Invited Paper: An Anomaly Detection System for Equipment Condition Monitoring

Makoto Imamura*, Michael Jones**, and Daniel Nikovski **

*School of Information and Telecommunication Engineering, Tokai University, Japan
** Mitsubishi Electric Research Laboratories, USA
*imamura@tsc.u-tokai.ac.jp
**{mjones, nikovski}@merl.com

Abstract - In industrial domains, equipment condition monitoring (ECM) has attracted much attention as the Internet of Things (IoT) has been emerging and growing. This paper describes our anomaly detection system for ECM to solve the requirements in terms of development efficiency, sensor big data management, and feature extraction characteristic of sensor data, which we have experienced in the development of practical systems. First, we proposed a novel relation based query language TPQL (Trend Pattern Query Language) as a basis for declaratively describing the conditions that anomaly sensor data satisfy in order to improve development efficiency and the maintainability of programs. TPQL provides a convolution operator and a time interval join as important common operations for anomaly detection. The former is for extracting features of time series segments, and the latter is for time consuming preprocessing, such as missing value completion and merging data with different sampling periods. Second, we introduce function libraries for TPQL in order to solve the problems in terms of sensor big data management and feature extraction. In terms of sensor big data management, we select key-value store database for accumulating sensor data and provide data transformation functions among key-value, stream and relation to enable the selection of data type in accordance with various processes such as storage, aggregation among relations, and time series processing. Furthermore, we propose an exemplar learning method that can summarize the features of training time series with a smaller set of exemplars for enabling fast anomaly detection even for big sensor data. In terms of feature extraction, we propose a novel leg vibration analysis that can extract the global trend pattern in time series with local fluctuations, so that it can capture the vibration behavior depending on a given amplitude and a given window size.

Keywords: Equipment Condition Monitoring, Anomaly Detection, Feature Extraction, Sensor Data

1 INTRODUCTION

As the Internet of Things (IoT) [1] has been emerging and growing, sensor big data that are streamed from various equipment in power plants, industrial facilities, and buildings can be made available for monitoring, diagnosis, energy-saving, productivity improvement, quality management, and marketing. As a result, industry has paid much attention to the use of big sensor data generated from equipment or facilities in order to create a smarter society.

Equipment Condition Monitoring (ECM) is a typical service that uses big sensor data, and machine learning techniques for big sensor data are key technologies to make ECM smarter [2].

We have published elemental technologies for anomaly detection [3]-[6]. This paper illustrates a total anomaly detection system exploiting the elemental technologies. The rest of our paper is organized as follows. Section 2 describes the problems to be solved, and then introduces an anomaly detection system based on Trend Pattern Query Language (TPQL) to solve them. Section 3 and section 4 discuss leg vibration analysis and exemplar learning, respectively, as key technologies of our system. Section 5 shows an evaluation of our system.

2 ANOMALY DETECTION SYSTEM-BASED ON TPQL

The problems of anomaly detection for ECM can be largely grouped into three categories: development efficiency, management of sensor big data, and accuracy of anomaly detection.

In order to solve the above problems, first, we have proposed a relation based query language TPQL (Trend Pattern Query Language) [3]-[4] to express constraints in time series data for anomaly detection. TPQL is an SQL-like language, and it can help programmers describe application-dependent conditions for anomaly sensor data with a common function library. And then, we implemented an anomaly detection system based on TPQL with Java, and applied it to real applications.

Figure 1 shows the overall structure of our system. A TPQL interpreter calls the common function library, key-value store database, and relational database, by means of a given TPQL script. A TPQL script describes anomaly conditions with the help of the function library, which is designed on the basis of the requirements that we have encountered through our experience in the development of practical ECM systems. Furthermore, configuration management information as input makes TPQL scripts separated from the parameters dependent on actual facilities. This separation improves maintainability of TPQL scripts.

This section mainly illustrates the problems to be solved, our solution to those, and the difference with respect to related work. In terms of the implementation details of TPQL, please refer to [3] and [4].

2.1 Development Efficiency

Programming for data preprocessing that is specific to each application has a large proportion in the development effort for anomaly detection functions. Therefore, we aim to propose a general relation based query language, TPQL, which has a common function library containing functions that are commonly used in our development of anomaly detection systems, while also maintaining the semantics of SQL. TPQL provides two basic common functions for preprocessing: a convolution operation and a time-interval join.

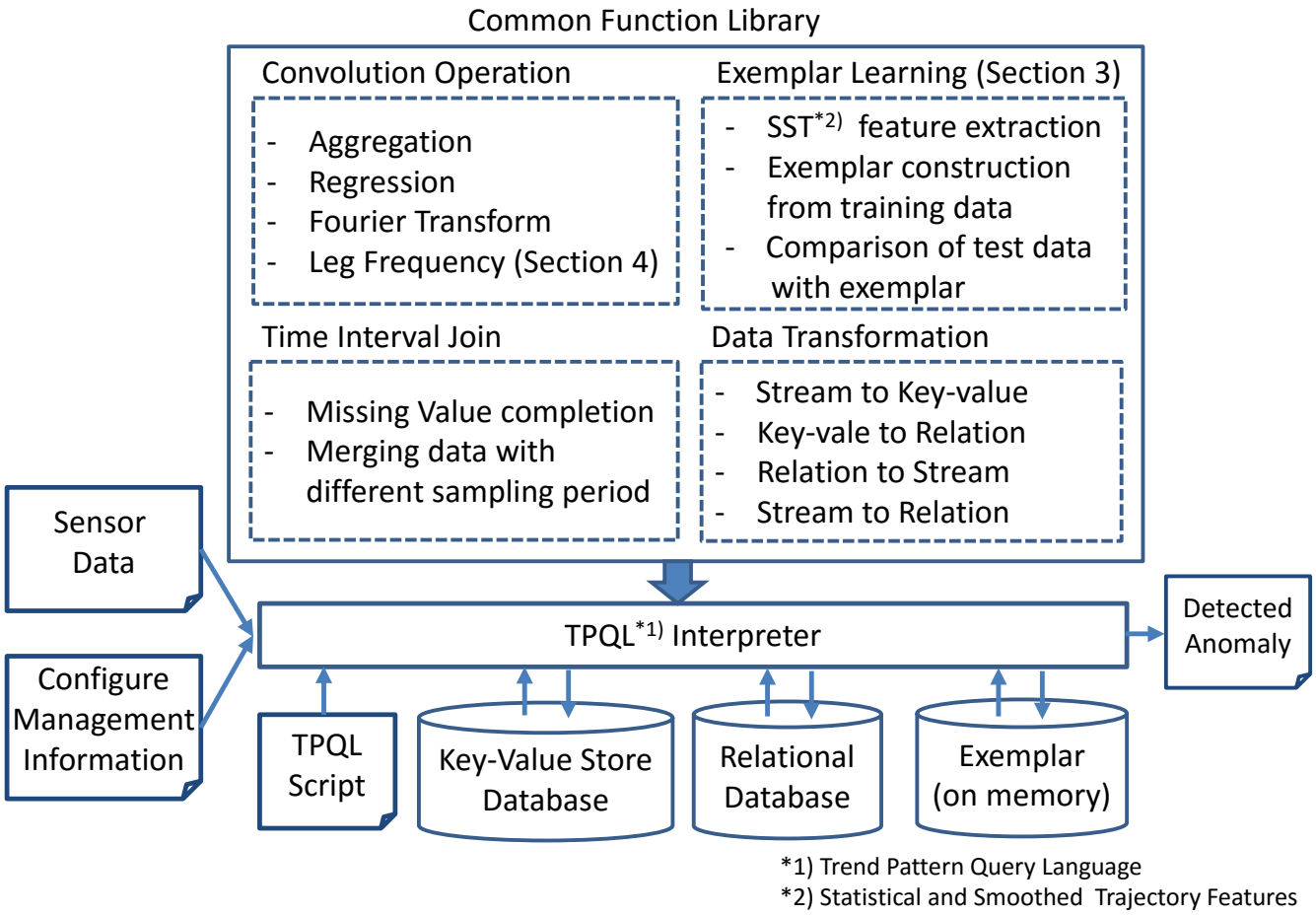

Figure 1: An Anomaly Detection System based on TPQL

(1) Convolution operation

One of the common operations of anomaly detection for sensor data is sliding window processing, because a time series segment for a given time period is a basic processing unit. In a typical anomaly detection procedure, features are extracted from each time segment, and then the extracted features from test data are compared with those for training data. If the features of the test data are different from those of the training data, the test data is determined to be anomalous.

A convolution operation is the cumulative sum of the repeated multiplication of sliding window segments and a given feature function. The output of a convolution operation is a time series which consists of feature values of each time segment of input time series. A convolution operation makes the description of a procedure clear by separating feature functions from the calculation.

Typical feature functions in convolution operations are aggregation, regression, and Fourier transform.

(i) Aggregation functions

TPQL provides standard aggregation functions for time series segments, such as max, min, average or standard deviation.

A typical query example with aggregation is as follows: Let f and g are time series of temperature. “Find the time t when average value of 25 minutes segment f starting from t is 5 degrees higher than that for g.”;

(ii) Regression function

Regression and auto-regression are used to estimate future value from past data. If the deviation between estimated values and actual measurement is large, it is judged to be anomaly.

A typical query example with regression is as follows: “Find the time when the difference between the actual measurement and the estimated value by auto-regression with window size 15 is larger than 5 degrees”.

(iii) Fourier transform

The Fourier transform is known to describe the spectral characteristics for steady-state sensor data. An anomaly is often detected as the change in the frequency spectrum. A typical query example with a Fourier transform is as follows: “Find the time when the spectral density in normal mode is below half that of usual data”.

(2) Time-interval based join

Typical pre-processing steps for raw data are as follows.

- Missing value completion
- Merging data with different sampling periods

We introduced time-interval based join for a time-series data table whose key is a pair of column “time” and column “time interval” to describe the above preprocessing steps, while maintaining the semantics of SQL. We use a method of constructing a subdivision that merges two different subdivisions on the same interval, used in the definition of the

Stieltjes integral, for joining tables with different time intervals.

A standard temporal query language TSQL [7] also supports operation over time intervals, such as intersection and inclusion and so on, but does not support time-interval join.

2.2 Sensor Big Data Management

There are a lot of sensors in a facility, so that a large amount of data will be accumulated as time passes. If there are 10,000 sensors, sampling period is one second and one byte per one point, the amount of data is about 1 Gbytes per one day. If the number of sensors per device is 50, the total number of 10000 sensors would be reched by as little as 200 devices in a facility, or 200 products in the consumer market. Therefore, 10,000 sensors is not an excessive assumption. Furthermore, missing values often occur in sensor data, so relational databases that are frequently used in enterprise domains, may not be suitable, because they have excessively strict data management functions. We propose two functions. One is for data transformation and the other is for fast processing.

(1) Data Transformation

With respect to storage, our system uses key-value store database, which is often used for big data, and provides mutual transformation functions among key-value data, relational data and stream data for developers to select a suitable data type in accordance with the purpose of processing in TPQL scripts. Generally speaking, key-value data are suitable for accumulating data, relational data are suitable for aggregation over relations, and stream data are suitable for data passing to external functions and time series analysis.

The typical stream query language CQL [8] is also a relation-based one, and has a sliding window process as one of its basic operations. CQL provides the transformation between relational data and stream data, that is, “Relation to Stream” and “Stream to Relation”. TPQL adds “Stream to Key-value” and “Key-value to Relation” as basic data transformation functions.

(2) Exemplar Learning

With respect to fast processing, we proposed a compression method that opearates by combining similar data segments into one segment, in order to speed up anomaly detection procedures. We call this method exemplar learning in this paper. Exemplar learning will be illustrated in the next section.

2.3 Feature Extraction Characteristics of Sensor Data

A lot of algorithms have been proposed for anomaly detection [9]. We have a policy to select existing algorithms in accordance with our application requirements. Preprocessing for sensor data is as important as the anomaly detection algorithms themselves. The techniques for preprocessing are sometimes called feature engineering [10], and are a very important factor in determining the success of anomaly detection systems.

Vibrational behavior is very important for anomaly detection. The Fourier transform is a useful and frequently used feature for steady state data, but it is not so useful for transient or non-periodical data from our experience. We introduce a novel feature which we call leg frequency in order to calculate the frequency of variations in time series that include an upward trend and a downward trend alternatively. Leg frequency is treated as a feature function in convolution operation in TPQL.

Rain flow method [11],[12] in material mechanics is a related work. It calculates the amplitude of deformation which causes the fatigues or the cracks form in materials. It calculates the maximal pair of upward trend and downward trend at each maximal point. But our leg vibration analysis calculates the frequency in a time series segment for given amplitude so that it can describe the degree of fluctuation for the given amplitude that is decided to distinguish anomaly from noise with domain knowledge.

A typical query with leg vibration analysis is used in detecting hunting in control systems. A query example is as follows: “Find the time when the number of the alternate occurrences of upward and downward trends whose amplitude are above 3 ℃ during 30 minutes window is larger than 2”. In this example, 3 ℃ and 30 minutes are the parameter that are decided by the application requirement.

Leg vibration analysis will be illustrated in section 4.

3 EXEMPLAR LERNING

3.1 Statistical and Smoothed Trajectory (SST) Features

We proposed statistical and smoothed trajectory (SST) features [5] that can capture the shape and the stochastic behavior of the time series within the window so that it can handle various types of sensor data.

To detect anomalies in a time series, we first learn a model of the time series given normal time series data. To learn a model we use a fixed-size sliding window over the training time series and compute a feature vector representing each window. Our model consists of a set of exemplars representing the variety of feature vectors that exist over all windows in the training time series. The feature vector that is computed for each window consists of a trajectory component that captures the shape of the time series within the window and a statistical component that captures the stochastic component. The trajectory component is designed to capture the low frequency information in the time series window. It is computed using a simple fixed window running average of the raw time series to yield a smoothed time series after subtracting the mean of the window. Because of smoothing, half of the values in the smoothed time series can be discarded without losing important information. Thus, the trajectory component has w/2 elements where w is the number of time steps in the window. Figure 2a) shows a noisy sine wave time series and the corresponding smoothed time series with half the values discarded is shown in Fig. 2b). The statistical component is a small set of statistics computed over time series values in the window which are mainly designed to characterize the high frequency information in the raw time series window. There are many possible choices for a set of statistics that well characterize the high frequency information in a time series window. The statistics used for experiments in this paper are mean, standard deviation (std), mean of the absolute difference $|z(t) - z(t+1)|$ (mean abs diff), number of mean crossings divided by window length (num. mean crossing/w), percentage of positive differences (% pos.diff), percentage of zero differences (% pos.zero), and the average length of a run of positive dif-

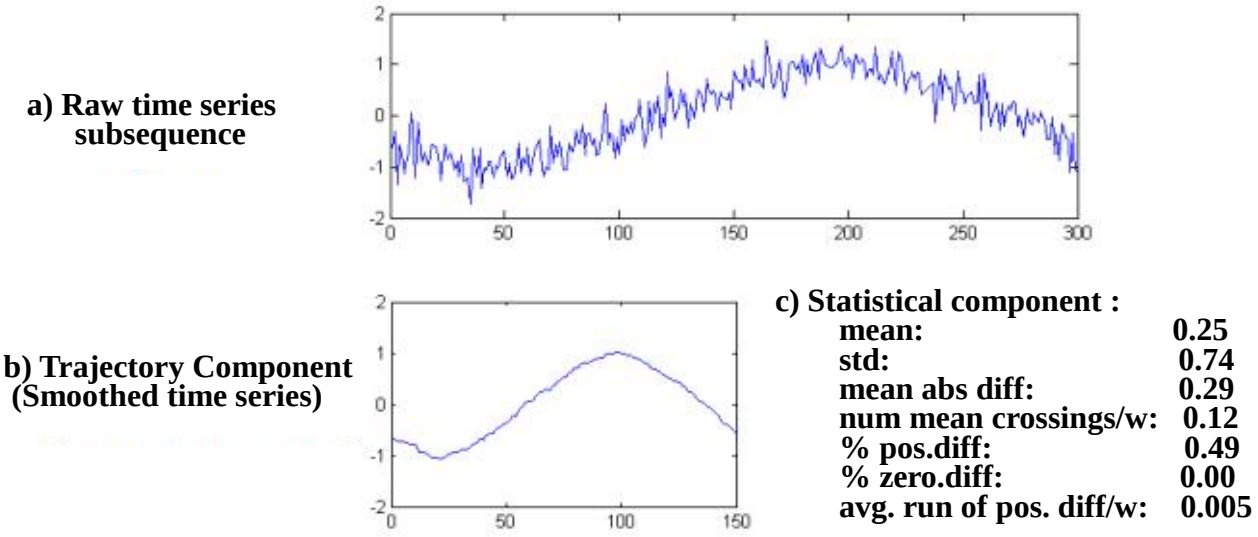

Figure 2: Example time series window (a) along with its trajectory (b) and statistical components (c) Statistical components

ferences divided by window length (avg. run of pos. diff/w). Here, $z(t)$ is the value of the raw time series at time t. Figure 5(c) shows the vector of statistics for an example window. This choice of statistics has worked well in practice across a variety of different time series, but as mentioned before other statistics would likely also work well. The trajectory component is half the length of the window (w/2 time steps), and the statistical component is 7 real numbers for a total of w/2+7 real values. We call this novel representation Statistical and Smoothed Trajectory (SST) features.

3.2 Anomaly Detection using Exemplar Learning

One possible model for a time series is simply the set of all SST features that are computed from all overlapping windows of the time series. This model would be an inefficient representation because the overlapping windows would produce many very similar feature vectors. A much more efficient model is created by finding a small set of exemplars that compactly represent the set of all SST features from the time series. An exemplar in this context is a representation of the SST features of a group of similar windows (overlapping or not) from the training time series. We use an agglomerative clustering algorithm to select SST exemplars from the set of all SST features for a time series.

The agglomerative clustering algorithm works as follows. After computing SST features for every window of the training time series, a set of exemplars is learned by initially assigning each SST feature as its own exemplar and then iteratively combining the two nearest exemplars until the minimum distance between nearest exemplars is above a threshold. This is illustrated in Fig. 3.

We use Euclidean distance to measure the distance between two exemplars:

$$dist(f_1,f_2) = \sum_{i=1}^{\frac{w}{2}} \left(f_1.t(i) - f_2.t(i)\right)^2 + \frac{w}{14}\sum_{i=1}^{7}\left(f_1.s(i) - f_2.s(i)\right)^2$$

where f_1 and f_2 are two feature vectors, $f_j.t$ is the length w/2 trajectory component of f_j, and $f_j.s$ is the length 7 statistical component of f_j. The w/14 coefficient causes the statistical and trajectory components to be weighted equally.

Two exemplars are combined by a weighted average of the corresponding elements. The weight is the count of the number of feature vectors that have already been averaged into each exemplar divided by the total count. Each resulting exemplar is thus simply the overall average of the feature vectors that went into it. The threshold that determines when to stop combining exemplars is set to $\mu + 3\sigma$ where μ is the mean of the Euclidean distances ($dist(f_1,f_2)$) between each initial SST feature vector and its nearest neighbor among the initial SST feature vectors and σ is the sample standard deviation of these distances. The running time of this exemplar selection algorithm is $O(n^2w)$ (where n is the length of the training time series and w is the chosen window size).

After exemplar selection, each exemplar is associated with a set of original SST features that were averaged together to form the exemplar. The standard deviation of each element of the $w/2+7$ length feature vector is then computed and stored with each exemplar. These standard deviations are computed over the set of SST feature vectors associated with a particular exemplar. An exemplar is thus represented by $w/2 + 7$ mean elements and $w/2 + 7$ standard deviation elements. In our experiments, the final exemplar set is typically between 1% and 5% of the total number of features (windows).

After the model is learned, anomalies are found in a testing time series as follows. For each window of the testing time series, an anomaly score is computed. This is done by first computing the SST feature of the window. Then the nearest neighbor exemplar to the SST feature is found. The distance function used is

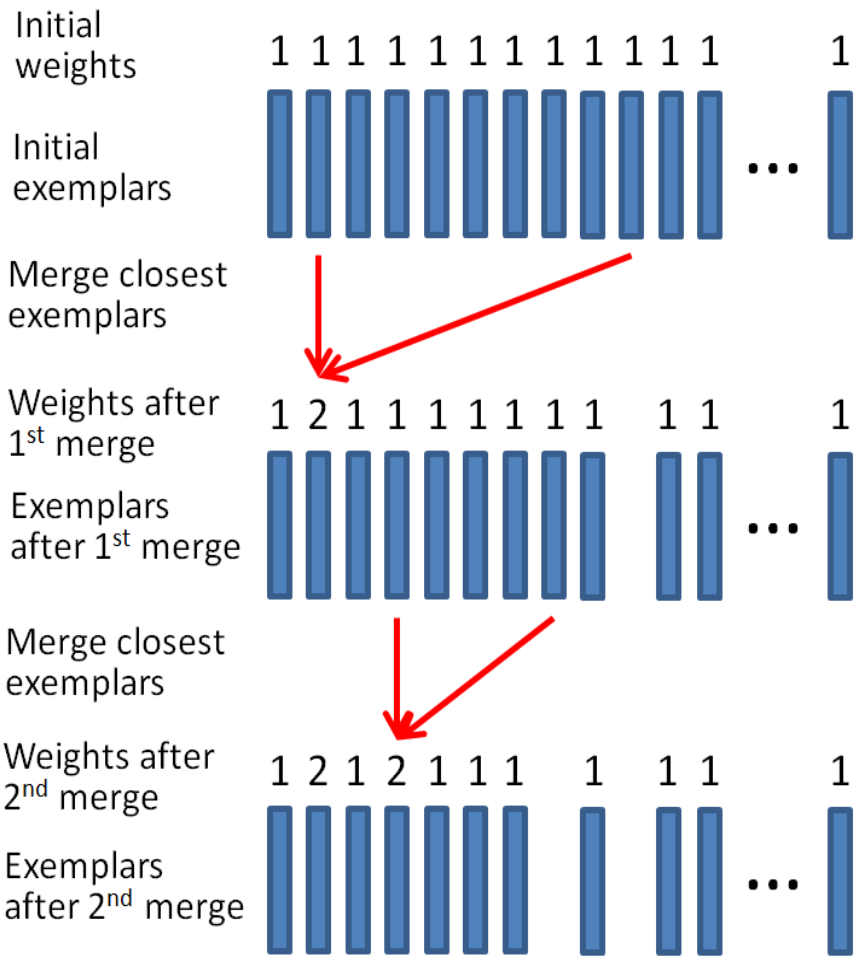

Figure 3: Illustration of agglomerative clustering for learning exemplars. The exemplars (which are SST feature vectors) are represented by blue rectangles. At each iteration the exemplars with minimum distance between them are averaged together using a weighted average. This process is repeated until the minimum distance is above a threshold.

$$d(f,e) = \sum_{i=1}^{\frac{w}{2}} \max\left(0, \frac{|f.t(i) - e.t(i)|}{e.\sigma(i)} - 3\right) + \frac{w}{14} \sum_{i=1}^{7} \max\left(0, \frac{|f.s(i) - e.s(i)|}{e.\varepsilon(i)} - 3\right)$$

where f is the SST feature vector for the current window consisting of a trajectory vector, $f.t$ and a statistical vector $f.s$, e is an exemplar for the current dimension consisting of trajectory $(e.t)$ and statistical $(e.s)$ vectors as well as the corresponding standard deviation vectors, $e.\sigma$ for the trajectory component and $e.\varepsilon$ for the statistical component.

This distance corresponds to assigning 0 distance for each element of the trajectory or statistical component that is less than 3 standard deviations from the mean and otherwise assigning the absolute value of the difference divided by the standard deviation for each element that is more than 3 standard deviations from the mean. In equation 2 and in our experiments, the statistical component is given equal weighting to the trajectory component, although this weighting can be changed based on the application.

4 LEG VIBRATION ANALYSIS

Facility maintenance in buildings, plants, or factories needs to calculate the frequency of variations in sensor data in order to detect a sign of failure or deterioration. Fink et al. proposed a leg search method [13] to find a global trend in a time-series including small variations such as noise. The dotted lines in Fig. 4 are examples of legs. Both lines show the global upward trend that includes local up-down segments.

However, their method treats only single legs so that it can find an upward or downward trend, but can't catch the frequency of variations. We developed leg vibration analysis that can calculate the frequency of variations in time-series that includes upward trends and downward trends that can appear alternately and iteratively. We showed an algorithm whose calculation order is linear in the window size. In contrast, the computational order of a naïve algorithm is factorial in the square of window size.

Definition: time series X, sbsequences X[p:q]

A *Time Series* $X=[x_1,\cdots,x_m]$ is a continuous sequence of real values. The value of the i-th time point is denoted by $X[i] = x_i$.

A Time Series *subsequence* $s = [x_p, x_{p+1},...,x_q] = X[p:q]$ is a continuous subsequence of X starting at position p and ending at position q. We denote the starting time point, the ending time point, and the length of a subsequence l by start, end and length respectively:

$$start(s) \equiv p$$
$$end(s) \equiv q$$
$$length(s) \equiv q\text{-}p\text{+}1$$

Definition: Leg

Let X be time series.

An *upward leg* l = X[p:q] is a subsequence of X that satisfies the following conditions from (1) to (3).

$$\forall i.\ p \le i \le q \quad X[p] < X[i] < X[q] \tag{1}$$
$$X[p-1] \ge X[p] \tag{2}$$
$$X[q] \ge X[q+1] \tag{3}$$

A *downward leg* $l = X[p:q]$ is a subsequence of X that satisfies the following conditions from (4) to (6)

$$\forall i.\ p \le i \le q \quad X[p] > X[i] > X[q] \tag{4}$$
$$X[p-1] \le X[p] \tag{5}$$
$$X[q] \le X[q+1] \tag{6}$$

If l is an upward leg or downward leg, l is called a *leg*.

Definition: Sign and amplitude of a leg

Let X[p:q] be a leg l . We define the *sign* and *amplitude* of a leg l by the following. We denote them by amp and sign respectively:

$$\text{sign}(l) \equiv \text{sign}(X[q] - X[p])$$
$$\text{amp}(l) \equiv \text{abs}(X[q] - X[p])$$

Definition: Leg Vibration Sequence

Let $l_1, l_2,.., l_n$ be legs, and A be a positive real number. A *leg vibration sequence with amplitude A* is a leg sequence $u = [l_1, l_2,.., l_n]$ that satisfies the following conditions from (7) to (9).

$$\text{For } 1 \leqq i \leqq n\text{-}1 \quad \text{end}(l_i) \leqq \text{start}(l_{i+1}) \tag{7}$$
$$\text{For } 1 \leqq i \leqq n \quad \text{amp}(l_i) \geqq A \tag{8}$$
$$\text{For } 1 \leqq i \leqq n\text{-}1 \quad \text{sign}(l_i) \times \text{sign}(l_{i+1}) < 0 \tag{9}$$

Definition: Frequency of a leg vibration sequence

Let $v = [l_1, l_2,.., l_n]$ be a leg vibration sequence. The *start*, *end*, *length*, *sign*, *amplitude* and *frequency* for a leg vibration sequence v are defined as follows. We denote them by start, end, length, amp and freq respectively:

$$\text{start}(v) \equiv \text{start}(l_1)$$
$$\text{end}(v) \equiv \text{end}(l_n)$$
$$\text{length}(v) \equiv n$$
$$\text{sign}(v) \equiv \text{sign}(l_1)$$
$$\text{amp}(v) \equiv \min_i \text{amp}(l_i) \text{ for } 1 \leqq i \leqq n$$
$$\text{freq}(v) \equiv \text{sign}(v) \times \text{length}(v)$$

Definition: Leg vibration sequence set of a subsequence for amplitude A

Let $X[p:q]$ and A be a subsequence of time series X and amplitude respectively. *Leg vibration sequence set for amplitude A* $V(X[p:q], A)$ is defined by a set of leg vibration sequences $v = [l_1, l_2,.., l_n]$ that satisfy the following conditions from (10) to (12).

$$\text{amp}(v) \geqq A \tag{10}$$
$$p \leqq \text{start}(v) \tag{11}$$
$$\text{end}(v) \leqq q \tag{12}$$

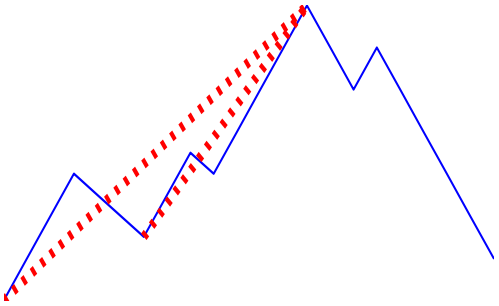

Figure4: Leg

Definition: Leg frequency of a subsequence for amplitude A

Let $V(X[p:q], A)$ be a leg vibration sequence set of a subsequence $X[p:q]$ for an amplitude A. *Leg frequency* $freq_A$ *of a subsequence X[p:q] for A* is defined by below.

$$\text{freq}_A(X[p:q]) \equiv \text{sign}(v_{\max}) \times \text{length}(v_{\max})$$
$$\text{where } v_{\max} = \underset{v \in V(X[p:q],A)}{\text{argmax}}\ \text{length}(v) \tag{13}$$

Leg frequency is well defined because $v_{\max}$ is not unique but the sign of $v_{\max}$s is unique due to the lemma below.

Lemma: Let $V(X[p:q], A)$ be a leg vibration sequence set of a subsequence $X[p:q]$ for amplitude A. The signs of leg vibration sequences that satisfy (13) are the same.

Proof. We assume that $u = [l_1, l_2,.., l_n]$ and $v = [m_1, m_2,.., m_n]$ are leg vibration sequences where both u and v have the maximal length n in $V(X[p:q], A)$ and have different signs. We will show this assumption implies contradiction. Without loss of generality, we can assume that sign(u) is positive and sign(v) is negative; leg l_1 is upward leg and m_1 is downward leg.

Since l_1 and m_1 cross, either one is included by the other, or either one proceeds the other, one of the following conditions is true.

$$\text{start}(l_1) < \text{start}(m_1) < \text{end}(l_1) < \text{end}(m_1) \tag{14}$$
$$\text{start}(m_1) < \text{start}(l_1) < \text{end}(m_1) < \text{end}(l_1) \tag{15}$$
$$\text{start}(l_1) < \text{start}(m_1) < \text{end}(m_1) < \text{end}(l_1) \tag{16}$$
$$\text{start}(m_1) < \text{start}(l_1) < \text{end}(l_1) < \text{end}(m_1) \tag{17}$$
$$\text{start}(l_1) < \text{end}(l_1) \leq \text{start}(m_1) < \text{end}(m_1) \tag{18}$$
$$\text{start}(m_1) < \text{end}(m_1) \leq \text{start}(l_1) < \text{end}(l_1) \tag{19}$$

The definition of leg implies that the above magnitude relations in the formulas from (14) to (17) satisfy not equality but inequality.

First, we will deduce the contradiction when the condition (14) is true. Since l_1 is an upward leg, X[start(m_1)] < X[end(l_1)]. Since m_1 is an downward leg, X[start(m_1)] > X[end(l_1)]. These equations contradict each other. When the condition (15) is true, a proof is in the same way.

Secondary, we will deduce the contradiction when the condition (16) is true. Since amp(m_1) $\geqq A$ and l_1 an upward leg, l_{1_1} = X[start(l_1): start(m_1)] is an upward leg whose amplitude is greater than or equal to A. Therefore, [l_{1_1}, m_1, m_2,.., m_n] is a leg sequence whose length is $n + 1$. It contradicts that v has the maximal length n in $V(X[p:q], A)$. When the condition (17) is true, a proof is in the same way.

Lastly, we will deduce the contradiction when the condition (18) is true. Since [l_1, m_1, m_2,.., m_n] is a leg sequence whose length is $n + 1$. It contradicts that v has the maximal

length n in $V(X[p:q], A)$. When the condition (19) is true, a proof is in the same way.

Therefore, the initial assumption – u and v have different sign – must be false. □

Definition: Leg Frequency $freq_{X,A,W}(t)$

Let X, A, W are time series, amplitude and window size respectively. A *Leg Frequency of X at time t with A and W*, $freq_{X,A,W}(t)$, is defined as follows:

$$freq_{X,A,W}(t) \equiv freq_A(X[t:t+W-1])$$

Leg frequency qualifies the vibration of a time series subsequence, when amplitude and window size are given. Larger amplitude means larger width of variation. The sign of frequency is the sign of the first leg of the longest leg sequence. That is, a positive frequency at time point t with window size W means that the first leg has an upward trend in the subsequence X[t:t+w-1]. Similarly, a negative frequency means that the first leg is downward. A frequency of 2 means that the subsequence has one convex pattern whose amplitude is larger than A, while a frequency of -2 means that it has one concave pattern. If the absolute value of the frequency for a subsequence is larger than 4, we know that the subsequence has at least 4 consecutive up-down trends with the specified amplitude within the specified window size. This rule is often used for detecting vibration over specified amplitude.

Figure 5 shows the leg sequences in the same subsequence for several different amplitudes. The amplitudes are 1 for the top, 2 for the middle, and 4 for the bottom, respectively. The frequencies are 6 for the top, - 5 for the middle and 2 for the bottom respectively.

Let n, W and A be the length of time series, window size and amplitude respectively. The computational order of calculating leg frequency $freq_{X,A,W}$ by an algorithm derived directly from the definition is O($n \times (W!)$), because searching all the possible legs in constructing $V(X[t:t+W-1], A)$ needs the computation whose order is O((W!)) in the worst case. However, we can obtain faster algorithm whose computational order is O($n \times W$) by using left most vibration sequence and the theorem which it satisfies.

Definition: Leftmost Leg Vibration Sequence

Let X and L be a time series and a set of legs whose amplitude are greater than or equal to amplitude A respectively. The *leftmost vibration sequence* in $V(X[p:q], A)$ is a leg sequence $[m_1,.., m_i,\ldots, m_n]$ when m_i is selected repeatedly until we can not select it in the below way:

$$m_1 = \text{argmin}_{l \in L}\ \text{end}(l)$$
$$m_{i+1} = \text{argmin}_{l \in L_i}\ \text{end}(l) \text{ for } \geq 1$$
$$\text{Where } L_i \equiv \{l \in L | \text{start}(l) \geq \text{end}(m_i) \text{and } \text{sign}(l) \cdot \text{sign}(m_i) < 0\}$$

Theorem: The leftmost leg vibration sequence is a leg sequence that has the maximal length in $V(X[p:q], A)$.

Proof. Let u = [l_1, l_2,.., l_n] be the leftmost leg vibration sequence whose length is n. We will reduce contradictory if v = [m_1, m_2,.., m_k] is assumed a maximal leg vibration sequence in $V(X[p:q], A)$ whose length k is greater than n.

First, we will prove that leg l_1 and m_1 have the same sign. Let us suppose that leg l_1 and m_1 have different sign. Since u is leftmost, we can prove [l_1, m_1, m_2,.., m_k] is a leg vibration sequence whose length is m + 1 by exploiting the reasoning

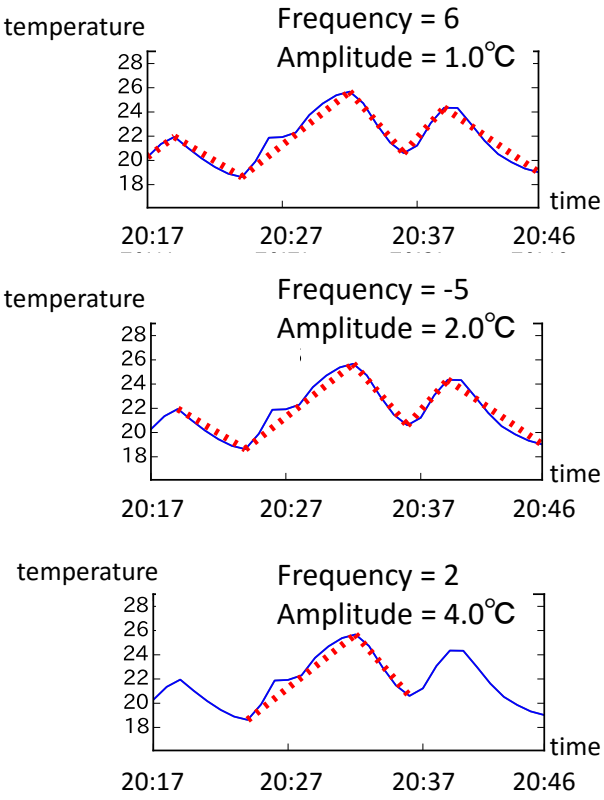

Figure 5: Trend Graphs of Experimental Data

method in the previous lemma. It contradicts the length of v is maximal. Therefore, the assumption – l_1 and m_1 have different sign – must be false.

Since l_1 and m_1 have the same sign and u is leftmost, it is true that end(l_1) ≤ end(m_1) ≤ start(m_2). Therefore [l_1, m_2,.., m_k] is a leg vibration sequence. In the same way, [l_1, l_2, m_3,.., m_k] is a leg vibration sequence. Since $k > n$, [l_1, l_2,.., l_n, m_{n+1},.., m_k] is a leg vibration sequence where the amplitude of each leg is greater than or equal to A. Therefore, a subsequence X[end(l_n) : end(m_k)] must include the leftmost leg l_{n+1} whose amplitude is greater than or equal to A and whose sign is the same as that of m_{n+1}, that is, different from that of l_n. This means that [l_1, l_2,.., l_n, l_{n+1}] is the leftmost vibration sequence. It contradict that u is the leftmost vibration sequence.

Therefore, the initial assumption – the length of v is greater than that of u – must be false. □

The computational order of constructing leftmost leg vibration sequence is O(W), therefore that of leg frequency $freq_{X,A,W}$ is O($n \times W$).

5 EVALUATION

We have developed a TPQL based anomaly detection system and applied it to a practical application system. With regard to development efficiency, we compared the number of lines of programs by TPQL to those by Java. We confirmed that the convolution operation for describing feature extraction for sliding windows can reduce about 100 lines to 1 line. However, enough evaluation of development efficiency for constructing real applications remains as a future

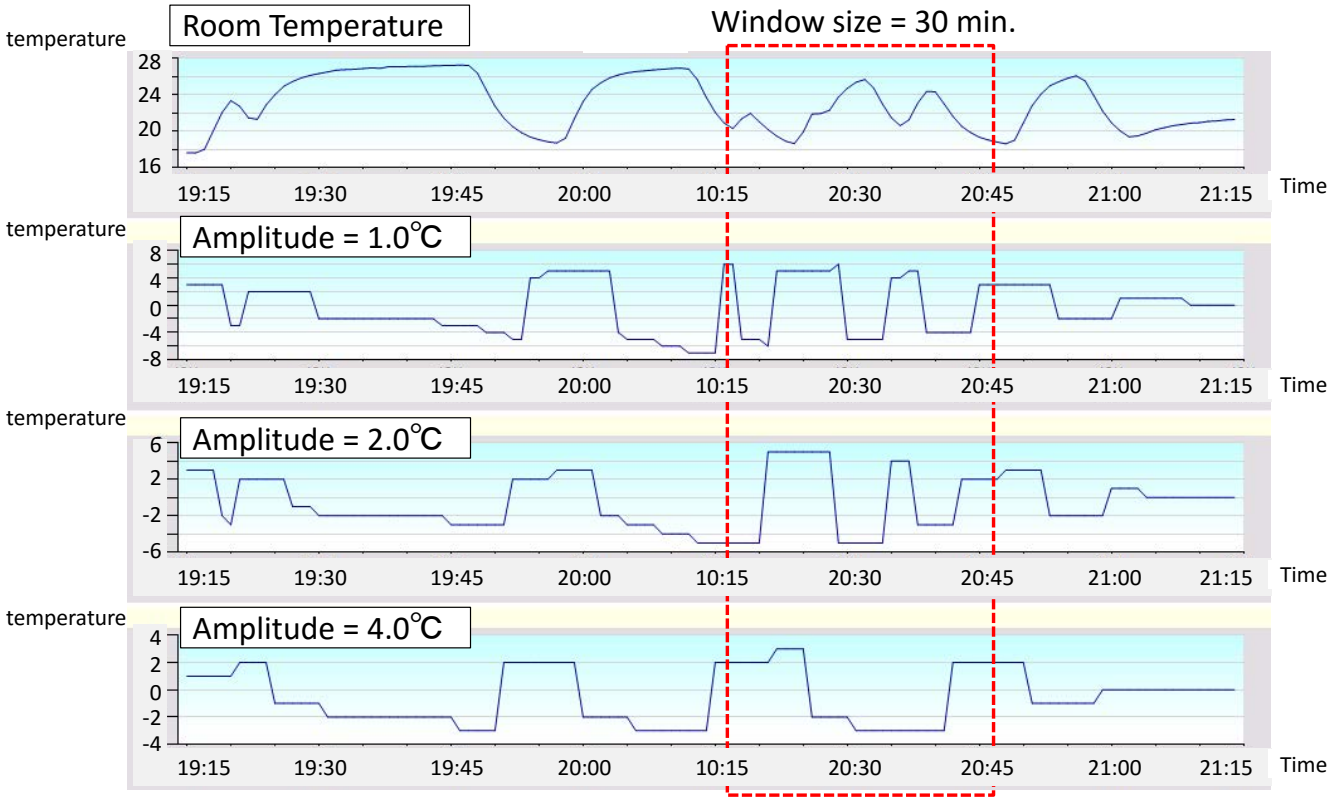

Figure 6: Trend Graphs of Experimental Data

work. The rest of this section shows the result of processing time.

(1) TPQL

We confirmed that our system satisfies the processing time that are required by our real application. Our requirement of ECM for buildings is that the processing must be completed within one day for the following conditions:

- 3 anomaly detection scenarios for each signal
- 5,000 signals with sampling period one minute in half a year

The detail results and the conditions in the experiment are described in [4].

(2) Exemplar Learning

We compared our algorithm to the simple yet effective Brute Force Euclidian Distance (BFED) algorithm [14] which has proven to be the most accurate over a variety of different testing time series [15]. Data for the experiment are 24 data sets that are available from the paper [16] and 2 synthesized data sets. The result shows that our algorithm is about from 5 times to 100 times faster than the BFED algorithm without losing accuracy. The detail results and the conditions in the experiment are described in [5].

(3) Leg vibration analysis

We applied leg vibration analysis to anomaly detection for HVAC (Heating Ventilation, and Air Conditioning) systems. It is known that a significant variation of room temperature during operation often shows an anomaly in the control and/or sensor system.

The data for the experiment consisted of room temperature readings with sampling period one minute, for a total duration of three years. The total number of time points is thus 1,578,239. We obtain leg frequencies with window size 30 minutes and with amplitude 1.0℃, 2.0℃, and 4.0℃, respectively. A higher amplitude means a higher warning level. Leg vibration analysis can enable adaptive monitoring by selecting an appropriate window size and amplitude.

Our leg vibration analysis software detected 1901 points (0.12%), 454points (0.029%), and 69 points (0.0044%) for amplitudes of 1.0℃, 2.0℃, and 4.0℃, respectively. Fig. 4 shows a snapshot of leg frequencies as a function of time for each amplitude. The top, the middle, and the bottom graphs correspond to amplitudes of 1.0℃, 2.0℃, and 4.0℃, respectively. Fig. 5 above shows the leg sequences for the subsequences that are surrounded by the rectangular area in Fig. 6.

The processing times for amplitudes 1.0℃, 2.0℃, and 4.0℃ were 0.612 sec., 0.554sec., and 0.489 sec. respectively. We set the threshold on the absolute value of leg frequency to 4, in order to detect anomalies for each temperature. All computational times satisfy the requirements of our application. The detail results and the conditions in the experiment are described in [6].

6 CONCLUSIONS

This paper describes a TPQL-based anomaly detection system whose main features are convolution operation, time interval join, data transformation, exemplar learning and leg vibration analysis.

Our efforts have been mainly focused on feature extraction and machine learning based anomaly detection, that is, a data-driven approach. A data-driven approach has a problem that it can detect the differences from the ordinary behavior of sensor data, but it cannot distinguish the symptoms of failures from only unusual behavior. Our future direction is to develop a hybrid method to combine a data-driven ap-

proach with a physical model-based approach with equipment domain knowledge in order to explain whether anomalous behavior is actually a symptom of a failure.

REFERENCES

[1] J. Zheng, D. Simplot-Ryl, C. Bisdikian, H.T.Mouftah: "The Internet of Things [Guest Editorial]," Communications Magazine, IEEE , Vol.49, No.11, pp.30-31 (2011).

[2] M. Imamura, D. Nikovski, Z. Sahinoglu, M. Jones: "A Survey on Machine Learning for Equipment Condition Monitoring Using Sensor Big Data," IIEEJ Transactions on Image Electronics and Visual Computing Vol.2 No.2, pp. 112-121 (2014).

[3] M. Imamura, S. Takayama, and T. Munaka: "A stream query language TPQL for anomaly detection in facility management," The 16th International Database Engineering & Applications Symposium (IDEAS '12), pp. 235-238 (2012).

[4] M. Imamura,T. Takeuchi, S. Kitagami, M. Kanno, T. Munaka: "Time Series Data Query Language TPQL for anomaly detection in facility," Journal C of Electronics and Communications in Japan,Vol.134, No.1, pp. 156-167 (2014). (in Japanese).

[5] M. Jones and D. Nikovski and M. Imamura and T. Hirata: "Exemplar Learning for Extremely Efficient Anomaly Detection in Real Valued Time Series," Data Mining and Knowledge Discovery (DAMI), First online: 25, pp 1-28 (2016).

[6] M. Imamura, T. Nakamura, H. Shibata, N. Hirai, S. Kitagami, T. Munaka: "Leg Vibration Analysis for Time Series," IPSJ Journal, Vol. 57, No.4, pp.1303-1318 (2016). (in Japanese).

[7] R. T. Snodgrass (Ed.): The TSQL2 Temporal Query Language. Kluwer (1995).

[8] A. Arasu, S. Babu, J. Widom: "The CQL continuous query language: semantic foundations and query execution," The International Journal on Very Large Data Bases archive, Vol. 15, No. 2, (2006).

[9] V. Chandola, A. Banerjee, V. Kumar: "Anomaly detection: a survey," ACM Comput Survey, Vol. 41, No. 3 (2009).

[10] W. Yan: "Feature Engineering for PHM applications," The 7th Annual Conference of the Prognostics and Health Management Society, http://www.phmsociety.org/sites/phmsociety.org/files/FeatureEngineeringTutorial_2015PHM_V2.pdf (2015).

[11] T. Endo, M. Matsuishi, K. Kounaga, K. Kobayashi, K. Takahashi: "Rain Flow Method and Its Application," Research Report of Kyushu Institute of Technology, http://www-it.jwes.or.jp/qa/details.jsp?pg_no=0040020170 (1974).

[12] I. Rychlik: "A new definition of the rainflow cycle counting method," International journal of fatigue Vol. 9. No. 2, pp. 119-121 (1987).

[13] E. Fin, B. P. Kevin: "Indexing of Compressed Time series," DATA MINING IN TIME SERIES DATABASES, World Scientific, pp. 43-65 (2004).

[14] T. RakthanmanonT, B. Campana B, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, E. Keogh Searching and mining trillions of time series subsequences under dynamic time warping. 18th ACM SIGKDD international conference on knowledge discovery and data mining, pp 262–270 (2012).

[15] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh: "Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures," VLDB 2008, pp.1542-1552 (2008).

[16] E. Keogh, J. Lin, A. Fu: "HOT SAX: finding the most unusual time series subsequence: algorithms and applications," The Fifth IEEE international conference on data mining, pp. 226–233, www.cs.ucr.edu/eamonn/discords/ (2005).

(Received September,30,2015)
(Revised June 10,2016)

Makoto Imamura He received a M.E. degree from Kyoto University of Applied Mathematics and Physics in 1986 and a Ph.D. degree from Osaka University of the Information Science and Technology in 2008. He is a professor of the school of Information and Telecommunication Engineering at Tokai University. He had been worked as a research staff at the Information Technology R&D Center in Mitsubishi Electric Corporation until 2015. He has worked on datamining methods for prognostics and health management and cyber-physical production system.

Daniel Nikovski He received a PhD in robotics from Carnegie Mellon University in 2002, and is presently a senior member of research staff and group manager of the Data Analytics group at Mitsubishi Electric Research Laboratories. He has worked on probabilistic methods for reasoning, learning, planning, and scheduling, and their applications to hard industrial problems. He is a member of IEEE.

Michael Jones He received a Ph.D. degree from the Electrical Engineering and Computer Science department of the Massachusetts Institute of Technology (MIT) in 1997, and is currently a senior principal member of the research staff at Mitsubishi Electric Research Laboratories. He has worked mainly on problems in computer vision, but recently has also focused on time series analysis.