Efficient Hybrid Fault Detection for Autonomous Robots

Extended Abstract

Eliahu Khalastchi
College of Management Academic Studies
Rishon LeZion, Israel
eliahukh@colman.ac.il

Meir Kalech
Ben-Gurion University of the Negev
Beer-Sheva, SRAEL
kalech@bgu.ac.il

ABSTRACT

The use of robots has increased significantly in the recent years; rapidly expending to numerous applications. Yet, these sophisticated and sometimes expensive machines are susceptible to faults that might endanger the robot or its surroundings (e.g., a crash of an Unmanned Aerial Vehicle (UAV)). To prevent such faults, the robot's operation needs to be monitored by Fault Detection (FD) algorithms. An autonomous robot, which is already engaged with heavy computational tasks, has to continuously apply FD on its own. Thus, the impact of a FD process on the robot's resources should be minimized. Unfortunately, the computational load of existing FD approaches, which may be very accurate, might be impractical for an autonomous robot. To solve this problem, we suggest to use a hybrid approach. A very efficient FD algorithm is applied continuously and is used to trigger a heavier, more accurate, FD approach that determines the occurrence of a fault. In this paper we focus on the efficient FD algorithm. We test the algorithm in several real-world and simulated domains and we show and discuss the promising results.

KEYWORDS

[ROB] Failure recovery for robots; [ROB] Long-term (or lifelong) autonomy for robotic systems

ACM Reference Format:
Eliahu Khalastchi and Meir Kalech. 2020. Efficient Hybrid Fault Detection for Autonomous Robots. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 3 pages.

1 A SIMPLE AND EFFICIENT FAULT DETECTOR - SEFD

We present a Hybrid FD approach which consists of two parts: (1) a Simple, very Efficient FD algorithm (SEFD), and (2) any other FD algorithm that is more accurate but might be too heavy to continuously run on the robot. SEFD has a very low computational impact and thus can continuously run in the background without interfering the other computational tasks of the robot. Upon the detection of a suspected fault SEFD triggers the other FD algorithm to determine whether indeed a fault has occurred.

The SEFD algorithm relies on correlated features for fault detection. The domain of robots is reach with sensors that are redundant or affected in the same way by the robot's behavior. For instance, consider an Unmanned Aerial Vehicle (UAV). The UAV's pitch indicator is correlated to the UAV's elevators and to the vertical speed sensor; all these features cause or respond to the change in the UAV's altitude. Intuitively, SEFD checks if features that are supposed to be correlated display an uncorrelated behavior that was not previously observed in past normal operations.

Thus, SEFD consists of 3 parts: (1) correlation detection, (2) thresholds learning, and (3) online fault detection. Parts 1 and 2 are done offline as a pre-processing phase, and part 3 is an online algorithm.

Algorithm 2 depicts the online detection process. Given D we learned offline, and the current values V_t we apply the same algorithm with the exception that if $z(d) > d_i$ and d is greater than the max value in D (line 5) then alert there is a possible fault associated with f_i and f_{iC}. Recall that as a result another FD algorithm, which is potentially more accurate but heavier on resources, is triggered to determine the occurrence of the fault. Thus, we aim to significantly reduce the computational load while still having high fault detection accuracy.

Note that we need to check that $d_i > max(D)$ to prevent a false alarm for a case where the point (x,y) suddenly got closer to the 45°line. This case indicates an increase of correlation which is not a symptom of a fault, yet it may cause a high z-score value, as discussed above.

If a point (x,y) slowly drifts from the line then the corresponding values of d remain similar and, in turn, the z-scores of d will remain low and an alarm will not be raised. However, a rapid drift of (x,y) form the 45°line leads d to be different than its previous values. As a result the z-scores of d may increase above the acceptable threshold d_i (learned from normal operations) and an alarm will be raised.

Algorithm 2: Online Fault Detection of SEFD

Input:
$V_t = \{v_{t,1}, v_{t,2}, \ldots, v_{t,n}\}$ – values of features 1..n at time-step t.
Output:
Alerts of possible detected faults

1. **For Each** feature f_i
2. **IF** f_{iC} exists **THEN**
3. $(x, y) \leftarrow ((norm(v_{i,t}), norm(v_{iC,t}))$
4. $d \leftarrow |y - x|$
5. **IF** $z(d) > d_i$ **AND** $d_i > max(D)$ **THEN**
6. Alert "possible fault associated with f_i, f_{iC} "

The time complexity of the above online part of SEFD is $O(n)$ where n is the number of features.

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

2 EXPERIMENTAL SETUP AND RESULTS

The Competing Approaches are the following. Incremental LOF [5] – a very popular density based outlier detection algorithm, which is able to detect local outliers by utilizing the K-Nearest-Neighbor algorithm. In our experiments, fault symptoms can be regarded as local outliers.

ODDAD [2] – uses a sliding window to detect correlations online and utilizes the Mahalanobis Distance (Mahalanobis, 1936) to detect faults. Mahalanobis Distance is actually a z-score function for multi-dimensions. As such, it requires to invert the variance-covariance matrix of the selected features. This increases the time of calculations. In addition, not every resulting matrix is invertible. As a result this method will be insensitive to some faults.

SFDD [3][1] – uses a sliding window to detect correlations online and a heuristic fault detection function that sates that if two correlated features display different behaviors (e.g., one drifts while the other is stuck) then it alerts a fault.

The Domains of our experiments consists of one simulated domain (for comprehensive testing) and 3 real-world domains: a physical robot, a Boeing 737 aircraft and a helicopter.

The first domain is the flight simulator FlightGear [4]. FlightGear is an open source high fidelity flight simulator designed for research purpose and is used for a variety of research topics.The injected faults affect the behavior of the aircraft [4].

We sampled 23 features in a frequency of 4Hz. These features present 5 flight controls (actuators), and 18 attributes of flight instruments (sensors). The data set contains one flight which is free from faults and 5 subsets that each contains 12 recorded flights in which different faults were injected. In total, the data set contains 62 recorded flights with almost 90,000 data instances (the flights duration is 6 minutes). We injected faults to 7 different instruments and to 4 different subsystems. These faults led their corresponding features to be stack on the same value or to slowly drift. Table 1 depicts the results for the competing fault detection algorithms for the FlightGear domain.

Algorithm	TPR	FPR
SEFD	0.90	0.0042
Incremental LOF	0.87	0.075
ODDAD	0.88	0.13
SFDD	0.98	0.06

Table 1: Results for the FlightGear domain.

Our algorithm (appears first) is SEFD. SEFD achieved a higher TPR (True Positive Rate) and significantly lower FPR (False positive Rate) than LOF and ODDAD. The SFDD got the highest TPR, yet with at the cost of almost 15 times more false positives compared to our algorithm. The very low FPR can be explained by the way our algorithm chooses the threshold – the maximum z-score observed in the normal operation. Since the normal operation contains similar noise to the faulty operations, then the threshold is set above the noise – which, in turn, lowers false alarms. In addition for being lighter on computational resources, the SEFD proved to be more accurate and thus it is more suited for being the trigger to a heavier FD algorithm.

Robotican1 is a robot that has 2 wheels, 3 sonar range detectors in the front, and 3 infrared range detectors which are located above the sonars. This redundancy reflects real world domains such as unmanned ground vehicles. In addition, Robotican1 has 5 degrees of freedom arm. Each joint has two electrical motors which provide a feedback value.

We applied the following scenario 11 times: the robot slows its movement as it approaches an object. Concurrently, the robot's arm is adjusted to grasp the object. We injected faults of type stuck or drift to different type of sensors (motor voltage, infrared and sonar) in 10 operations. We sampled 15 sensors in 8Hz. Scenarios duration lasted only 10 seconds where faults lasted only 1.25 seconds. In total, the test set contains 800 instances out of which 90 are expression of faults.

Algorithm	TPR	FPR
SEFD	0.80	0.03
Incremental LOF	0.70	0.08
ODDAD	1	0.017
SFDD	0.56	0.067

Table 2: Results for the Robotican1 domain.

Table 2 depicts the results for the competing fault detection algorithms for the Robotican1 domain. In this domain ODDAD produced the best results, and SEFD came second in both TPR and FPR. This can be explained by the short duration of the faults (1.25s, 9-10 instances long).

Algorithm	Boeing 737	Apache Helicopter
SEFD	0.32sec	6.7sec
SFDD	20.7sec	19.6sec

Table 3: Time measurements for high-dimensional domains.

Some robots have very complex machinery and provide high-dimensional data where scalability becomes an issue. The FlightGear and Robotican1 domains do not offer such a challenge due to their small number of features and short time of operations. The **Boeing 737** domain is a dataset comprised of two flights: a normal flight, and an identical flight that the manufacture injected a fault to one of the sensors (altitude stack). 63 features were sampled at 8Hz for a duration of over 2 hour flight. In total, 7,980 data instances per flight. The domain is relevant for robotics due to its sensor redundancy and due to the autopilot. The **Apache Helicopter** domain is a dataset comprised of a normal flight, and a flight that ended with an unfortunate crash of the vehicle. 267 features were sampled at 10Hz for a duration of over 1.5 hour flight. In total, 58,080 data instances per flight. The large amount of features requires a fault detection approach to be very scalable.

The relevant competing approach for the scalability test is the SFDD. The disadvantage (w.r.t time) is the online correlation detection but the fault detection is based on a simple heuristic. We measured the time it took the SEFD and SFDD to compete the task of fault detection on these two domains (under the same conditions). Table 3 depicts the results.

REFERENCES

[1] Eliahu Khalastchi and Meir Kalech. 2018. A sensor-based approach for fault detection and diagnosis for robotic systems. *Autonomous Robots* 42, 6 (2018), 1231–1248.

[2] Eliahu Khalastchi, Meir Kalech, Gal A Kaminka, and Raz Lin. 2015. Online data-driven anomaly detection in autonomous robots. *Knowledge and Information Systems* 43, 3 (2015), 657–688.

[3] Eliahu Khalastchi, Meir Kalech, and Lior Rokach. 2013. Sensor fault detection and diagnosis for autonomous systems. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 15–22.

[4] Alexander R Perry. 2004. The flightgear flight simulator. In *Proceedings of the USENIX Annual Technical Conference*.

[5] Dragoljub Pokrajac, Aleksandar Lazarevic, and Longin Jan Latecki. 2007. Incremental local outlier detection for data streams. In *2007 IEEE symposium on computational intelligence and data mining*. IEEE, 504–515.