Prediction of Component Failures of Telepresence Robot with Temporal Data

Jayesh Soni, Nagarajan Prabakar, Jong-Hoon Kim(1)

School of Computing and Information Sciences
Florida International University, Miami, FL 33199

(1) Department of Computer Science, Kent State University, Kent, OH 44242

jsoni@fiu.edu, prabakar@fiu.edu, jkim@cs.kent.edu

ABSTRACT

With recent advances in computer and sensor technologies in the last few decades, the use of robots for various applications has increased enormously. The reliability of robots depends on the minimization of component failures and downtime. To improve the reliability, periodic monitoring of components and their behavior are essential to inference component fatigue and potential breakdowns.

Since fully autonomous robots are very expensive, telepresence robots are affordable for mass scale deployment and can be controlled by a trained human operator like avatars. To increase the efficiency and to reduce the downtime of telepresence robot service, it is essential to observe the various commands performed on the robot and to analyze the samples of component status over a long period.

We propose an efficient data driven model with a collection of frequent time-stamped data from various components of a telepresence robot and predict potential failure warnings. The collected historical datasets are analyzed to determine an accurate machine learning model for increased failure prediction of components. Analysis of this large collection of data will be performed on a cloud computing platform to alleviate the computational load on telepresence robots. With the incoming temporal data, this machine learning model predicts the component status and probability of failure in real-time.

Potential Applications of the proposed approach also includes detection of component malfunction, estimating the degree of movement of various components for satisfactory level of performance, and migration of workload among multiple telepresence robots in a team work environment.

Keywords

Telepresence Robot, Machine Learning, Predictive Modeling, Predicting Component Failure

1. INTRODUCTION

With recent advances in computer and sensor technologies in the last few decades, the use of robots for various applications have increased enormously. Globally, many research organizations are concentrating in research and development of various types of robots based on their requirements and applications. Telepresence robotics is one such area of robotics that allows human controllers to remotely operate through a real-time multimedia interface in wireless mode. Remote teleoperation of advanced robotic technologies has great potential in many areas, including applications in education, deep sea exploration [1] explosive mine removal, hazardous environments [2] off-shore projects [3], space explorations [4], etc.

Scheduled maintenance in such robotic system is widely used to ensure that equipment is operating correctly so as to avoid unexpected breakdowns. Such maintenance is often carried out separately for every component, based on its usage or simply on some fixed schedule. However, scheduled maintenance is labor-intensive and ineffective in identifying problems that develop between technician's visits. Unforeseen failures still frequently occur. In contrast, predictive failure techniques help determine the run-time condition of components in order to predict when and what repairs should be performed. The main goal of predictive failure is to prevent unexpected equipment failures.

Predictive failure requires insight into the dynamics of operational components. This can be gained by adding sensors to components for recording and monitoring of signals of interest (such as temperature and voltage). Sensor data log with time stamp information will provide further understanding of a working component. Most of the components are usually operated via software. For example, in case of Telepresence Robot, all device operations, from body movement to usage of battery power, are controlled by various applications. These applications record operation logs. Theoretically, one can trace back how a component was used by analyzing all its associated logs. Mining such rich information can help in detecting potential issues like component failures in advance. A good technique in this area can lower the costs of damage, improve security and reduce the number of unnecessary maintenance service.

2. PROPOSED METHODOLOGY

The use of component logs to predict failures poses many challenges and has not yet been fully explored. Since logs are mainly used for debugging purposes, they (i) rarely contain explicit information for failure prediction; (ii) contain heterogeneous data including symbolic sequence, numeric, time series and categorical variables; and (iii) contain massive amounts of data, posing computational challenges. Before the data can be analyzed, it must be determined how the data can be acquired, what kind of data is accessible, and which format it takes. This data acquisition process is discussed in section 2.1. After the acquisition of data, features need to be extracted from the raw data which is known as feature engineering that involves signal processing, presented in section 2.2. For the feature analysis, the relationship between given input and expected output must be established through data labeling techniques as described in section 2.3. Finally, a set of machine learning (ML) techniques that analyzes the features and predict component failures as outlined in section 2.4. The workflow model for these four phases is shown in Figure 1.

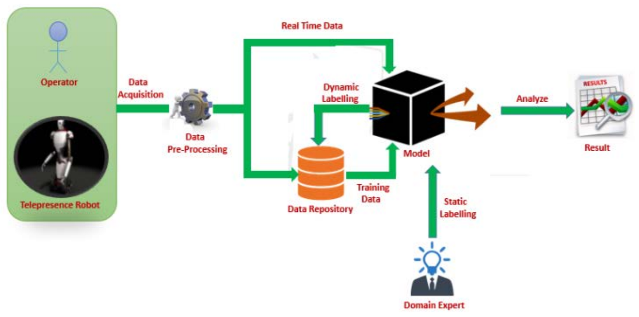

Figure 1. Workflow of the proposed model

2.1 Data Acquisition

2.1.1 *Types of data*

The two most important types of data for component failure prediction are sensor data from sensors and event data from log files.

Sensor data

In recent years, sensors have become smarter, smaller, easier to implement in existing systems, as well as cheaper and more reliable [5]. A sensor converts physical values into electrical values (voltage, current or resistance). Usually, a sensor measures a mechanical value; for example, the mechanical values of acceleration, pressure, flow, torque or force. With this mechanical value, one can interpret the vibration data, acoustic data, temperature, altitude, etc.

Log-files

Events of a system can be recorded to a log file with time-stamp. In this manner, the declaration of an event is very broad. For instance, an event could be component breakdown, replacement, maintenance service, etc. Descriptive log files can be written by humans describing maintenance actions, or any failures or errors detected.

To make use of log data, we first have to interpret the logs, filter out a large amount of noise (i.e. data irrelevant to our goal) and extract predictive features. Next, we have to collect known failure cases for learning/evaluating models. Then, we have to apply machine learning techniques to accurately predict component failures. And finally, we have to choose the best Machine Learning model by choosing appropriate evaluation strategies.

2.1.2 *Data Acquisition Technique*

Data can be collected through either pull-based (polling at periodic intervals) or push-based (when event occurs).

Time Interval

Time interval data acquisition system sends the acquired data at periodic time intervals. This type of system is used when communication costs are high. In some cases, it is necessary to have synchronized clocks with the communication partner. They have to determine an interface for how and what data have to be exchanged. Such interfaces can differ, from being very abstract (i.e., all values will be transferred) to very special (transfer a value only when a constraint is fulfilled).

Event Driven

The event driven data acquisition technique is a push-based approach. Such systems are also called publisher/subscriber systems [6], because the user of such systems subscribes to a publisher of events. In our case, a subscription is a definition of behavior of the real-world system that the user wants to observe. The data acquisition system publishes the value changes detected (called Event) and sends them to all registered users.

2.2 Feature Engineering

For most failure type detection and predictive maintenance applications, the values obtained from the data acquisition system must typically be preprocessed before transforming them into the predictive features for improved accuracy of the machine learning algorithm. The most important processing of the feature engineering for failure detection as shown in Figure 2 with the following stages: i) Signal Processing: The interpretation, generation and transformation of raw unprocessed data, ii) Feature Selection: Selection of the most representative features, and iii) Feature Extraction: Generation of new information by combining features.

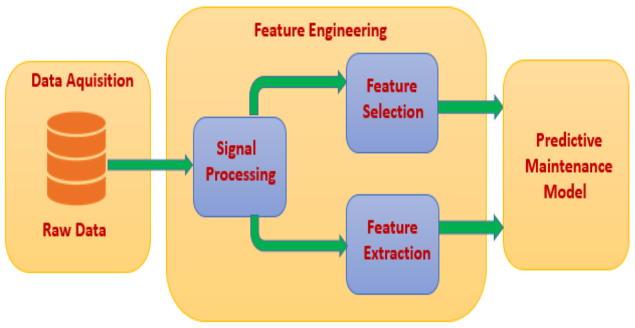

Figure 2. Feature engineering process

2.2.1 *Signal Processing*

A training set is initially extracted from the acquired data set for building the ML models. Since the performance of a classifier is directly influenced by the quality of its training data set, the presence of noise in the training data may affect its performance, decrease its accuracy, and increase its complexity [7].

Two different combinations of the classifiers results are considered. The first is majority voting, in which an example is noisy if the majority of the predictions made by the combined classifiers is not in accordance to its recorded label. The second considers a consensus of the predictions, by which an example is considered noisy if all classifiers make erroneous classifications for it.

Since there are four classifiers being combined for noise detection, a data item is considered noisy by consensus only if it is misclassified by all four classifiers. Similarly in majority voting, a data item is noisy if at least three classifiers make incorrect predictions for it.

After the noise is identified, the associated data can be processed in the following two ways. The first approach removes noisy data from the training data set. The second approach reclassifies the noisy data with new class labels. The new class will be the one, that will be predicted by most of the noise-detection classifiers.

2.2.2 Feature Selection

To select appropriate features, we will use Principal Component Analysis (PCA) which is an orthogonal transformation technique that converts a set of features into a set of linear uncorrelated features, called principal components. The assumption of the PCA is that features with the largest variance have the largest informative content. The samples are centered and rotated in accordance to most relevant features. The outcome of the PCA yields the features with the largest variance that are orthogonal to one another.

2.2.3 Feature Extraction

Feature selection picks a subset of the most representative features, whereas feature extraction derives new information from original features. Feature extraction can be a non-linear process, and thus the results are not self-explanatory. An advantage of feature extraction over feature selection is that the features can be reduced to a much greater extent. We will try with different combination of the feature extractions and will check which combination gives the best accuracy.

2.3 Data Labelling

For supervised learning, historical data is needed which has to be labeled. Machine learning approaches can create relationships between the input feature vector and the class labels.

ML can distinguish between classification and regression. Classification means that an output variable is discretized to a defined class labels. Regression means that an output variable takes on continuous values. The state-driven data labeling technique is used for classification in component failure prediction.

State Driven Data Labeling

The state-driven data labeling technique labels the historic data into different states. There are two different types of states in component failure detection:

- Failure type state
- Lifetime state

A component can have one or more failure types. A failure type can also depend on the consequences of other preceding failure (see Figure 3). The lifetime state is usually equally distributed over the lifetime of a component as illustrated in Figure 4.

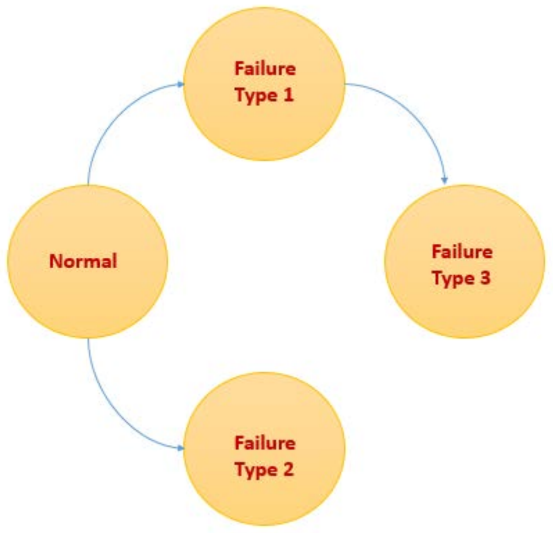

Figure 3. State machine for failure type detection

According to recent literature, the historic data with logged failure types is needed for detecting these different failure types. This implies that every failure type must occur in the historic data, and the more often it occurs, the higher the possibility to eliminate noise factors and achieve a better accuracy. The problem of generating failure type data is costly, and therefore it is not always possible to force the system into a failure situation. Alternatively, many real-world systems are maintained with preventive techniques, where components are replaced after a predefined time, before a failure can occur. It does not matter whether the component is close to failing or not. Failure type detection, without the failure type of historic data, is not easy and not mentioned in the literature. However, it might be useful in analyzing failure types, because some these do not lead to a system crash, but can influence other components and their healthy state.

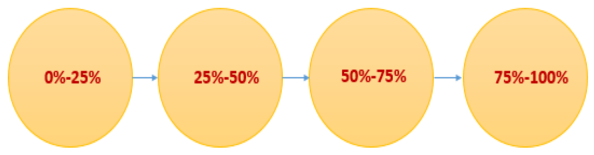

Figure 4. State machine with equal distributed states

In lifetime-state data labeling for predictive maintenance, it is necessary to categorize the historic data into the lifetime states used. As mentioned above, these states are generally equally distributed over the lifetime of a component. There are two minimum number of states for predictive maintenance (<100% and 100%).

2.4 Machine Learning Approaches

In recent years, machine learning has become increasingly important in computer science because data could be collected and stored easily. The collected data is usually so extensive that it is not practical to analyze the data manually. In such a scenario, the machine learning technique plays a key role.

Another reason for the growing popularity of machine learning is decreasing computational costs. With the evolution of hardware in recent years, the usage of machine learning approaches has become efficient in terms of both cost and time, especially for detecting component failures. Different machine learning algorithm requires different set of parameters that can be learned from the dataset.

We conducted experiments on simulated data with the following three most widely used machine learning algorithms as outlined in Figure 5:

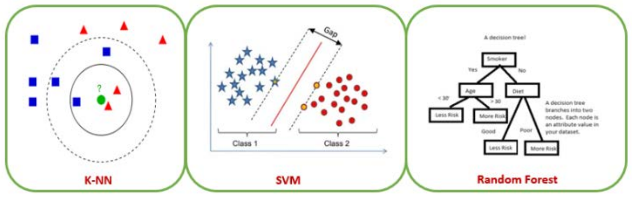

Figure 5. Machine learning models

a) **K-NN** (K-Nearest Neighbor)
In KNN classification [8], the output is a class membership. A new object is classified by a majority vote of its already classified neighbor objects, with the object being assigned to the most common class among its k nearest neighbors.

b) **SVM** (Support Vector Machine)
Support Vector Machines [9] are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. A new object is classified into the most suitable classes using the decision planes.

c) **Random Forest**
Random forests [10] are ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees (classes) at training time and output of a new object is the class that is the most common predicted by the decision trees.

We used a simulated data set with 2207 data points each with 300 features and 23 labels (pre-classified data points) for this experiment. K-Fold [11] cross-validation technique is employed to find the optimal values for parameters for training each of these three ML models. The results of classification of data points are listed below.

Table 1. Experimental results

Machine Learning Model	Accuracy
K-NN	81%
SVM	87%
Random Forrest	85%

Based on the experimental results, SVM performed better than other models.

3. CONCLUSION

We proposed a new approach to predict component failures using past data collection with known component failure types. This will identify the necessary preventive actions that will minimize system downtime and cost. We introduced machine learning approach to analyze the data set and to predict the upcoming component failure.

Potential Applications of the proposed approach also includes detection of component malfunction, estimating the degree of movement of various components for satisfactory level of performance, and migration of workload among multiple telepresence robots in a team work environment.

A large collection of component failure types will enhance the quality of analysis. Since it is a tedious process to identify all possible component failure types, a cloud based data analysis from several telepresence robots will be a viable alternative. As the dataset grows extremely very large, designing a distributed machine learning approach would be needed. Such an approach would offer the advantage that large amounts of data could be analyzed in parallel, which results a shorter time to reach a decision.

4. REFERENCES

[1] A. Forrest, B. E. Laval, Lim, D. Williams, A. Trembanis, M. Marinova, R. Shepard, A. Brady, G. Slater, M. Gernhardt, and C. M. Kay, “Performance evaluation of underwater platforms in the context of space exploration”, *Planetary and Space Science*. vol.58, no. 4, pp.706-716, 2010.

[2] L. Parker and J. Drape, “Robotics applications in maintenance and repair”, 2nd. ed. vol. 1. Hoboken, NJ, John Wiley and Sons, Inc., 1998.

[3] C. Heyer, “Human robot interaction and future industrial robotics applications,” in Proc. of the *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[4] T. W. Fong, “The human exploration telerobotics project,” *Global Space Exploration Conference*, Washington, 2012.

[5] Chris Van Hoof, Kris Baert, and Ann Witvrouw, “The best materials for tiny, clever sensors,” *Science*, vol. 306, no. 5698, pp. 986–987, 2004.

[6] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec, “The many faces of publish/subscribe,” *ACM Computing Surveys (CSUR)*, vol. 35, no. 2, pp. 114–131, 2003.

[7] X. Zhu, X. Wu, “Class noise vs. Attribute noise: A quantitative study of their impacts,” *Artificial Intelligence Review*, vol. 22, no. 3, pp. 177–210, 2004.

[8] https://en.wikipedia.org/wiki/Knearest_neighbors_algorithm

[9] https://en.wikipedia.org/wiki/Support_vector_machine

[10] https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

[11] https://en.wikipedia.org/wiki/Cross-validation_(statistics)