mejia@stud.h-da.de

[2] Hochschule Darmstadt, Fachbereich Elektrotechnik und Informationstechnik, oscar.isaias@stud.h-da.de. Stipendiary from CONACyT and DAAD

[3] Hochschule Darmstadt, Fachbereich Elektrotechnik und Informationstechnik, priyanka.paschapur@stud.h-da.de

nature of attributes can be of different types, such as binary, categorical, or continuous, and determines the applicability of different anomaly detection techniques [CBK09].

Movement recognition has been an active research area, with many practical applications in robotics, human-computer interaction, physical rehabilitation systems, and sign language. Nowadays, sensors with individual sensing capability are used in this field [CBK09]. Our research obtains the input data from the Bosch XDK sensor that allows for several sensing capabilities at once, especially those relevant for motion.

Different kinds of algorithms such as Support Vector Machines, replicator Neural Networks, and Density Estimation, have been used to implement Anomaly Detection [Ho14]. However, these methods compare similarly behaving data dimensions against each other, and classify sudden divergence as suspicious. In robot anomaly detection, many of the dimensions cannot be compared by common distance metrics as Euclidean distance. Dynamic Time Warping manages to compare timeseries that are not equal in size or that are similar but not fully synchronous by allowing one data point to be associated with more than one data point in the opposing timeseries.

The goal of this paper is to use DTW and machine learning on the data of robot movements and detect anomalies. Clustering of time series was used in the past by [ESL19] for data preprocessing in trajectory learning (motion) algorithms and [Ma19] who focused on anomaly detection of network traffic using DTW, applying this method on a labeled dataset for measuring the detection accuracy for different anomalies and providing the starting ground for our research. However it was further expanded to compare the analysis of dependent and independent DTW algorithms, as used in [Sh17], where the advantages and disavantages of dependent and independent DTW approaches were analyzed.

Henceforth, this work comprises a study analyzing the algorithm's performance applied to Anomaly Detection considering multiple factors such as different approaches to the DTW algorithm while using the Bosch XDK sensor as a data acquisition kit.

Section 2 of the paper presents the components of our system; section 3 describes the implementation of the algorithm; section 4 the performance criteria applied to our model; section 5 provides insight into the collaborative robot's movements, and section 6 details our analysis of the adaptability of DTW for this application.

2 Main components of the system

2.1 Bosch Cross-Domain Development Kit (XDK)

The Bosch XDK is, according to Bosch [Cr20], the "Swiss army knife" for IoT, and can be used for rapid prototyping in different areas of automation. It is an almost ready sensor which requires minimum programming experience to use [Bo20]. This paper utilizes two of its sensors for data collection: the accelerometer and the gyroscope, to analyze

movement. This data is comprised of accelerometer data in directions x, y, z and gyroscope's data in directions x, y, z and finally, their respective timestamp.

2.2 Dynamic Time Warping

Finding anomalies based on a collection of samples without hard criteria for the distinction between normal and anomalous movements requires a comparison rule. Similarities between these samples can be assessed in a consistent manner. As the recordings from an accelerometer and a gyroscope are presented to the identification algorithm as a vector containing the sampled data throughout a single execution of the movement to be identified, the first method that comes to mind for determining the similarity between sequences is using the Euclidean distance between them.

However, for the given application, the Euclidean distance will fail to account for differences between two timeseries, which may not have the exact same length (thus having different amount of points to compare in a pairwise manner) or may contain similar information occurring at different moments. The first issue will render the Euclidean distance mathematically impossible to implement without ignoring chunks of information and the latter will cause a high distance output for sequences that may be very similar but have delays with respect to one another. These two issues are represented in Fig. 1, which depicts two timeseries with different lengths but similar behavior. Data contained within the yellow circle (A) would have to be ignored to implement the algorithm and data in the green circles (B and C) is expected to be associated together, but it would not be the case [Co17].

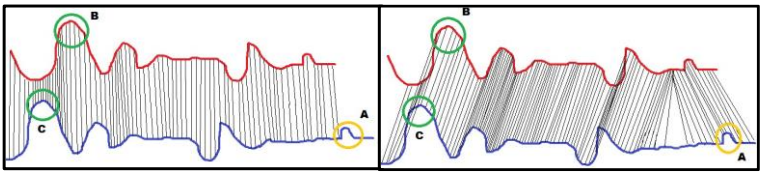

Fig. 1: Euclidean (left) vs DTW (right) Matching

The Remaining Useful Life (RUL) similarity models are used to predict when a given machine or equipment will fail to take preventive actions beforehand. Different approaches to this exist but the Pairwise Similarity Model computes the distance between different timeseries, which is fitting for our goal [RU20]. There are several approaches to achieving this but for the sake of this application the Dynamic Time Warping (DTW) was selected as its implementation is not challenging and it solves the comparison issues that arise from Euclidean distance measures.

DTW manages to compare timeseries that are not equal in size by allowing one data point to be associated with more than one data point in the opposing timeseries. Moreover, each measurement will be associated with a corresponding one from the opposing timeseries based on its Euclidean based similarity to measurements in the vicinity. Thus, two points

that are very similar but not taken at matching times will be linked together which accounts for differences in data capturing of the movements to be processed for anomaly detection.

As opposed with Euclidean distance matching, it can now be seen from the previous graph that all data points are used to determine the similarity of the blue curve with respect to the red one and that data within the green circles is linked together, as expected.

2.3 Classification of Distance Measures

Once a consistent method for assessing the similarity between two timeseries has been defined, the processing of this information must be established. As the system is intended to be implementable on virtually any movement (even non-robotic), a statistics-based approach seems viable.

After DTW is performed to the incoming measurements, information is reduced from a multidimensional representation into a one-dimensional distance measurement. Given that there are enough measurements, for a given application it can be assumed that the distance of each new measurement to a selected template will follow a normal distribution. From fitting the training data to a normal distribution, the first two parameters of the anomaly detector appear: the mean and the variance [Ma20].

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \mu)^2 \tag{2}$$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{1}{2}\cdot\left(\frac{x-\mu}{\sigma}\right)^2} \tag{3}$$

Equation 3 is commonly known as probability density function. This will provide the likelihood of the occurrence of a given input, in our case, a distance to the template. By establishing a threshold of likelihood for the distances, the third parameter for the anomaly detector arises.

The main idea is to test the likelihood of every distance obtained to a template and if this result fails to meet the threshold then the measurement will be considered as anomalous [Ma20]. It is worth noting that this must only occur for measurements whose distance lays in the right tail of the probability density function. If this distinction is not considered, then measurements that are very similar to the template would also be considered as anomalies because of the unlikelihood of this scenario, cf. Fig. 2.

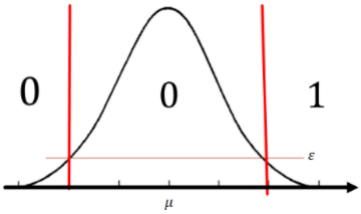

Fig. 2: Detection criteria along the probability density curve

The output of the anomaly detector algorithm is then represented in equation (4), where ε represents this so-called threshold.

$$y(x) = \begin{cases} 0 : \text{if } f(x) \geq \epsilon \\ 1 : \text{if } f(x) < \varepsilon \text{ and } f(x) > \mu \end{cases} \tag{4}$$

The mean and the variance are a result of fitting the training data to a Gaussian distribution. As no feedback is given to the algorithm at the end of this process this is an unsupervised methodology. A Gaussian distribution is assumed a-priori based on the assumption that there will be enough data for such a distribution to fit to the data and the resulting model is not compared to any other information.

However, this model will not be unique, especially if the number of samples used to fit the model decreases. Different selections of training data will result in different models or, in other words, different means and variances. This adds to the issue of setting a coherent value for the anomaly threshold, as the performance of the algorithm will completely depend on the three parameters (mean, variance, and threshold) to be optimally tuned for the specific application they are to be used in.

Although there is no simple way to predict the behavior of the data, there are means to assess whether a given sample is anomalous or not. This can be used to label the training data not only to assess the performance of the system, but also to feed this information back to the system so that the threshold can be set to optimize the performance of the anomaly detector given a probability density function for the data. As labeled data is fed back to the system so that its performance can be tuned (as opposed with the probability density function), the determination of the threshold is performed in a supervised manner.

The reason for choosing this hybrid approach instead of a purely supervised one is that the amount of positives (anomalies) is expected to be minimal with respect to the amount of negatives within all the data collected. The nature of anomalies is not easy to predict as anything that causes a significant deviation from the template will result in an anomaly. Because of this, a typical supervised approach, like a neural network, would face trouble identifying the “anomalous” category.

3 Matlab Implementation of the Algorithm

Data collected from the sensor is presented to the algorithm in Matlab as a collection of csv-files that contain the information from all 7 data sources (3 dimensions from the gyroscope, 3 from the accelerometer, and the timestamp). This information must be extracted into Matlab structures so that it can be used in the algorithm.

Once the data is available in a workspace structure, it must be divided into a training set and an evaluation set. A template is selected from the training set to be the standard against which all measurements are going to be compared. All the training data points are then compared to the template using DTW to obtain similarity values based on distance to the template. For this Matlab provides the command *dtw(X,Y)* which will output the distance between X and Y. The function will also be useful when classifying test data and new incoming data.

For fitting the model the Matlab command *fitdist(X,DISTNAME)* is leveraged. This function will return a variable containing information on the probability distribution of X when fitted to the type of distribution DISTNAME, including the mean and variances, which are needed to build the anomaly detector. A normalization of the data (subtracting the mean and division by the standard deviation of the dataset) supports fitting to a Gaussian model and makes data comparable when it comes from different sources and happens to be of different scale or range.

The search for the last parameter, the threshold, is done by searching for the value that maximizes performance of the model based on mean and variance obtained from *fdist*. To do this, the likelihood of a given value according to the obtained model must be calculated. This can be obtained directly from equation 3 or by using the expression *Y = normpdf(X, MU, SIGMA)*. This is the same expression that will allow for the implementation of the anomaly detector when evaluating test data or new incoming data after the '*dtw*' function has been used on it.

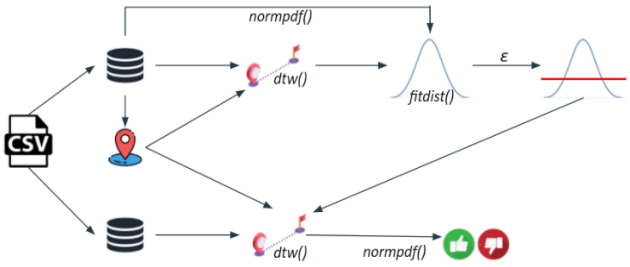

Fig. 3: Graphical overview of the algorithm and the main Matlab functions involved in it

4 Performance Criteria

Model evaluation is an integral part of the model development process. Evaluating model performance with the data used for training is not acceptable because it can easily generate overoptimistic and overfitted models. To avoid overfitting, a test set or collection of takes (not seen by the model) is used to evaluate model performance.

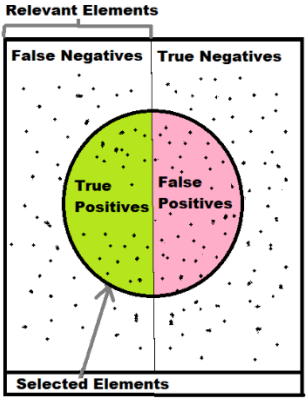

Fig. 4: Possible Outcomes of Anomaly Detection

In order to test the performance of the model, precision, recall and F1 were taken into account. Within the context of this paper, these concepts are defined as follows using the possible outcomes of anomaly detection seen in Fig. 4:
Precision refers to the number of correct classifications among all identified anomalies.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{1}$$

Recall illustrates the percentage of all anomalies that the model found.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} * 100\% \tag{2}$$

F1 represents the harmonic mean of precision and recall and is typically used when both precision and recall matter, which is typically the case [Mo20].

$$F1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{3}$$

5 Movements of the Collaborative Robot (Cobot)

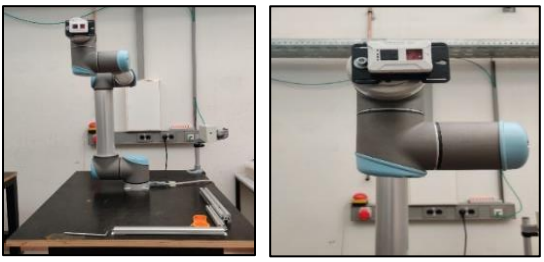

Fig. 5: UR Cobot (left) and XDK Sensor Placement (right)

5.1 Sensor placement, Robot Trajectory and Test Cases

The Bosch XDK Sensor was attached to the Cobot Tool Center Point (TCP) with the orientation as shown in Fig. 5. The robot followed a trajectory consisting of cartesian paths in 3 different planes, covering the workspace evenly to capture motion data from both gyroscope and accelerometer. 88 normal datasets and 16 anomalous datasets were obtained. Since the industrial robots are highly consistent in movement, in most cases, no anomalies would normally be available. Hence, to obtain anomalies some disturbances were introduced by human-robot interactions such as blocking the robot's path, bringing it to (safe) emergency stops, restarting the trajectory at random points by remote control, changing the sensor's orientation, pausing the robot's trajectory, and hitting (safely) the robot. This corresponds with industrial safety practices of Cobots which typically have human interactions.

5.2 Testing regime and observed results

Changed Parameters	Recall	Precision	F1
Training pool size	0.9091	1	0.9524
Gyroscope data only	0.7	1	0.8235
Accelerometer data only	0.6	1	0.75
Without Normalization	1	0.137	0.241
Using filter	0.6	1	0.75
Excluding timestamp	0.6	1	0.75
Increase anomalies ratio	0.6	1	0.75

Tab. 1: Overview of test results

The following test regime was carried out leading to the results in Tab. 1:

1. All data included: This means that all 88 datasets as well as 16 anomalies were evaluated under a single run of the algorithm. After this test, the datasets were

divided into a training pool (to train the algorithm) and a test pool (to test the trained algorithm).

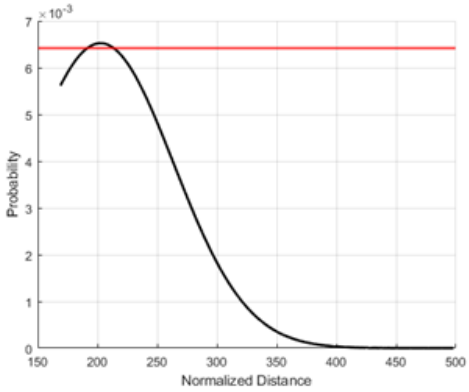

Fig. 6: Probability density graph based on normalized distances of experimental data

2. Testing only accelerometer data: This means that the first 3 dimensions, as well as their timestamp (dimension 7 of any dataset), were used.

3. Testing only gyroscope data: This means, the remaining dimensions, as well as the timestamp of the dataset were used. This improved model performance.

4. Changing training pool size: The effects of different training pool sizes were assessed by adding 6 datasets to the training pool in contrast to other parts of the regime. The best F1 is obtained by increasing the training pool size.

5. Excluding filter or normalization: During data pre-processing, a median filter was used to eliminate extreme outliers and data normalization was used to make data fit into the same normal range. The F1 did not change by turning off the filter and the worst F1 was obtained if the data is not pre-processed (normalized).

6. Increase anomalies ratio: Adding more anomalies in the training pool resulted in identifying more anomalies than before, as the algorithm becomes more robust to recognizing anomalies recorded during this research for the tests.

7. Shuffling Datasets: A pseudo-randomizer function which shuffles the data sets used for training and testing and is then fed to the algorithm. Shuffling the datasets leads to a better F1 score and is useful in identifying more number of anomalies. Among the 105 datasets obtained from Bosch XDK Sensor, training was run on 40 datasets (training dataset), and the model was tested against the other 65 datasets (called the validation dataset or testing set). The goal of cross-validation was to test the model's ability to predict anomaly in new data that was not used in estimating it, in order to flag problems like overfitting or selection bias.

The best and the worst F1 are 0.9524 and 0.241 respectively and this shows that the algorithm is heavily affected by the training pool size and data pre-processing. Accelerometer data seems to be noisier than the gyroscope's and hence provides poorer results by only using it as compared to only analysing gyroscope results. However, increasing the anomalies ratio within the training pool is effective in identifying all the anomalies, if those are labelled correctly as anomalies. Therefore, labelling is important, hence the mentioned combination of supervised and unsupervised processing is helpful.

6 Adaptability of DTW applied to the developed algorithm

6.1 Dependent and Independent DTW

As the data coming from the sensor is constituted of 7 different dimensions, it is of particular interest to look into how the DTW algorithm is taking this information into account and how it affects the performance of this application. The built-in function of Matlab takes a data series in, regardless of its number of dimensions, and proceeds to calculate the distance between both timeseries in what is described by [Sh17] as a "dependent" approach to the DTW. However, in this same work, it is described how not necessarily all dimensions of a given application are directly dependent with each other and how this affects the result of the anomaly detection.

The next step of this study was then directed into precisely this: implementing different methodologies for the DTW calculation and evaluating how the result is influenced by this. Results for a completely dependent approach were already obtained from the previous phase and two further variations were proposed:

- A completely independent calculation in which each dimension is compared only with the corresponding timeseries in the template and the individual results of all dimensions are then summed together to obtain a single value.
- A clustered independent calculation where data coming from the accelerometer is treated as a 3D timeseries and then compared in a dependent manner with the corresponding timeseries of the template. The result from this calculation is summed up with the result of the gyroscope´s data which is treated in the same way and with the result of the timestamp which is purely one-dimensional.

For this phase, the same implementation was used, except that the calculation of the distances through DTW was expanded to include the new approaches.

For the evaluation of all the approaches the size of the training and testing data sets was varied from 20 to 75 with step sizes of 5 to find an optimum size for each case. Tab 7.1 shows the performance of the algorithm while varying the training pool size for each of the proposed approaches of the DTW:

Pool Size	Dependent F1 (Test)	Dependent F1 (Training)	Clustered F1 (Test)	Clustered F1 (Training)	Independent F1 (Test)	Independent F1 (Training)
20	1	0.8	1	0.7368	1	0.7368
25	1	0.8	1	0.8571	1	0.8
30	1	0.8	1	0.8571	1	0.8571
35	1	0.9565	1	0.8571	1	0.9091
40	1	0.9565	1	0.8571	1	0.9565
45	1	0.9565	1	0.9565	1	0.9565
50	1	1	1	0.9565	1	0.9565
55	1	1	1	0.9565	1	0.9565
60	1	1	1	0.9565	1	1
65	1	1	1	0.9565	1	1
70	1	1	1	0.9565	1	1
75	1	1	1	0.9565	1	1

Tab. 2: Performance for different DTW approaches in training and test environments

The performance achieved in both training and in testing for all three approaches is outstanding. The difference lies in the training performance of the different approaches. We can notice that the dependent approach reaches an ideal performance with a training pool size of 50, while the independent does so with 60 and the clustered one never reaches the ideal training performance, although it still obtains an F1 of more than 95% with 45 samples. As the accelerometer and the gyroscope are attached together and their measurements are the result of the same movements it is not surprising that this application tends to behave better under a dependent approach. This would be the recommended implementation for a real-life scenario due to its simple implementation and desired results, while mentioning that any of the three approaches would yield an excellent performance.

Conclusions

Anomaly detection based on the automatic determination of the parameters of the probability density function (including a threshold) of the distance of a certain timeseries to a given “ideal” sequence proved to be a very suitable way of detecting deviances in the movement of a robotic arm. The achieved system performance exceeded our initial expectations; upon increasing the number of anomalies presented to the algorithm during training the final performance generally tends to increase.

DTW provided a versatile and handy concept that is useful to compare timeseries that may differ in length and may lack synchrony between them without losing information and retaining a real measure of similarity, without a high computational complexity. Different approaches for its implementation, such as clustering input dimensions, can provide advantages on particular applications. However, for our Cobot scenario, either of the

approaches showed a sufficiently good performance.

Acknowledgements

We would like to thank Prof. Dr.-Ing. Karl Kleinmann who provided guidance throughout this research and Mr. Timm Streul, Ms. Rashmi Rajanna who supported this work during its initial phases.

Bibliography

[BJ15] Blomquist, H.; Möller,J.: Anomaly Detection with Machine Learning. Quality Assurance of Statistical data in the Aid community, Uppsala Universiteit. 07/2015.

[Bo20] Bosch, YouTube Channel, https://www.youtube.com/watch?v=FAlyjBO0-7g&ab_channel = Bosch Connected Devices and Solutions, accessed: 05/2020.

[CBK09] Chandola, Varun; Banerjee, Arindam; Kumar, Vipin: 2009. Anomaly detection: A survey. ACM Comput. Surv. 41, 3, Article 15 (July 2009), 58 pages.

[Cr20] Cross Domain Development Kit XDK, https://www.Bosch-connectivity.Com/products /cross-domain/cross-domain-developement-kit/, accessed: 05/ 2020

[Co17] Costa B.G. et al. (2017) Fault Classification on Transmission Lines Using KNN-DTW. In: Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA 2017.- ICCSA 2017. Lecture Notes in Computer Science, vol 10404. pp.174-187 Springer, Cham.

[ESL19] Eiband, T.; Saveriano, M.; Lee, D.: Intuitive Programming of Conditional Tasks by Demonstration of Multiple Solutions, (IEEE): IEEE Robotics and Automation Letters, vol. 4, no. 4, pp. 4483-4490, Oct. 2019

[Ho14] Hornung, R. et al.: Model-free robot anomaly detection, (IEEE): 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3676-3683, 2014

[Ma20] Machine Learning, https://www.coursera.org/learn/machine-learning, accessed: 03/2020.

[Ma19] Mahmoud, D. et al.: Anomaly Detection Using Dynamic Time Warping, (IEEE): 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), pp. 193-198, 2019

[Mo20] Model Evaluation, https://www.saedsayad.com/model_evaluation.htm, accessed: 20/06 /2020.

[RU20] RUL Estimation Using RUL Estimator Models, https://de.mathworks.com/help/predmaint/ug/rul-estimation-using-rul-estimator-models.html, accessed: 08/2020.

[Sh17] Shokoohi-Yekta; et al.: Generalizing DTW to the multi-dimensional case requires an adaptive approach. Data Mining and Knowledge Discovery 31, pp. 1–31 (2017).