

A Non-intrusive Fault Diagnosis System for Robotic Platforms

Youssef Mahmoud Youssef¹ and Paul G. Plöger²

¹ ²Hochschule Bonn-Rhein-Sieg

e-mail: youssef.youssef@smail.inf.h-brs.de

e-mail: paul.ploeger@h-brs.de

Abstract

The increasing complexity of tasks that are required to be executed by robots demands higher reliability of robotic platforms. For this, it is crucial for robot developers to consider fault diagnosis. In this study, a general non-intrusive fault diagnosis system for robotic platforms is proposed. A mini-PC is non-intrusively attached to a robot that is used to detect and diagnose faults. The health data and diagnosis produced by the mini-PC is then standardized and transmitted to a remote-PC. A storage device is also attached to the mini-PC for data logging of health data in case of loss of communication with the remote-PC. In this study, a hybrid fault diagnosis method is compared to consistency-based diagnosis (CBD), and CBD is selected to be deployed on the system. The proposed system is modular and can be deployed on different robotic platforms with minimum setup.

1 Introduction

In the fast developing world of today, robots are being designed and manufactured to aid humans in tasks which are repetitive, dangerous or tedious. One of the main fields of development for robotics is Unmanned Ground Vehicles (UGVs). UGVs are fast growing and are of high interest to both the commercial and defence industries because of their practicality in different use-cases. Scenarios such as search and rescue, explosive ordnance disposal, mine clearing, or perimeter monitoring are attractive for using technologies such as UGVs. Some of the reasons why this technology is continuing to grow is because of the increasing availability of low-cost sensors and microcontrollers, increased labour cost due to globalization, change of the acceptability of human casualties and loss of life and many more reasons [1]. Recent studies have shown that the UGV market is projected to grow from USD 1.49 Billion in 2016 to USD 2.63 Billion by 2021 ¹.

An interesting aspect of the UGV market survey is that based on mode of operation, teleoperated robots lead the market among tethered, semi-autonomous, and autonomous robots. This is because of the unreliability of autonomous

robots, and the nature of the dynamic environments that humans want to deploy robots in. Unreliability is an aspect of being autonomous; since there is no human intervention in truly autonomous robots, the internal communications and decision-making processes of the robot are not known to the user. Due to these reasons and the unreliability which comes from the robot being autonomous, one can expect faults to occur frequently.

It is therefore crucial for robot designers and developers to consider faults during their design process, either in building systems that are fault tolerant, or in creating systems that have fault diagnosis capabilities. In the robotic domain, fault tolerance can be achieved by adding redundant sensors, actuators, or even redundant software components that make sure some computations are always executed. Fault tolerance is usually required in systems which are safety-critical or systems that require low downtime. Moreover, adding redundant components increases the total cost of the system, or adds greater computational complexity to the system.

Another aspect that comes to mind regarding designing systems that are fault tolerant or have fault diagnosis capabilities, is the tracking of the status of components, and storage of important information in order for the users or experts to be able to assess and analyse the status of a system over time when required. One of the most notable systems that do so are aircrafts. Aircrafts have dedicated data recorders, famously known as black boxes, which store important information and data logs of multiple signals coming from different components. Over the years, flight data recorders (FDRs) have proven to be crucial, in analysing the status of aircrafts after accidents or faults have occurred.

Data representation and standardisation is also an issue that needs to be considered while developing robotic systems. Since robotic platforms nowadays are highly customizable, and a range of different components are available to a developer, one can expect such complex systems to exchange data in both high volume and rate. It therefore makes sense to have a generalized method to represent the health of different components in a way that is thorough enough to give the user or developer an overview of the robot's health, and be able to cope with the different components running on the robot.

The goals of this study is to develop a fault diagnosis system that can do the following:

- Design a fault diagnosis system that can be deployed on varying robotic platforms with minimum setup.
- Non-intrusively monitor and diagnose varying robot

¹<https://www.marketsandmarkets.com/Market-Reports/unmanned-ground-vehicles-market-72041795.html>

components.

- Standardize the health messages produced by the diagnosis system.
- Transmit standardized health messages to a remote-PC.
- Log the health data produced by the diagnosis engine of the monitored components.

2 Related Work

To achieve the goals of this study, fault diagnosis methods needed to be reviewed in order to select an appropriate method. Robotic platforms nowadays have a wide range of components deployed on them and since one of the goals of the study is to design a system that can incorporate different components of a robot, the selection of the diagnosis method is crucial.

Having a wide range of components to monitor means that a substantial amount of data is to be expected to be generated by the diagnosis engine. Standardizing the representation of the health data produced by the diagnosis engine would be useful to the developers and users of robotic platforms. This would help in easing the integration of new components to be monitored in the diagnosis system, since they will be enforced to follow a data structure. Standardizing the health messages as well will help in creating independence between the fault diagnosis methods used, and the diagnosis data produced by the method.

2.1 Fault Diagnosis Methods

The selected method of diagnosis should be light weight and low in computational complexity since it is intended to be deployed on a mini-PC with limited resources.

Fault diagnosis methods can be split into mainly four categories, namely, model-based, knowledge-based, data-driven and hybrid methods. Khalastchi and Kalech [2], provide a comprehensive survey of fault diagnosis methods with regards to the robotics domain.

Model-based diagnosis (MBD) relies on creating a model of a system by either having a set of analytical equations or a set of logical formulas that describe the behaviour of the system. When an inconsistency is found between the actual output of the system and the expected output from the model, the system is considered then to be faulty. Analytical or logical equations usually describe a specific behaviour of some phenomena. Therefore they usually require prior knowledge of the system. Structural models however, describe the dependency of components to each other, which is by comparison to behavioural models, easier to derive since describing relations between components is not as complicated as describing phenomena through analytical or logical relations. Some studies have been where model-based diagnosis is applied to the robotics field. The interested reader can review the following studies; [3], [4]. Model-based diagnosis face some challenges when applied to the robotic field, some are summarized here:

- Constructing a model that represents the behaviour of components on the robots such as sensors, that generate noise and operate in continuous time,
- Presenting the dynamic nature of the environment which the robot is deployed in, or representing the robot-environment interaction.

Knowledge-based approaches are used in a way that mimics human reasoning where it can diagnose a fault based on known symptoms or based on previous knowledge. Knowledge-based diagnosis branches mainly into causal analysis [5] and expert systems [6].

Data-driven approaches are mainly classified into two main branches, statistical approaches [7], [8], [9] and machine learning approaches [10]. These methods rely on monitoring the online data produced by different components in a system to detect and diagnose faults.

Finally, hybrid approaches are using any of the aforementioned methods in a combined manner. For this study, two fault diagnosis approaches were of interest consistency-based diagnosis (CBD) [11] and sensor fault detection and diagnosis (SFDD) [12]. CBD is a model-based approach that has the advantage of being light weight and can be used to model complex behaviours of different components. SFDD is a hybrid combination between model-based and data-driven approaches and was found to be attractive because of the ability to use the available data from components on a robotic platform and the ease of modelling by only requiring a structural model. Both methods will be addressed in detail in section 4.

2.2 Data Standardization

Since robotic platforms are highly customizable with regards to their function and number of components deployed on them, it is difficult to find a data standard that is imposed on robotic platform components and accepted widely by the robotics research community. Nevertheless, this is not the case with military vehicles, which is a more well developed field, where data standards already exist for electric and electronic components representation and communication [13], [14]. Taking inspiration from military vehicles, one can take advantage of already well developed standards and apply them to the robotics field. Of course, there exists a huge difference between components deployed on vehicles and robots. However, since fault diagnosis methods which can be deployed on robots can also be deployed on vehicles, one can benefit from using standards related to fault diagnosis and health monitoring of vehicle components.

For this study, access to the NATO Generic Vehicle Architecture (NGVA) was granted. NGVA² is based on the UK Generic Vehicle Architecture [14]. It is an open architecture design approach to land platforms. GVA uses open standards to software and hardware interfaces, in order to ease upgrading or replacing electric or electronic components on a vehicle, rather than being dependant on third party hardware or software interface providers. GVA was later adopted and enhanced by European nations as NATO GVA (NGVA).

Fig. 1 gives an overview of the NGVA data infrastructure. The NGVA data model contains a collection of data modules, that represent different subsystems or components of a system. NGVA uses Data Distributed Services (DDS) [15], which is a machine-to-machine middle-ware. DDS has the advantage of providing reliable real-time information exchange between system components or subsystems. DDS works in a publish-subscribe manner, where it allows applications to publish or subscribe to topics on different computers within a network or on the same computer simultaneously.

²<https://natogva.org/>

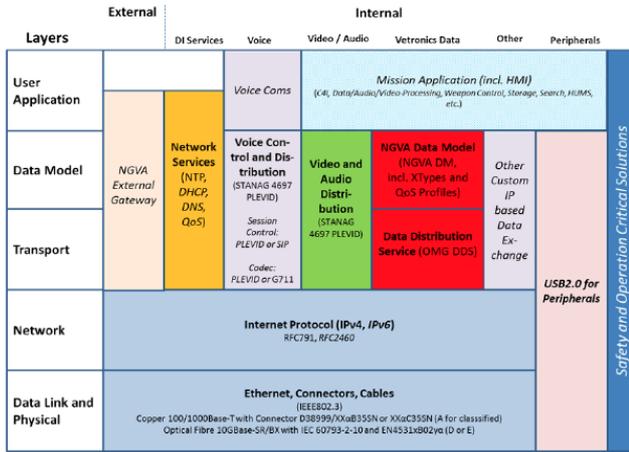


Figure 1: NGVA data infrastructure layered view [16].

Adopting such a standard to the robotics field with regards to fault diagnosis can be of great advantage, especially with regards to the goals of this research, since it provides an independent platform for communicating with a remote-PC, giving a higher robustness overall to the system, by not depending on the robot's operating system. Furthermore, since NGVA enforces a data infrastructure for the data being exchanged, one can benefit from having a standard to represent the health of components regardless of the fault diagnosis method used. From having an overview of the different data standards available from NGVA, the Usage and Condition Monitoring System (UCMS) module was selected as the most appropriate standard to represent the health of subsystems and components deployed on the robot.

3 System Architecture

To develop a system that satisfies the goals and requirements of this project, a system architecture had to be developed. Since the main goal of developing such a system is to non-intrusively monitor and diagnose faults of a robotic platform, a mini-PC was selected to be used to gather information on monitored components. The mini-PC would have its own power supply and would connect to the robotic platform via the available communication method to listen/spy to the data that can be used later on for fault diagnosis. This way, the fault diagnosis system would not burden the robotic platform with more computational tasks or use any of the robotic platform's power resources.

Since many of the robotic platforms nowadays use Robot Operating System (ROS), it was used as the main framework for this study [17]. ROS is a commonly used robotic framework that is based on a publish subscribe model. A robot running ROS contains nodes that can perform computations, and can publish or subscribe to topics. For example, a robotic platform with a camera connected would have a camera node that would publish the raw frames produced by the camera and another node can subscribe to these published frames to perform further computations.

As for the standardization of the health messages that will be produced by the diagnosis engine deployed on the robotic platform, the UCMS data model module was selected to be used and evaluated.

After considering the requirements and constraints of the system to be developed, a conceptual system architecture was developed using an attribute-driven design method and can be seen in Fig. 2.

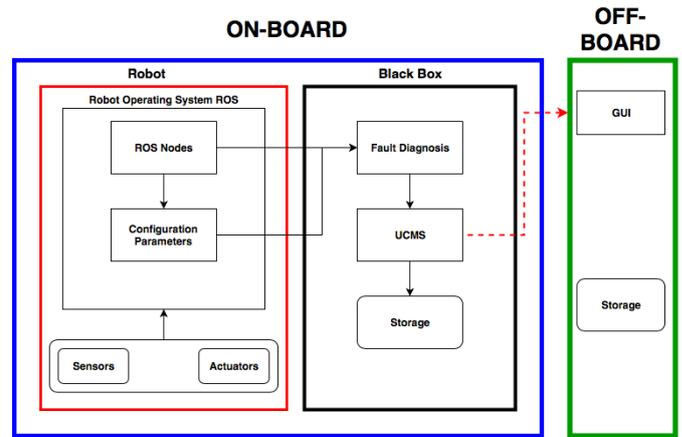


Figure 2: Conceptual system architecture of the non-intrusive black box connected to the robot.

The architecture can be divided into two main components, namely, on-board and off-board. The on-board components include the robotic platform along with the black box that contains the complete fault diagnosis system. The robotic platform has several components connected to it such as sensors and actuators. Sensors can produce information which can be used by ROS nodes and nodes can also publish commands to different actuators. For the purpose of diagnosis, configuration parameters can be derived from the different ROS nodes that represent the different components deployed on the robotic platform. The configuration parameters can help the diagnosis system by providing the necessary metadata, semantics or static information required by the fault diagnosis method.

The black box contains the fault diagnosis module which gathers data about the monitored components and produces a diagnosis based on the selected method. The produced diagnosis is then standardized using the UCMS data module and transmitted on the network using DDS as the middleware. The storage device also collects the produced diagnosis data and can be used for multiple purposes such as time analysis of component's health.

The off-board component represents the remote-PC that contains a graphical user interface (GUI) and a storage device. The GUI displays the standardized UCMS health information and can also keep a log of the data via the storage device available. The remote-PC can have multiple uses as well, however, for this study the focus is the black box and the diagnosis module.

Fig. 3 presents a tier view of the system architecture, the system can be divided into 4 tiers. The logic tier contains the black box that contains the fault diagnosis module. Both Fig. 2 and 3 can give the reader more insight into the designed system architecture.

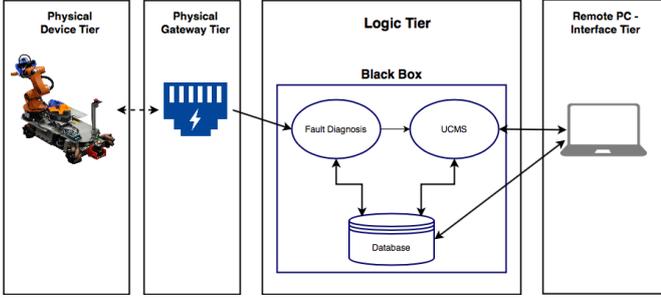


Figure 3: Tier view of system architecture containing physical, physical gateway, logic and remote PC - interface tiers.

4 Consistency-based Diagnosis Vs. Sensor Fault Detection & Diagnosis

4.1 Consistency-Based Diagnosis

Founded by Reiters [11], consistency-based diagnosis is used in diagnosing systems by describing the correct behaviour of the components and the way components interact. CBD uses First Order Logic (FOL) to describe the behaviour of the system including its components. A model of a system is then a collection of the FOL statements describing the behaviour of the components. These models are used to diagnose the system based on observations of the real system. Describing a system in terms of its components means that the system can be decomposable, which can help in fault isolation. A description of a system can then be split into three main parts [18]:

- Behaviour of component types.
- List of Components.
- Component structure.

The following example will be used in describing the modelling and diagnosis process of Consistency-based diagnosis. Usually, logic circuits are used to describe CBD, such as the circuit shown in Fig. 4.

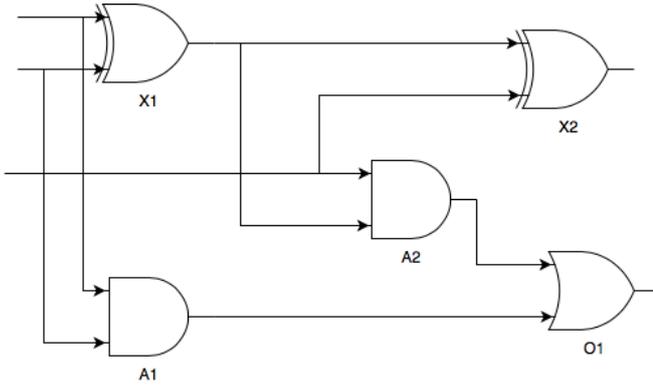


Figure 4: Structure of a circuit.

In Fig. 4, components of type AND, OR, XOR gates are connected together. The description of behaviour of component type will be in the following form:

$$type_i \wedge ok(x) \rightarrow \Phi(x) \quad (1)$$

In equation 1, Φ represents a generic formula that states how the component behaves, $type_i$ represents the types of components and x is the component that belongs to type x .

After describing the behaviour of component types, the components in the system need to be listed along with their types. Therefore for Fig. 4, the COMPTYPES list will be as follows: {ANDG(A1), ANDG(A2), ORG(O1), XORG(X1), XORG(X2)}

Finally, the system structure needs to be given, stating the connections between the components.

$$\begin{aligned} CONN = \{ & out(X1) = in2(A2), \\ & out(X1) = in1(X2), \\ & out(A2) = in1(O1), \\ & in1(A2) = in2(X2), \\ & in1(X1) = in1(A1), \\ & in2(X1) = in2(A1), \\ & out(A1) = in2(O1) \} \end{aligned} \quad (2)$$

After listing the connections between the components (CONN), the system description (SD) becomes complete. Therefore a complete system description is:

$$SD = TYPES \cup COMPTYPES \cup CONN \quad (3)$$

This system description is now the model of the system that can be used for diagnosis. In order to diagnose a system, observations (OBS) are required to be drawn from the real system. A diagnosis problem is defined as a triple:

$$(SD, COMPS, OBS) \quad (4)$$

where SD is the system description, COMPS is the set of suspected components and OBS is a finite set of first-order sentences containing the observations made of the system. As the system is running, consistency is checked between the model and the observations. If an inconsistency exists, a diagnosis is called for. In a CBD problem, a diagnosis assumes that only a subset of the components are faulty and the rest are normal. Therefore:

$$D \subset COMPS \quad (5)$$

where D is minimal set of components such that:

$$SD \cup OBS \cup \{AB(c) | c \text{ in } D\} \cup \{\neg AB(c) | c \text{ in } COMPS - D\} \quad (6)$$

is consistent. Meaning that the system description along with the observations are consistent with the system components contain a set of faulty components or in other words abnormally behaving components.

Finally, to find the faulty components in D, a hitting set algorithm is used. This is done by assuming that one of the components in D is faulty, and then checking for consistency with the system description, this is done repeatedly until a component can be declared as faulty.

Using CBD to diagnose faults in robots has been done previously in different studies [3], [19], [20]. The advantage of using CBD for robots, is that it can be applied to different levels of the system. For example, CBD can be used in diagnosing faults in a wheel motor by modelling the components of the motor and checking for consistency, or it can be used in diagnosing the whole robotic system by modelling all the available components and their connections on a system level, rather than a component level. The limitations of CBD is that it can only diagnose what is stated in the models, therefore, if a fault occurs in a component which is not

modelled, it will not be detected. Another limitation is that modelling a system thoroughly takes a substantial amount of time, and the diagnosis will only be as accurate as the modelling is.

4.2 Sensor Fault Detection & Diagnosis

In [12], Khalastchi et al. derive a method for sensor fault detection and diagnosis for autonomous systems. The fault detection method used in this paper is a combination of a data driven approach with a model-based approach. The advantage of this method is that it can be used online, as the data is being produced by the sensors.

The study focuses mainly on two kinds of sensor faults, which are difficult to detect; stuck and drift. Stuck is when a sensor is producing the same value regardless of the state of the system, drift is when the readings being produced are gradually increase or decrease over time and deviate from the real reading. It is difficult to imply that a sensor is faulty if it is stuck, since it might be operating within its normal range, therefore using minimum and maximum values as thresholds will not detect that the sensor is in a faulty state. As for drift, it is also difficult to imply that a sensor is faulty when there is drift because of the same mentioned problem.

To address this problem, the authors combine a data driven approach that determines the correlation of sensors to each other using Pearson Correlation, and a structural model that represents which sensor relies on which internal hardware component. The advantage of using such a model over other methods such as mathematical models is that it is easier to derive and represent.

The authors describe their process as follows:

- A sliding window of size $m \times n$, named H^t is taken as input to the process. Where m is the number of time steps per window containing the reading of the sensor, and n is the number of sensors to be monitored.
- The sliding window is split into two halves. The first half of the matrix, is subjected to a correlation test to determine which sensors are correlated to each other. The output of this test, for each monitored sensor, is a set of correlated sensors.
- After the correlation is determined, the second half of the sliding window H^t , is checked to see if the correlation of the sensors hold. If there exists a correlation between two sensors in the first half, and the correlation does not hold in the second half, the sensor is marked as uncertain, and a pattern is assigned to the sensor.
- Using the structural model of the system, and the marked sensor, along with its set of correlated sensors, a check is performed to find out if this change in correlation is due to a fault or a response of the system.
- If it is found that a fault has occurred, the structural model is again used to determine which component or sensor of the system is responsible for the fault.

The algorithms developed in this [12] study were tested on a laboratory wheeled robot, having three sonar range detectors and three infra-red range detectors, and on a flight simulator which gives the authors the ability to inject faults and test their algorithms. The results of the tests were promising showing high fault detection rates and very low false alarm rates.

A drawback of the proposed approach is that it only targets sensors which provide single dimension data. Meaning that some complex sensors such as cameras can not be taken into consideration when using this algorithm. It was also noted that the proposed method does not take care of positive or negative correlations, and takes only the absolute value of the correlation. Since the positive and negative correlations are not taken care of, the drift pattern proposed in the study, can not also be detected. Finally, the computational complexity of the algorithm increases with the increase in number of sensors monitored, making the algorithm difficult to implement on a PC with limited resources.

4.3 Comparison

After investigating both methods, the following analysis can be made. Table. 1 summarizes the key differences between both methods. It can be seen from the comparison and the discussions presented on both methods, that consistency-based diagnosis is more suitable for our project since it is, most importantly, lower in computational complexity and has the ability to cover a wider range of components without the need of extracting a special kind of data, such as one-dimensional data for SFDD. CBD also requires less pre-processing to the incoming data since the data only needs to be filtered from noise, if it exists. On the other hand, SFDD will require the signals from different components to be one-dimensional, filtered from noise, subjected to a correlation test, subjected to a test to check if the correlation holds, before being used for fault detection or diagnosis. For these reasons, the selected method to be tested for this study is CBD.

Table 1: Comparison Between CBD and SFDD.

Category	Consistency-based Diagnosis	Sensor Fault Detection & Diagnosis
Operation	online.	online.
Modelling	Requires model containing components, connections, and component types.	Requires structural model.
Preprocessing	Filter readings if required in windowed fashion.	filtering in a windowed fashion + correlation test + List of suspicious sensors.
Monitored Components	Can monitor different kinds of components.	Can monitor different kinds of components provided they generate one-dimensional data.
Computational Complexity	Low computational complexity.	Computational complexity increases with increase in components monitored.
Comments	Low initial set up time. Will use the available data from components for fault diagnosis.	Higher initial set up time, since one-dimensional data needs to be drawn from components, which may not be available to start with.

5 Health Data Standardization & Remote Communication

This section will give an overview of the Usage and Condition Monitoring System (UCMS) data model module and the Health and Usage Monitoring System (HUMS) [21] data standard which was developed by the UK GVA for vehicles. The UCMS data model module is derived from the HUMS data standard and is developed mainly for military vehicles.

The UCMS data model module contains multiple classes that are to be used for health representation of components and subsystems across a military vehicle platform. The following classes are currently part of the data model:

- **Monitored Entity:** Represents what component or subsystem is being monitored and contains information on the health status of the component or subsystem.
- **Monitored Entity Specification:** Contains information on the specifications or static data that are related to the monitored entity.
- **Monitored Characteristic:** Represents which aspect of the monitored entity or subsystem is being monitored. It contains information on the value of the aspect that is being monitored.
- **Monitored Characteristic Specifications:** Contains the static data that is related to the monitored characteristic, such as the unit and description of the monitored characteristic.
- **Threshold:** Contains information on the thresholds that are related to the monitored characteristics, such as the value of the threshold and the threshold type.
- **Threshold Specification:** Contains the static information related to the threshold.
- **Threshold Exceeded Event:** Represents when a threshold is exceeded and records the time that this event occurred and the value that was recorded of this event.
- **Failure Event:** Represents an event where a failure of the monitored entity has occurred and contains a description of the failure along with other information that can be useful to the operators.

Naturally, vehicle components are fundamentally different than components that are to be deployed on robotic platforms. Vehicle components, especially military vehicle components, are much more durable and are manufactured for higher reliability since they are to be deployed in harsh environments and also, to give confidence to the crew operating these components since they may be putting their lives in danger. However, faults may still occur and they need to be handled quickly to avoid breakdowns or complete failures of components or subsystems. Moreover, military vehicles are constantly upgraded to cope with the technological advancements and to have tactical advantage being on the field. Therefore, adopting a health standard that enables easier and faster upgrades to the vehicle platform would be crucial. This requirement is also shared with robotic platforms. Because of the higher availability of components such as sensors and actuators in the market, robotic platforms receive component upgrades and get reconfigured frequently to adopt to different use cases.

Since the NGVA community, which developed the UCMS data model module uses Data Distribution Services (DDS) for data exchange, it was also used in this project to communicate with the remote-PC. The specific DDS software used in this project was developed by RTI³. Some of the advantages and key features of using DDS will be explained in this section.

DDS provides a databus for applications which require real-time data exchange. Distributed systems, such as our project here, require real time data exchange, between the black box and the remote-PC for example. Applications such as this project as well, require reliability and independence from the robotic platform itself, since the goal is to

monitor the health of the robot, regardless of its state and always give the user an overview of the behaviour of the robot and its monitored components. Using DDS in this project helps achieve this goal, since it creates independence between the black box and the remote-PC. Moreover, since the data would be available on the databus, the applications will only deal with the data, instead of needing to deal with another application, along with its dependencies. Another advantage of using DDS, is the availability of Quality of Service (QoS) policies, which allow the publishers, subscribers, data readers or writers of the applications to adopt certain behaviours. Some of these attributes include:

- **Durability:** a parameter that can specify if the data that is produced by a certain entity, such as a data publisher, is available for new or late joiners such as applications requiring this data.
- **History:** it is a parameter that specifies the volume of data to store on the data bus to be available for other subscribers.
- **Reliability:** a parameter that deals with how the network deals with lost samples.

6 Experimental Evaluation

6.1 Setup

Since the goal of the study is to design a fault diagnosis system that can incorporate hardware and software components, the following setup was used for experimenting:

- A service robot running ROS, with a wide range of components such as motors, cameras, laser range finders.
- A Raspberry Pi 3 Model B with a 375 GB storage device attached to the robotic platform via the robot's network interface.
- A remote-PC to receive the standardized health messages and inform the user of the health status of the monitored components.

As for monitoring and diagnosing the health of components, three components were selected to be modelled using a CBD-ROS library⁴, namely, a USB camera, a microphone, and a software node generating a noisy signal. Fig. 5 depicts the setup of different ROS nodes and topics used for experimentation.

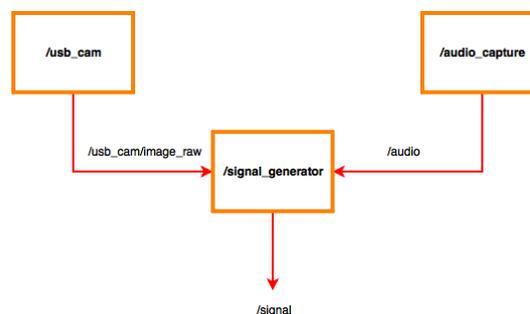


Figure 5: Setup of ROS nodes and topics for experiment.

Having the robot and different nodes setup for the experiments, the observers can now be applied to the ROS nodes

³<https://www.rti.com>

⁴http://www.ist.tugraz.at/ais-wiki/model_based_diagnosis

and topics. This system description is to be deployed on the Raspberry PI, which contains the nominal behaviour of how the different components and topics behave. Fig. 6 presents the different observers deployed on each node and topic.

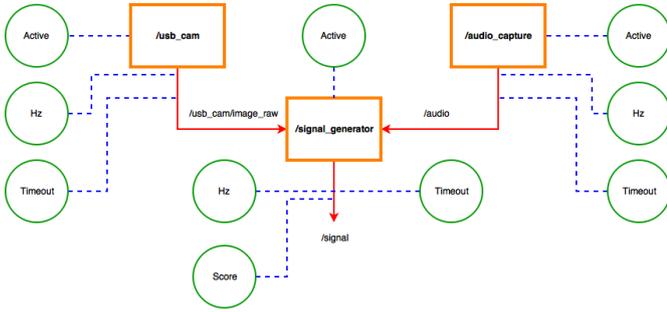


Figure 6: Observers applied on nodes and topics.

The parameters used for each of the observers are given in table 2. The parameters here represent the expected behaviour of the nodes. After the diagnosis engine is setup on the Raspberry Pi, the observations produced by the engine need to be mapped to the UCMS standard discussed in section 5.

Table 2: Assignment of observers to ROS nodes and topics.

Node/Topic	Observer parameters			
	Active	Hz	Timeout	Score
/usb_cam	Applied			
/usb_cam/Image_raw		Filter: mean Window size: 50 Hypothesis: Gauss mean: 15 std_dev: 0.01	1 second	
/audio_capture	Applied			
/audio		Filter: mean Window size: 50 Hypothesis: Gauss mean: 30 std_dev: 0.01	1 second	
/Signal_generator	Applied			
/signal		Filter: mean window size: 50 Hypothesis: Gauss mean: 15 std_dev: 0.01	1 second	Filter: mean Window size: 50 Hypothesis: Gauss mean: 15 std_dev: 0.5

The mapping of the ROS nodes and topics to the UCMS data model module is shown in Fig. 7 and table 3.

Table 3: Mapping of CBD output to UCMS topics.

Consistency-based Diagnosis output	UCMS mapping
Observer Resource	Monitored Entity
Observer type	Monitored Characteristic
Observer threshold	Threshold
Observer threshold exceeded	Threshold Exceedence Event
Diagnosis	Failure Event

From this mapping, each ROS node will be mapped as a Monitored Entity in UCMS along with Monitored Entity Specifications. Each ROS node produces topics that publish information to the robot network that can be used by other nodes or can be used to activate some actuator for example. Each ROS node and its related topics can have multiple observers, which will be mapped to the Monitored Characteristic and Monitored Characteristic Specification topics. The different observers will be easily distinguishable since they will have different source IDs. The thresholds that will be applied to each observer (if applicable) will be mapped to the Threshold and Threshold Specification topic. Once an

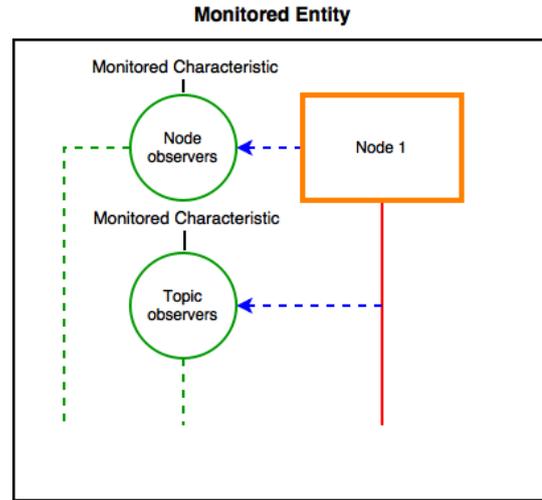


Figure 7: Mapping of nodes and observers.

observer reports a threshold has been exceeded, this will be mapped to a Threshold Exceedence Event topic. If a faulty observation is observed, and the diagnosis engine produces a node that is marked as faulty, this will be mapped to the topic Failure Event.

6.2 Results

After injecting random faults into the system such as killing of nodes, changing the frequency of publishing of topics and unplugging the actual camera and audio devices to induce faults into the system. The results of the tests were recorded after 20 runs of 5 minutes for each test with a random number of faults. The results can be summarized as follows:

Table 4: Experimental Results

TOTAL Injected	290
TOTAL Detected	320
TOTAL True Positive	276
TOTAL false positive	23
TOTAL false negative	12
Accuracy	88%
Precision	0.92
Recall	0.96
F1 score	0.93
Average CPU usage	32.5%
Average Memory Usage	142.2 MB

The results show that a complete diagnosis system can be deployed on a mini-PC while using only less than 35% of its computational powers by using consistency-based diagnosis. Using DDS for remote communication with the off-board PC helps in creating independence between the robotic system along with the black box and the remote-PC, since the remote-PC is only interfacing with the health data which was standardized after being produced by the diagnosis engine. As well as transmitting health messages to the remote-PC, the data store also keeps the health data of the monitored components, acting as a redundant component in case communication is lost with the remote-PC.

7 Conclusion

This study presented a general system design for non-intrusive fault diagnosis of robotic platforms by the addition of a black box. The goals of the study is to create a system that can be deployed on different robotic platforms easily with minimum setup. Two methods of fault diagnosis were selected to be investigated, namely, consistency based diagnosis and sensor fault detection and diagnosis [12]. CBD was selected to be implemented in our proposed system and showed a high accuracy of 88%.

As for standardizing and transmitting health data of monitored components to a remote-PC, the usage and condition monitoring system (UCMS) data model module, developed by NATO generic vehicle architecture team was used. It was found that using DDS for transmitting the standardized health messages to the remote-PC helped in adding reliability to the system since it is independent of the robotic platform. Also, the addition of a storage device to the system for logging the health data produced by the diagnosis engine increases reliability since it keeps a log of the health of the components even in case of loss of communication with the remote-PC.

It was found that a complete diagnosis system can be deployed on a mini-PC while only using less than 35% of its computational capabilities.

References

- [1] D P. Sellers, A J. Ramsbotham, Hal Bertrand, and Nicholas Karvonides. International assessment of unmanned ground vehicles. page 76, 02 2008.
- [2] Eliahu Khalastchi and Meir Kalech. On fault detection and diagnosis in robotic systems. *ACM Computing Surveys (CSUR)*, 51(1):9, 2018.
- [3] Safdar Zaman, Gerald Steinbauer, Johannes Maurer, Peter Lepej, and Suzana Uran. An integrated model-based diagnosis and repair architecture for ros-based robot systems. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 482–489. IEEE, 2013.
- [4] Eliahu Khalastchi, Meir Kalech, and Lior Rokach. Multi-layered model based diagnosis in robots. *23rd International Workshop on Principles of Diagnosis, UK*, 2012.
- [5] Leo H Chiang, Evan L Russell, and Richard D Braatz. *Fault detection and diagnosis in industrial systems*. Springer Science & Business Media, 2000.
- [6] Monica L Visinsky, Joseph R Cavallaro, and Ian D Walker. Expert system framework for fault detection and fault tolerance in robotics. *Computers & electrical engineering*, 20(5):421–435, 1994.
- [7] Stergios I Roumeliotis, Gaurav S Sukhatme, and George A Bekey. Sensor fault detection and identification in a mobile robot. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, volume 3, pages 1383–1388. IEEE, 1998.
- [8] Kaci Bader, Benjamin Lussier, and Walter Schön. A fault tolerant architecture for data fusion: A real application of kalman filters for mobile robot localization. *Robotics and Autonomous Systems*, 88:11–23, 2017.
- [9] Michał Zając. Online fault detection of a mobile robot with a parallelized particle filter. *Neurocomputing*, 126:151–165, 2014.
- [10] Mien Van and Hee-Jun Kang. Robust fault-tolerant control for uncertain robot manipulators based on adaptive quasi-continuous high-order sliding mode and neural network. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 229(8):1425–1446, 2015.
- [11] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57 – 95, 1987.
- [12] Eliahu Khalastchi, Meir Kalech, and Lior Rokach. Sensor fault detection and diagnosis for autonomous systems. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 15–22. International Foundation for Autonomous Agents and Multiagent Systems, 2013.
- [13] NATO. STANAG 4754, NATO Generic Systems Architecture (NGVA) for Land Systems, Edition 1, Ratification Draft 1, November 2016.
- [14] Generic Vehicle Architecture GVA. issue 1, 2010.
- [15] Gerardo Pardo-Castellote. Omg data-distribution service: Architectural overview. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, pages 200–206. IEEE, 2003.
- [16] NATO. AEP-4754 NATO Generic Systems Architecture (NGVA) for Land Systems, Volume III: Data Infrastructure, Edition A, Version 1, Ratification Draft 1, November 2016.
- [17] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.
- [18] Claudia Picardi. A short tutorial on model-based diagnosis. 2005.
- [19] Ahmad Drak, Youssef Youssef, Anastassia Kuestenmacher, and Paul Plöger. Remote Fault Diagnosis of Robots Using a Robotic Black Box. In *The 27th International Workshop on Principles of Diagnosis: DX-2016*, Denver, Colorado, 2016.
- [20] Stefan Loigge, Clemens MÄijhlbacher, Gerald Steinbauer, Stephan Gspandl, and Michael Reip. A model-based fault detection, diagnosis and repair for autonomous robotics systems. 5 2017.
- [21] Defence Standard 23-009 Part 3 Generic Vehicle Architecture (GVA) Part : 3 : Health and Usage Monitoring System (HUMS). (3), 2016.