Anomaly Detection Algorithm

Table of contents

Introduction

This project was created by Šimon Rozsíval as part of the Seminar on Artificial Intelligence II. [1] at the Faculty of Mathematics and Physics of the Charles University in Prague under the supervision of prof. Roman Barták.

The goal of this project was to create an online anomaly detector for a flying drone. The drone produces a stream of sensor readings and sends them over to a host computer. The data should be analyzed and each time some suspicious reading from the sensors is received the controller of the drone must be informed to respond accordingly.

Anomaly Detector

The input of the detector is a stream of sensor readings from the device with a timestamp. One reading can be an arbitrary vector of real values. The length of the vector is not limited in any way, it only has to be the same for all of the readings.

The expected output are annotations of the input data. For each sensor reading it is decided whether it is considered anomalous or not.

Unsupervised Learning Algorithm

The Online Anomaly Detection in Unmanned Vehicles paper [2] was used as the main source for implementing of the algorithm. This paper describes an algorithm which uses unsupervised learning to detect the anomalies among data from sensors of an Unmanned Aerial Vehicle (UAV).
The main specifics of the algorithm are the use of changes of the sensor readings instead of their absolute values and the use of the Mahalanobis distance metric [3] between the mean of the distribution of nominal points and the new sensor reading vector in the units of standard deviations of the distribution. When a point is too far from the set of nominal data, then it is considered anomalous.The unsupervised learning consists of determining the thresholds of distances below which a point is still valid and above which it is already considered to be too far and labeled as an anomaly.

The Implementation

The algorithm was implemented using Python 3 and the NumPy [4] and scikit-learn[5] libraries and it is available to the public in a git repository [6]. The source code covers all of the functionality described in the paper [2] with the only exception of the *Z* filter. There are two parameters which can be modified: the size of the sliding window and the correlation threshold (`ct`) parameter.

The size of the window has effect on how many nominal points are used for the online training. The value should correspond to the frequency of the incoming data points so the sliding window is not too large but it contains enough data to avoid false positives with noisy data during movement of the drone.

The value of the correlation coefficient was selected experimentally to minimize the number of false positives while detecting all significant anomalies according to the measurements done in the paper and our own experiments. A better fine tuning of this parameter could lead to more accurate results.

Experiments and Results

We have conducted several experiments on BeBop and Parrot AR.Drone with an external camera attached to it. The recorded data are available on the website of the seminar [1].

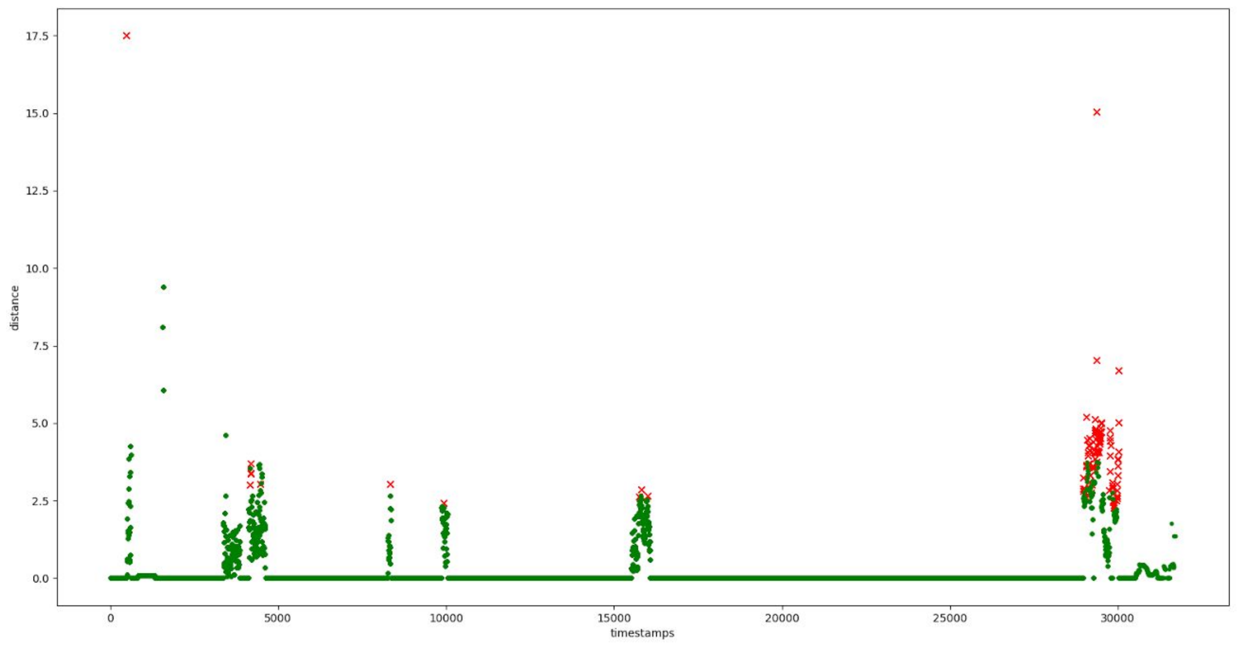
The detector produces a graph for the test data which contains the mahalanobis distance of the next sensor reading from the nominal points in the sliding window. The *x* axis corresponds to the number of elapsed milliseconds from the start of recording, the *y* axis then denotes the distance.

The experiments

For the experiments, a sliding window of size 100 was used and the correlation threshold parameter was set to 0.9.

Test “camera_auto01”

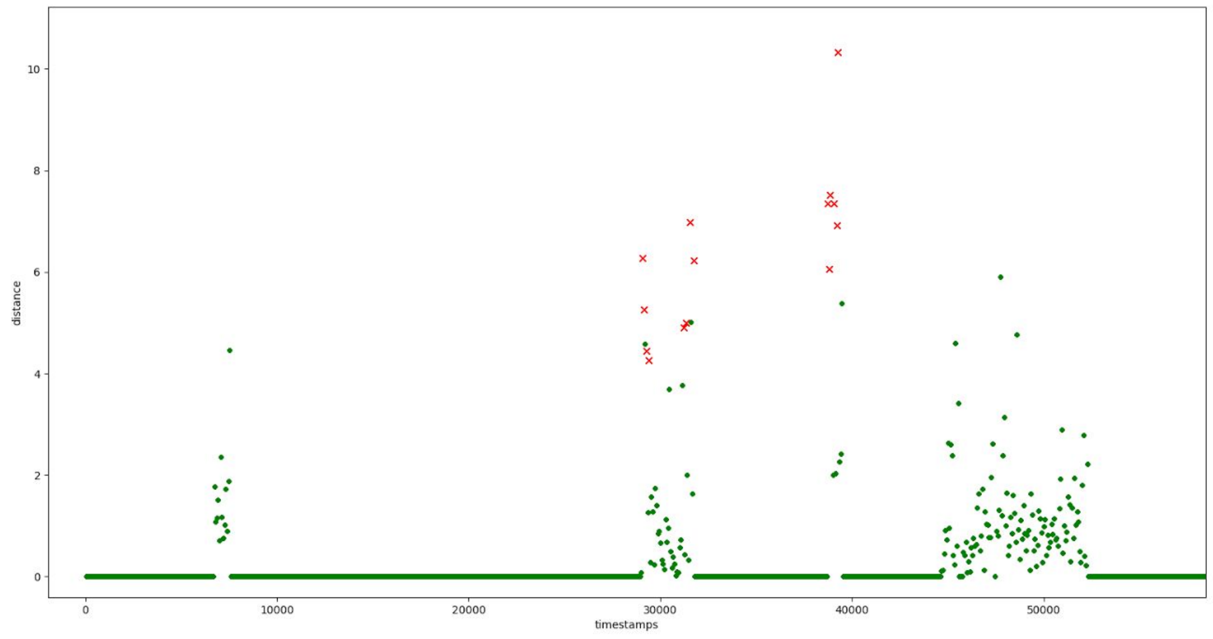
This ~30 seconds long test consists of a take off (~4 s), hovering (~4 s), flight forward (~5 s), coming to full stop (~3 s), flying backwards (~12 s) and then crash landing (~3 s). The video is offset from the data by 10 seconds.



The transitions between the parts of the flight are clearly visible in the graph - they correspond to the red points in the graph.

Test “dataBeBop”

The data from the BeBop drone come only from the camera feed. This video was processed using optical flow and each of the frames is annotated with 18 attributes which somehow describe the motion of the drone. The anomaly detector does not need to know anything about the nature of the data stream so it can process this dataset without any internal changes. The frame rate of the video is 15 FPS which is very easy for the anomaly detector to process and unlike with the AR.Drone, the anomaly detection could be run in real time on this data.

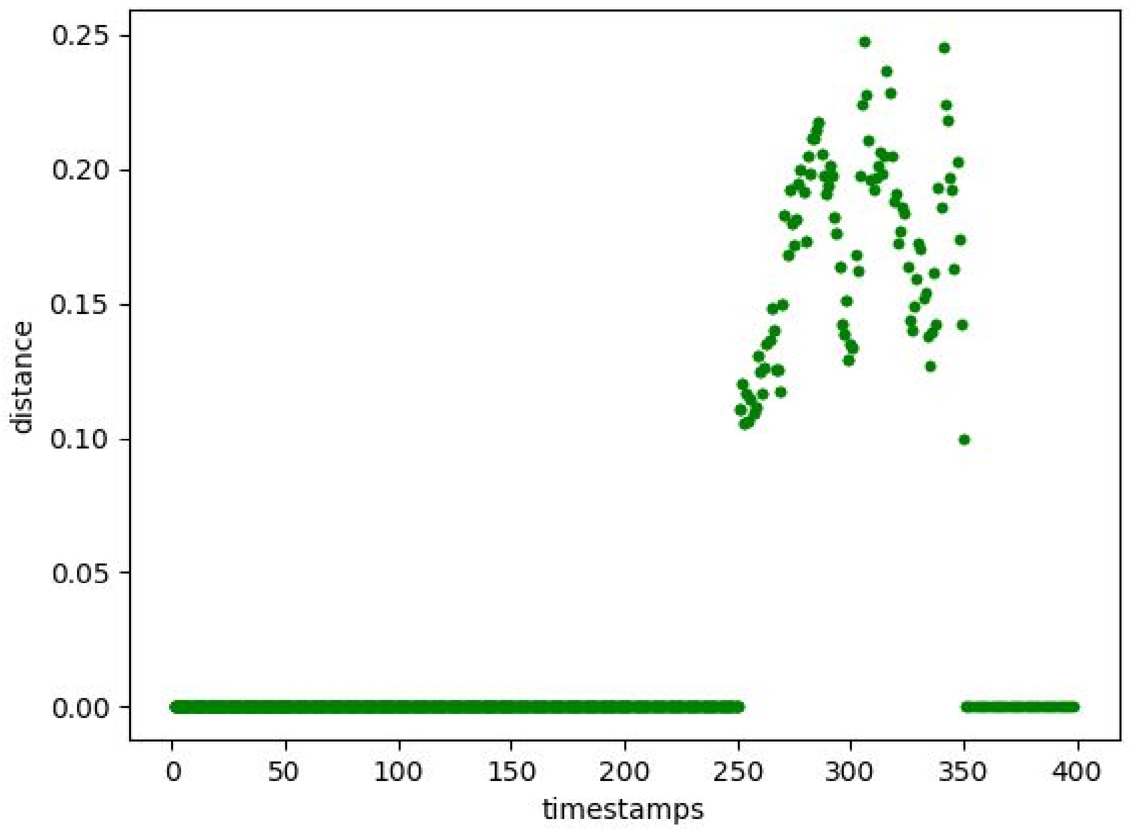


The algorithm has detected some anomalies around 30 s and 40 s of the video. The drone does fast rotations at these times. The reason for the detection of anomalies might be either the quickly changing acceleration or the quality of the image processing because of the combination of poor lighting conditions and the swift movement of the camera.

Test “mock_faulty_sensor”

This test was generated by creating a random point and then adding a new random value from the normal distribution to the point. After iteration 250 the system simulates a failure of the sensor and all readings since then contain only zeros.

For this test, the value of the correlation threshold constant was changed to 0.65.



This experiment covers a different type of anomaly that the previous two tests. The system can detect some suspicious points, but the distances are not large enough to be declared as anomalies.

This anomaly was not detected by the detector even for different values of the parameters. Setting the value of the correlation threshold to an even lower value led on the other hand to false positives.

Conclusion

The implementation of the anomaly detector was applied on the data we measured using the AR drone and the detector has marked the points with suspicious changes in the data well.

Not all of these changes were real anomalies. The detector had no information about the commands sent to the drone and did not therefore anticipate for example significant changes in acceleration when the controller of the drone wanted to suddenly stop while the drone was flying at full speed. To create a fully functional anomaly detector, another layer which would ignore the “anomalies” when the controller sends a different command to the drone should be implemented on top of this algorithm.

Another drawback of the developed algorithm is its performance. The algorithm cannot process the incoming data at high frequencies (~100 Hz). One solution might be to preprocess the data either by processing every *n*-th sensoric reading and ignoring the rest of the readings or by combining several readings into one record and then passing them to the anomaly detector. The size of the sliding window could be also decreased to fasten the creation of the covariance matrix - this would probably lead to an increase of the number of false positives though as even small changes in the data may seem different from the previous points.

A different solution would be to optimize the algorithm itself. The most time-consuming operations are calculations of the covariance matrix and its inverse matrix on the data in a sliding window. These matrices could be recalculated every time a new item is added and removed from the sliding window and thus avoid calculating inversion of a large matrix.

Even though there are some improvements to be made, the algorithm [2] itself proved to be useful and suitable for anomaly detection for AR drone and any similar device with an online stream of sensor readings without any previous training of the model and the implementation itself to work properly for anomaly detection though not as an online anomaly detection system.

References

[1] Seminar on Artificial Intelligence II, Seminar LS 2016/2017 (NAIL052), available online: http://ktiml.mff.cuni.cz/~bartak/ui_seminar/index.html. Reviewed on 5/28/2017, reviewed on 5/29/2017

[2] Online Anomaly Detection in Unmanned Vehicles, Eliahu Khalastchi, Gal A. Kaminka, Meir Kalech and Raz Lin, Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011), Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 115-12, Available online: http://u.cs.biu.ac.il/~linraz/Papers/elietal-AAMAS11.pdf (PDF). Reviewed on 5/28/2017

[3] On the generalised distance in statistics, Mahalanobis, Prasanta Chandra (1936). Available online: http://insa.nic.in/writereaddata/UpLoadedFiles/PINSA/Vol02_1936_1_Art05.pdf (PDF). Proceedings of the National Institute of Sciences of India. 2 (1): 49–55. Retrieved 2016-09-27.

[4] NumPy library, available online: https://www.numpy.org, reviewed 5/28/2017

[5] scikit-learn library, available online: http://scikit-learn.org/, reviewed on 5/28/2017

[6] The source code at Github.com, Šimon Rozsíval, available online: https://github.com/simonrozsival/mff-ai-anomaly-detector, reviewed on 5/28/2017