IET Wireless Sensor Systems, Vol. 6, No. 1, 2016, pp. 26-33 (DOI 10.1049/iet-wss.2014.0090).

A Wireless Sensor Network-based Pattern Matching Technique for the Circumvention of Environmental and Stimuli-related Variability in Structural Health Monitoring

William Contreras[1] and Sotirios Ziavras[2]
New Jersey Institute of Technology
University Heights
Newark, N.J. 07102, USA
[1]wc42@njit.edu, [2]ziavras@njit.edu

Abstract
Much research has gone into using wireless sensor networks to monitor structural health by sensing and measuring vibrations. One problem here is that structural vibrations can be affected by many factors, which can make it difficult to determine the contribution of structural condition to measured vibrations. We propose a robust solution to this difficulty that consists of a wireless sensor network that implements a highly efficient, fully distributed pattern matching algorithm. Here, we exploit correlation between sensed vibration signals at different locations on the structure under measurement to detect damage. Potential applications for our system are numerous. They include many infrastructure applications such as those involving railroad and pipeline monitoring. The general solution is described, including tradeoffs between accuracy and energy/memory consumption. It is shown that the accuracy of our approach grows gracefully at the expense of memory and energy consumption. Additionally, a case study involving railroad applications is discussed. Simulations in our case study indicate that our distributed approach can reduce the consumed energy for transmitted data by 50% compared to a centralized architecture.

1. Introduction

Much research has been done regarding using wireless sensor networks (WSNs) to do structural health monitoring (SHM) on engineering structures. Most commonly, SHM is performed by analyzing structural vibrations [1]. The problem with this is that vibrations can be affected by many factors, not just the condition of the structure (factors other than structural condition will herein be referred to as ‘extraneous factors’). Environmental factors such as temperature and humidity are examples. Also factors related to the stimuli that produce the vibrations being monitored, such as the magnitude and placement of the stimuli, affect the vibrations. These factors all contribute to the vibrations and make it difficult to determine the contribution of structural condition to the vibrations [2, 3].

Our solution to this problem consists of a WSN with motes (i.e., sensors nodes with limited computational resources) distributed throughout the monitored structure. Each mote measures vibrations and uses the data to compute a relevant statistical feature. In this paper, the term statistical feature is used to refer to a quantity calculated from the raw vibration data that provides useful information about the data as a whole, such as a time domain average. Once a particular mote computes the value of the statistical feature, it maps the value into a discrete index value. The motes share their index values amongst themselves to form tuples, or patterns, of index values. The resulting patterns are checked against a reference list of 'healthy' patterns. These healthy patterns are determined during a training phase that takes place immediately after the system is installed. If a match is not found, potential damage is flagged.

Overall, our system uses a statistical approach to flag conditions that are abnormal with respect to operating conditions observed during the system's training phase. It is not intended to determine the exact condition of a structure (e.g. 50% reduction in thickness of a particular structural element). If the system is installed on new construction, a flag serves to indicate that the structure is transitioning to an unhealthy state. If it is installed on a structure that already has damage, a flag would serve to indicate that the condition of the structure is growing worse. By employing our statistical approach, we are able to achieve the following three goals: (1) an efficient, computationally simple solution suitable for implementation in a WSN; the structure does not have to be mathematically modeled and the WSN system does not have to solve equations. (2) a portable solution, independent of structure that is easy to deploy. (3) a solution that does not require the values of involved parameters to be input to the system. Additionally, our system has many potential applications and is especially well suited for structures that carry live loads in a linear fashion, such as pipelines (e.g. water or gas) and railroads.

2. Previous Work

The problem of environmental and operational variability has received a good amount of attention in research. Approaches to the problem can be divided into two main categories: input-output and output-only. With input-output approaches, in determining structural condition, measured values of extraneous parameters are input to regression models that describe the relationship between structural response and extraneous parameters. The disadvantage of this approach is that it is often difficult to establish good regression models. With output-only approaches, statistical methods are used to determine structural condition without measuring the extraneous factors. Our method is of the output-only type.

Many different output-only methods have been proposed. One technique that has been investigated is the usage of factor analysis [4, 5]. Another popular approach is the usage of principal component analysis (PCA) [6-8]. The authors of [9] use PCA to perform SHM on wind turbines. In [10], the authors use adaptive PCA to continually update the PCA model with new data to improve the performance of the system. In [11], to compensate for environmental and operational variability, an output-only method involving regression analysis is proposed. The authors of [12] suggest using a spatio-temporal infinite impulse response to filter out the environmental and operational effects from damage estimates. Overall, these approaches are promising with regard to achieving immunity to environmental and stimuli-related variability but they tend to be computationally intensive.

More efficient output-only methods suitable for implementation in WSNs have also been proposed. The authors of [13] use the efficient Goertzel algorithm to compute transmissibility. The authors of [14] use an autoregressive process model to fit vibration data to sets of coefficients for different environmental and operational conditions. In [15-17], WSNs are used to perform modal analysis on structures. These approaches have the potential to achieve computational efficiency, however the extent to which they ameliorate the problem of environmental and operational variability is unclear. Also, regarding modal analysis in particular, an accurate synchronization of the motes in a network is required at the onset of data collection [18]. Our approach is both computationally efficient and demonstrably effective at circumventing variability due to extraneous parameters.

3. Proposed Method

In our method, each of the motes measures vibrations and uses the data to calculate a statistical feature. Overall it is helpful to use the simplest feature possible to impose as little burden as possible on the WSN motes. We emphasize the usage of time-domain features in this paper, and the average absolute deviation (AAD) feature in particular. The general formula for computing AAD can be seen in Eq 1. In the equation 'a_b' is the value, at a particular instant in time, of the parameter being measured, 'a_M' is a DC offset, and 'T' is the total number of samples over which the average is to be performed.

$$AAD = \frac{1}{T}\sum_{b=1}^{T}|a_b - a_M| \qquad \textbf{Eq 1.}$$

The computation of AAD requires relatively little energy, as it can be computed primarily in an online fashion and involves only simple operations. It is also of substantial use as it is sensitive to many different types of damage. One important type of damage that it is sensitive to is corrosion: that is, the gradual wearing-away of structural material. When an element of a structure corrodes it loses thickness, and as such its elastic properties change. In general, as thickness decreases the elastic modulus, which is defined as stress per unit strain, decreases; that is it takes less stress, which is defined to be force per unit area, to produce the same strain in the element. Given this, for a particular vibration inducing-stimulus, we would expect the amplitude of the vibrations in an element to increase as corrosion sets in. As such, the AAD of the vibrations should increase.

The AAD feature is sensitive not only to damage (corrosion and other forms) but also to extraneous factors. Temperature, for instance, can change the elastic properties of structural elements and thus the amplitude of their vibrations for a given stimulus. Changes in the intensity of the stimulus producing the vibrations can also change the intensity of the vibrations and thus the value of the AAD feature. For instance in the case of a railroad bridge, the intensity of vibrations in the bridge deck at a location over which the train is moving will increase with train weight.

Because extraneous factors tend to be random in nature, the statistical features that they effect may be treated as random variables [19]. We focus on the AAD feature in particular in this paper. Assume now a group of N motes distributed about a particular structure. Each mote measures vibrations and calculates the AAD of the data. As such, there is an AAD random variable associated with each mote. The AAD random variables will be denoted 'x_i,' where 'i' is the number of the mote. The set of AAD random variables may be expressed as a vector, as seen in Eq 2.

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \qquad \textbf{Eq 2.}$$

Assume now that the random variables in the '$\boldsymbol{x}$' vector are correlated. That is, assume that none of the elements of the covariance matrix associated with the '$\boldsymbol{x}$' vector are equal to zero. This is a valid assumption in many cases, and is especially true when the motes are in spatial proximity to each other [20-22]. The elements of the covariance matrix can be found using Eq 3.

$$\boldsymbol{C}_{i,j} = cov(x_i, x_j) \qquad \textbf{Eq 3.}$$

When the structure is healthy, the '$\boldsymbol{x}$' random variables follow a particular probability distribution. The probability density function (PDF) for this distribution will be denoted $p_h(\boldsymbol{x})$. There is a good chance, for instance, that the random variables will follow (at least approximately) a multivariate Gaussian distribution [19]. The PDF for an arbitrary multivariate Gaussian distribution can be seen in Eq 4. In the equation '$\boldsymbol{\mu}$' is the mean vector corresponding to the '$\boldsymbol{x}$' vector.

$$p_g(\boldsymbol{x}) = \frac{1}{2\pi^{N/2}\sqrt{\det(\boldsymbol{C})}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{C}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \qquad \textbf{Eq 4.}$$

An illustration of 1000 outcomes from a sample healthy distribution for a two mote system can be seen in Figure 1. For the figure, $p_h(\boldsymbol{x}) = p_g(\boldsymbol{x}),\ \boldsymbol{x} = [x_1, x_2]$. Note the dense concentration of outcomes in the middle of the plot.

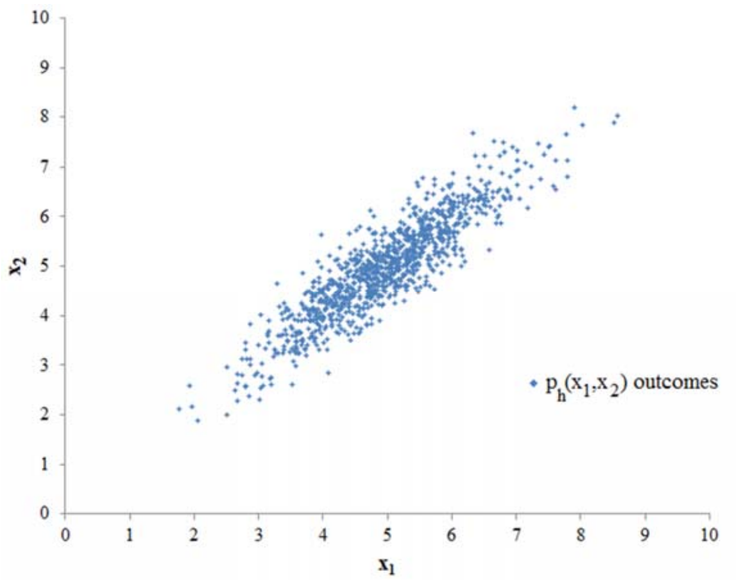

Figure 1. 1000 outcomes from a sample healthy distribution for a two mote system. In the figure $p_h(x_1, x_2) = p_g(x_1, x_2)$. For $p_g(x_1, x_2)$, the following values were used for the mean vector: $\boldsymbol{\mu} = [\mu_1 = 5, \mu_2 = 5]$. Also, the covariance matrix was as follows: $\boldsymbol{C} = [1, 0.9;\ 0.9, 1]$.

For a general '$\boldsymbol{x}$' vector, the set of N AAD random variables of the vector can be mapped into an N-dimensional space, where a point is identified as an N-tuple consisting of the values of the N random variables. For instance in Figure 1, the random variables x_1 and x_2 are mapped into a two dimensional space. Here a point is identified as a 2-tuple consisting of the values of the random variables as follows: (x_1, x_2).

For correlated random variables, the outcomes of a healthy distribution will generally fall within a particular region of the N-dimensional space formed by the N random variables. This region will be denoted 'S'. This is exemplified in Figure 1 where the outcomes of $p_h(x_1, x_2)$ are concentrated toward the middle of the plot. When damage occurs around one of the N motes, the distribution shifts [19]. The distribution that results will be denoted $p_d(\boldsymbol{x})$. When this happens, outcomes occur outside of the region 'S' more frequently than with the healthy distribution, $p_h(\boldsymbol{x})$. This is illustrated in Figure 2.

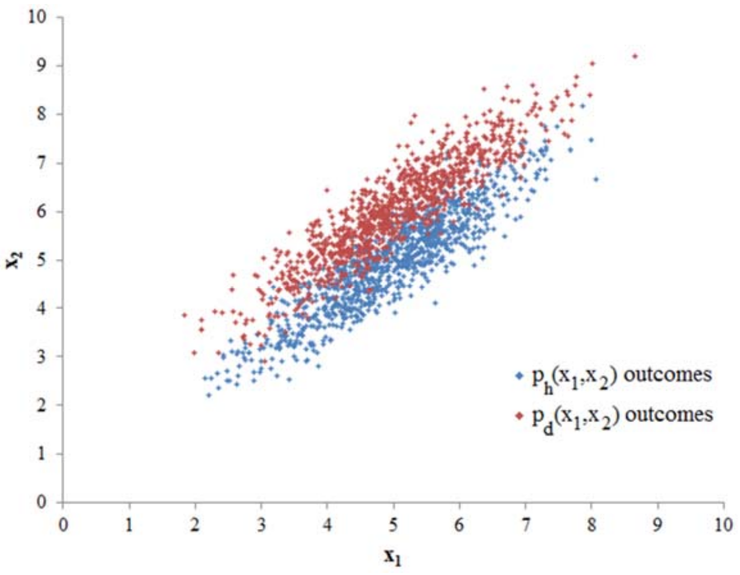

Figure 2. 1000 outcomes from the distribution of Figure 1 along with 1000 outcomes from a sample damage distribution $p_d(\boldsymbol{x})$. The damage outcomes were generated by shifting the mean of the x_2 random variable as follows: $p_d(x_1, x_2) = p_h(x_1, x_2 + c),\ c = 1$.

Our method calls for the N-dimensional space associated with the N AAD random variables to be divided into N-dimensional blocks. This can be done by mapping the values of each random variable x_i into a discrete set of index values. In doing so, an index value random variable is created for each x_i variable. The index value random variables may be expressed as a vector, as seen in Eq 5. One way to map the x_i variables into index value variables is to use the floor function, as seen in Eq 6. The simplest way to perform the mapping is to do so in a linear fashion, with a linear $g(x)$, as seen in Eq 7.

$$\boldsymbol{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{bmatrix} \qquad \textbf{Eq 5.}$$

$$v_i = \lfloor g_i(x_i) \rfloor \qquad \textbf{Eq 6.}$$

$$v_i = \left\lfloor \frac{x_i}{Y} \right\rfloor \qquad \textbf{Eq 7.}$$

The N-dimensional space associated with the N v_i random variables is a discrete mapping of the continuous N-dimensional space associated with the N AAD random variables. The space associated with the v_i variables consists of blocks of points from the AAD variable space indexed by N-tuples, or patterns, of index values. An illustration of this for a two mote system can be seen in Figure 3. In the figure, x_1 and x_2 are mapped into v_1 and v_2, respectively, using Eq 7 with $Y = 2$.

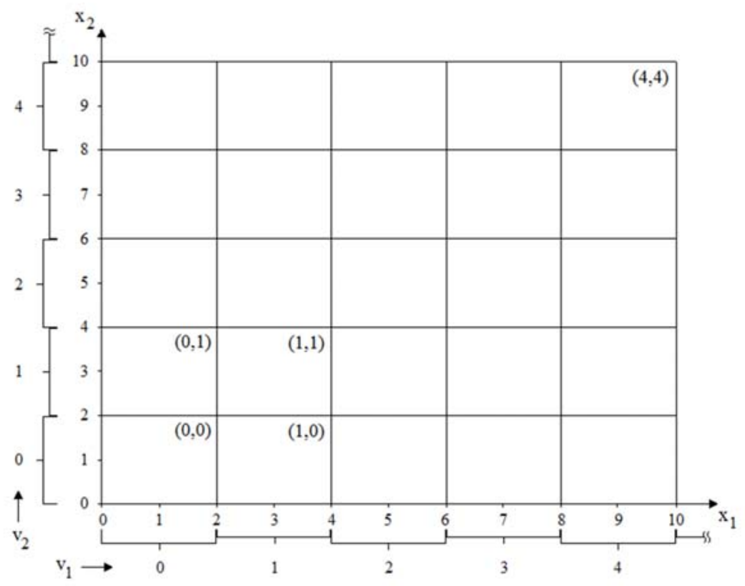

Figure 3. Illustration of the discretization of the space associated with the AAD random variables for a two mote system. The variables x_1 and x_2 are mapped into v_1 and v_2, respectively, using Eq 7 with $Y = 2$. Each block of points is indexed by a 2-tuple consisting of the values of v_1 and v_2. Examples of 2-tuples appear in parentheses within several of the blocks. They are formatted as follows: (v_1, v_2).

Our method calls for two primary phases: a training phase and an operational phase. During the training phase, the N motes obtain a number of data points, where each data point consists of an AAD value from each one of the motes. To generate a data point each mote collects vibration data, calculates the AAD of the data, and sends its AAD value to a base station. The timing of the data collection between the motes depends upon the application and is arranged to achieve as high a correlation between the AAD random variables as possible. Once the base station collects all of the AAD values for a particular data point, it takes each mote’s AAD value and determines the corresponding index value using each mote’s mapping function, $g_i(x_i)$ (Eq 6.). It combines the index values to form an N-tuple of index values and thus determine which block the data point falls into. It then stores the N-tuple in memory. The base station collects many data points and thus builds up a large database of N-tuples corresponding to the healthy structure, thereby, in terms of blocks, establishing the healthy behavior of the structure. An illustration of this process for a two mote system can be seen in Figure 4.

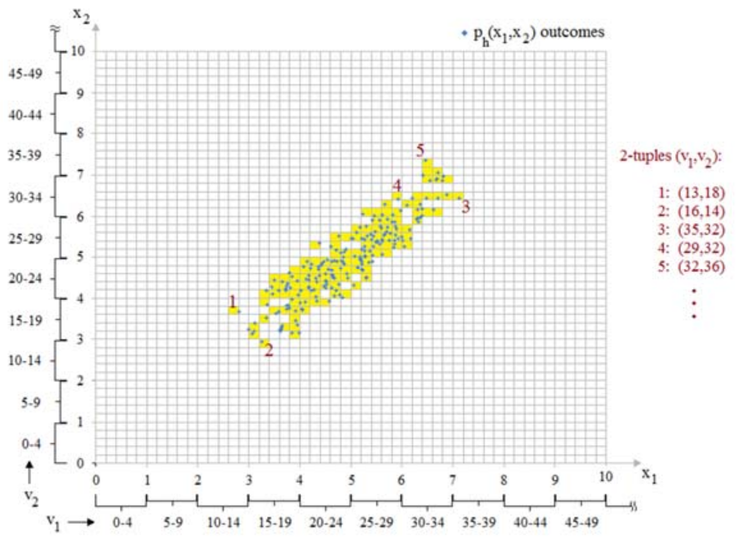

Figure 4. Illustration of the training phase process for a two mote system. Outcomes from a healthy distribution are indicated by blue markers. The PDF for the healthy distribution, $p_h(x_1, x_2)$, was the same as that used for Figure 1. The AAD variables x_1 and x_2 were mapped into v_1 and v_2 using Eq 7 with $Y = 0.2$. Yellow colored blocks indicate blocks in which outcomes fall.

From the N-tuple database for the N motes, subset databases can be obtained for subsets of the motes. For example, for the system of Figure 4, a database of 1-tuples could be generated for the mote associated with the x_1 random variable by simply deleting the v_2 values from the entries of the 2-tuple database. We use this to implement the operational phase of our system in a distributed manner. During the operational phase, to assess the condition of a structure, the motes must (in the aggregate) collect a data point, determine which block it falls into, and then determine whether or not the corresponding N-tuple is one recorded by the base station during the training phase. If it is not, damage is indicated. To accomplish this in a distributed fashion, among the N motes we define a transmission chain. Each mote is assigned to a particular link in the chain. The N motes can be denoted 'm_1' through 'm_N'. Mote m_1 is defined to be the first mote in the transmission chain, and mote m_N is defined to be the last mote in the transmission train. Each mote contains a subset of the base station's N-tuple database that corresponds to the subset of motes consisting of itself and its predecessors in the transmission chain.

Pseudocode for the operational phase algorithm can be seen in Pseudocode 1. In the pseudocode, variables with overbars are vectors. For each data point each mote collects vibration data, calculates the AAD of the data, and then maps the AAD value to an index value using its designated mapping function, $g_i(x_i)$. These operations correspond to lines 8 through 10 in Pseudocode 1. The mote then combines its index value with those of any predecessors in the transmission chain to form a tuple of index values, and checks the tuple against those in its database to determine if it is present. This process corresponds to lines 11 through 18 in Pseudocode 1. If the tuple is not present, the mote immediately sends an alert to the base station indicating that damage might be present. The action taken by the mote based upon whether or not it finds the tuple in its database is represented by lines 19 through 25 in Pseudocode 1. In Pseudocode 1, one iteration of the while loop corresponds to one data point. It should be noted that, assuming no damage is detected, the base station does not have to do any processing. In the operational phase, the only role of the base station is to process alerts from the motes when they detect an abnormal condition. For instance, the base station might relay an alert over the internet to the engineering office in charge of the structure on which the system is installed.

Pseudocode 1 Algorithm for Operational Phase

```
1:   i ≡ index of link in transmission chain
2:   N ≡ number of motes in the system (as defined earlier)
3:   $m_i$ ≡ mote with link index *i*
4:   $D_i$ ≡ database for mote $m_i$
5:   tuple ≡ pattern of index values (as defined earlier)
6:   **while** TRUE
7:       **for** i=1:N
8:           $\overline{\text{data}}$ = *collect_vibration_data*();
9:           AAD = $\frac{1}{T}\sum_{b=1}^{T}\left|\overline{\text{data}}(b) - a_M\right|$      //from Eq 1
10:          index_value = $\lfloor g_i(\text{AAD}) \rfloor$;     //from Eq 6
11:          **if** i = 1
12:              tuple = index_value;
13:              result = *check_$D_i$_for_tuple*(tuple);
14:          **else**
15:              $\overline{\text{received}}$ = *wait_for_tuple_from_ $m_{i-1}$*();
16:              tuple = *concatenate_tuples*(index_value, $\overline{\text{received}}$);
17:              result = *check_$D_i$_for_tuple*(tuple);
18:          **end**
19:          **if** result = FOUND
```

```
20:             if i < N
21:                 transmit_tuple_to_m_{i+1}(tuple);
22:             end
23:         else
24:             transmit_alert_to_base_station();
25:         end
26:     end
27:     wait_for_stimulus();
28: end
```

When monitoring an entire structure, with motes distributed throughout the structure, it is unlikely that all of the associated AAD random variables will be strongly correlated with each other. In this case, the motes can be divided into several highly correlated N-mote groups. Here, a cluster topology such as that proposed in [23] can be implemented, where each cluster independently runs an instance of our method. In general, the motes in each cluster can be connected using a ring topology. During the training phase the cluster head can be elected using algorithms such as those discussed in [24].

4. Computational Complexity

The operational phase of our algorithm is to be implemented in a distributed fashion, on the network's motes. As described, each of the motes must check a tuple obtained through measurement against a reference list of healthy tuples. In Pseudocode 1, this process is represented by the '*check_D$_i$_for_tuple*' function. Assuming that, during the training phase, the base station sorts the lists of healthy tuples before storing them on the motes, the process of checking a list of healthy tuples for a measured tuple amounts to searching a sorted array. As such, the '*check_D$_i$_for_tuple*' function can be implemented with a binary search algorithm. On any given mote, the binary search algorithm has a worst-case run time of $\lceil \log_2(p_i + 1) \rceil$, where p_i is the size of the list of healthy tuples on mote m_i (the size of D_i). Since p_i tends to increase exponentially as 'i' increases, the worst-case run time for each individual mote increases as 'i' increases. The worst case run-time for the algorithm of Pseudocode 1 as a whole is equal to the sum of the worst-case run times for the individual motes, as seen in Eq 8.

$$T = \sum_{i=1}^{N} \lceil \log_2(p_i + 1) \rceil \qquad \textbf{Eq 8.}$$

By making some simplifying assumptions, a closed form solution to the worst-case run time for the algorithm of Pseudocode 1 can be determined. Firstly, if each value of 'p_i' is a power of two, the worst-case run time for a given mote reduces to $\log_2(p_i) + 1$. The 'p_i' values can be made to be powers of two by padding the D_i databases. Secondly, it should be assumed that 'p_i' increases with 'i' in the manner seen in Eq 9. This can be achieved (at least approximately) in most cases by using Eq 6 such to realize Eq 9. That is, subject to other constraints to be introduced shortly, blocks can be generated using Eq 6 such to make Eq 9 true. Given these assumptions the worst case run time for the algorithm of Pseudocode 1 can be found from Eq 10.

$$p_i = {p_1}^i \qquad \textbf{Eq 9.}$$

$$\begin{aligned} T &= \sum_{i=1}^{N} \left(\log_2\left({p_1}^i\right) + 1\right) \\ &= \log_2(p_1) \sum_{i=1}^{N} i + N \\ &= \frac{\log_2(p_1)}{2} N^2 + \left(\frac{\log_2(p_1)}{2} + 1\right) N \qquad \textbf{Eq 10.} \end{aligned}$$

In Eq 10, it can be seen that the worst-case run time of the algorithm grows quadratically with the number of motes in the system. As such, in terms of 'N', the worst case performance of the algorithm may be expressed as $O(N^2)$. It can also be seen from Eq 10 that the run time increases logarithmically with 'p_1'. In terms of 'p_1', the worst case performance of the system can be expressed as $O(\log(p_1))$.

Overall, the performance of the algorithm decreases as the number of motes in the system increases. It also decreases as the number of healthy tuples increases. One way to optimize performance is to keep the number of tuples as small as possible. This can be done by using non-linear mapping functions (non-linear $g_i(x_i)$) for the motes. The idea here is to achieve large blocks in the space where

$p_h(\boldsymbol{x})$ has its highest density (the space 'S') and smaller blocks elsewhere. This helps to reduce the number of blocks and thus the number of healthy tuples. To do this, each mapping function $g_i(x_i)$ should be developed based upon the marginal PDF of the random variable that it is mapping. In particular, in terms of the value of the random variable being mapped, the minimum value of the derivative of the mapping function, $g_i'(x_i)$, should correspond to the maximum value of the PDF. This is the case because we want to achieve the fewest index values per AAD value (largest blocks) where the random variable has its highest density. As the PDF decreases on either side of its maximum value, the value of the derivative of the mapping function should increase.

An illustration of blocks obtained through non-linear mapping for a two mote system can be seen in Figure 5. For the system the healthy distribution used for Figure 1, Figure 2, and Figure 4 was used. Recall that a bivariate Gaussian PDF was used for $p_h(x_1, x_2)$: $p_h(x_1, x_2) = p_g(x_1, x_2)$. With a multivariate Gaussian PDF, the marginal PDF for each random variable is a Gaussian distribution. As such, graphically speaking, for each random variable 'x_i' a concave function centered about the variable's mean should be used for the derivative of the mapping function. The simplest function to use here is a quadratic function of the form seen in Eq 11. In this equation, the constants 'm' and 's' are used to tune the mapping function to achieve an appropriate block size. The corresponding mapping function (the integral of $g_i'(x_i)$) can be seen in Eq 12. In Eq 12, the value of 'c' is determined based upon the constraint that $v_i = 0$ ('0' index value) should correspond to $x_i = 0$ ('0' AAD value): $g_i(x_i = 0) = 0$.

$$g_i'(x_i) = \frac{(x_i - \mu_i)^2}{m} + s, \qquad \textbf{Eq 11.}$$

$$g_i(x_i) = \frac{(x_i - \mu_i)^3}{3m} + sx + c, \quad c = \frac{{\mu_i}^3}{3m} \qquad \textbf{Eq 12.}$$

To generate the blocks of Figure 5, Eq 12 was used to map the AAD random variables into index value random variables. In particular, the following values were used for 'm' and 's' for both $g_1(x_1)$ and $g_2(x_2)$: $m = 2$, $s = 1$. From $p_g(x_1, x_2)$, μ_1 and μ_2 were both equal to five. In the figure, vertical lines indicate index values associated with x_1 and horizontal lines indicate index values associated with x_2.

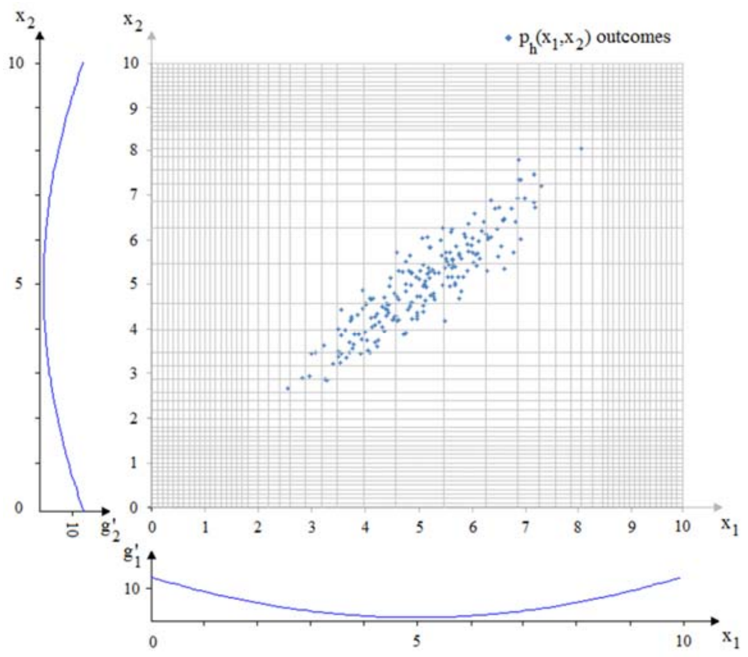

Figure 5. Illustration of blocks obtained through non-linear mapping for a two mote system. Random variables x_1 and x_2 were mapped into index values using Eq 12, using $m = 2$ and $s = 1$. Vertical lines indicate index values associated with x_1 and horizontal lines indicate index values associated with x_2. The derivatives of the mapping functions are shown next to the two axes. Note that block size gets larger as the derivatives approach their minimum values, where the distribution of outcomes is the densest.

Using MATLAB, for the two mote system of Figure 5, we performed a calculation to show the effect of 'm' on the number of tuples recorded during the training phase. Here, we first generated 200 sample outcomes of $p_g(x_1, x_2)$. Based upon the outcomes, we then determined the number of tuples that would be recorded if the system were to use linear mapping, as described by Eq 7. Here, we used $Y = 0.2$. We then determined, also based upon the outcomes, the number of tuples that would be recorded if the system were to use non-linear mapping using Eq 12, for several different values of 'm.' Here, we used $s = 1$. The results of our calculations are summarized in Table 1.

Mapping Technique	Linear, Y=0.2	Quadratic, m=0.1	Quadratic, m=0.5	Quadratic, m=1	Quadratic, m=1.5	Quadratic, m=2
Number of Tuples	112	81	52	37	29	25

Table 1. Number of tuples for several different values of 'm' for the two mote system of Figure 5. The number of tuples for a linear mapping with $Y = 0.2$ is also shown.

The results of Table 1 show that the number of tuples that must be stored decreases as 'm' gets larger. This is because, as 'm' is made larger, block size is being strategically increased in the areas where the distribution of outcomes is the most dense. The reduction in tuples comes at a cost, however. In general, when using non-linear mapping, there is a tradeoff between accuracy and memory/energy consumption. As blocks are made bigger, the number of tuples that must be stored in memory decreases. Additionally, the amount of energy that the motes must spend searching their databases, on average, decreases. However, the potential for false negatives increases. That is, it becomes more likely that a data point from a damaged structure (one that falls outside of the region 'S') will fall inside a healthy block and thus go undetected. Accuracy can be improved by decreasing block size. As block size approaches zero, the likelihood that a data point outside of the region 'S' will fall inside a healthy block also approaches zero. Accuracy, in this regard, approaches 100%.

5. Hardware and Software

We used Zolertia's Z1 mote for our simulations and experimental system. The Z1 mote is a general purpose development platform for WSNs. It is compatible with the popular Tmote family of motes. According to Zolertia, the Z1 offers roughly a two times improvement in performance over the Tmote motes [25].

One possible operating system to use for our system is Contiki [26]. This is the operating system that we used for our experimental system and simulations. Contiki is a lightweight, open-source operating system designed for use in WSNs. It is based on an event-driven kernel, where preemptive multithreading is implemented as an application library that can be linked to programs that require it. As a result, it combines the advantages of event-driven systems and systems that use multi-threading.

6. Simulation and Experimentation

IET Wireless Sensor Systems, Vol. 6, No. 1, 2016, pp. 26-33 (DOI 10.1049/iet-wss.2014.0090).

There are many potential applications for our SHM technique. One such application is that of railroad monitoring. We investigated this potential application in detail through experimentation and simulation. For our experiments, we used a model railroad setup. This allowed us to carefully control extraneous factors (such as train weight, track temperature, and train speed) and structural condition. Importantly, the results obtained from the model setup provide insight into the behavior of full-scale systems. A picture of the setup can be seen in Figure 8. For our simulations, we used Cooja. Cooja is a network simulator that was designed to simulate WSNs whose motes run the Contiki operating system [27]. In Cooja, a simulated Contiki mote is an actual compiled and executing Contiki system, compiled for a specific hardware platform.

a. Energy consumption simulation results

We performed simulations to investigate the improvement in efficiency garnered by implementing our system with a distributed architecture, using the algorithm of Pseudocode 1, over a centralized architecture in a railroad monitoring application. To measure energy in the simulations, Contiki's Powertrace application (a power state tracking mechanism) was used.

Overall, we performed two simulations. For both simulations a system with ten motes and a base station was used. The motes and base station were placed along a virtual, closed railroad track. A visualization of the system can be seen in Figure 6. A virtual train was made to move around the track in a clockwise fashion, starting and ending at the base station. As the train passed over the motes, the motes were made to collect simulated vibration data. Regarding transmissions, each node (including the base station) had only enough power to reach its immediate neighbors. Contiki's Rime protocol stack was used for wireless communication.

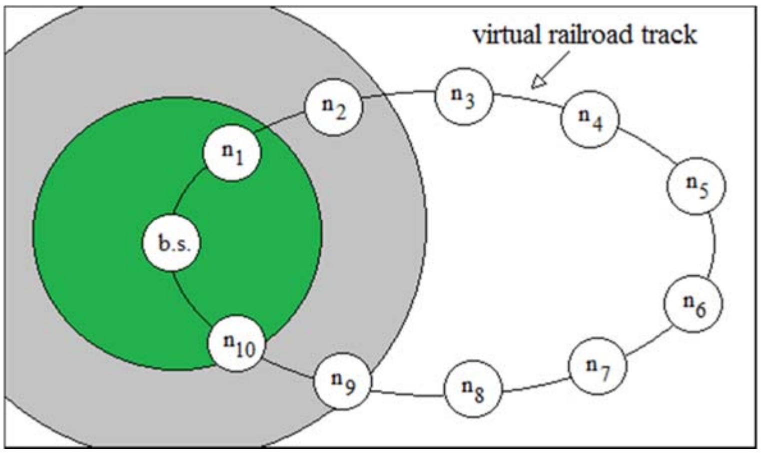

Figure 6. Illustration of the system used in the energy consumption simulations. The inner circle surrounding the base station, which has a radius of 50m, indicates the size of the area over which the base station and motes can reliably transmit.

In the first simulation, a centralized architecture was implemented in which all of the processing was done by the base station. As the train passed over each mote, each mote was made to collect vibration data and immediately transmit the raw data to the base station. Since the nodes only had enough power to transmit to their two nearest neighbors, multi-hop communications were necessary. Here, a shortest path routing protocol was used. In the second simulation, a distributed architecture was used. The algorithm of Pseudocode 1 was implemented on the motes. As the train passed over each mote, the mote was made to collect vibration data, calculate the AAD of the data, and determine the corresponding index value. The mote was then made to combine its index value with those from any previous motes to form a tuple and search for the tuple in its database. Assuming no damage was found, the mote would then send its index value and those from previous motes on to the next mote in the transmission chain (the next mote in the direction of the movement of the train). In each simulation, we measured the amount of energy that each mote spent on transmitting. Figure 7 shows the data obtained through the simulation. The curves in the figure are fits to the data determined using MATLAB.

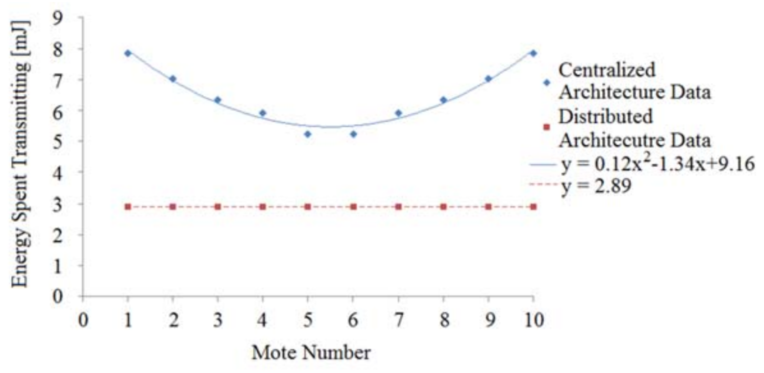

Figure 7. Plot of energy spent transmitting versus mote number for local and distributed processing. The mote numbers correspond to those in Figure 6. The discrete points represent the data obtained from the simulations.

Figure 7 shows that when the vibration data is processed locally, all of the motes spend the same amount of energy transmitting. This is because the number of transmissions per mote and the amount of data per transmission are the same for each mote. It can also be seen that when the vibration data is processed by the base station, each mote spends substantially more energy transmitting than it does in the case when processing is done on the individual motes.

Recall that in the simulation for the centralized architecture, a shortest path routing algorithm was used. Referring to Figure 6, this means that motes n_1 through n_5 transmit counterclockwise through the network to the base station, and that motes n_6 through n_{10} transmit clockwise through the network to the base station. As a result of this, motes n_1 and n_{10} had to forward the most packets. This means that they also had to spend the most energy transmitting. Motes n_2 and n_9 had to forward the second most amount of packets, and thus had to spend the second most amount of energy transmitting. In general, the number of packets that a mote must forward, and the amount of energy that a mote must spend transmitting, decreases as one moves away from the base station. This is why the graph for the centralized architecture, as seen in Figure 7, is concave.

It should be noted that, although motes n_5 and n_6 only have their own data to transmit and no packets to forward in the centralized architecture, they still spend more energy transmitting than they do when processing is done locally. This is because, when the data is processed by the base station, the motes have to send more data per transmission than they do when processing is done in a distributed fashion. When the data is processed on the motes, it is compressed into a smaller representative form before it is transmitted. On average, with the distributed architecture, the motes used 50% less energy transmitting than they did with the centralized architecture.

b. Results from our experimental system

We tested an experimental version of our system on a model railroad setup. The system consisted of two motes and a base station. The two motes were modified versions of Zolertia's Z1 mote and the base station was a PC. We modified the Z1 mote by adding an analog accelerometer and an external voltage reference. We did this because the digital accelerometer that the Z1 comes with was not capable of sampling fast enough for our application (it could sample at 3.2kHz whereas we required 10kHz) and the supplied voltage reference was not stable enough for our analog-to-digital conversions.

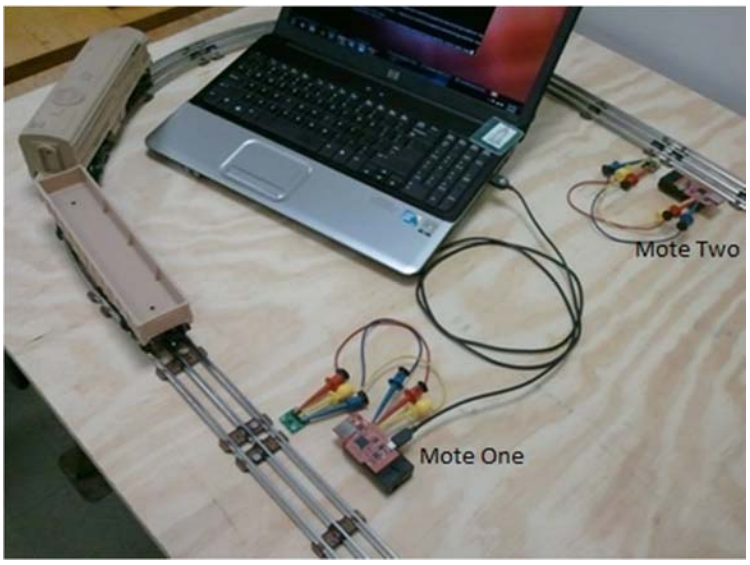

Figure 8. Picture of our experimental system.

In the experiment, to simulate track damage, fastening elements in the track were loosened. Through preliminary experimentation, we found that the AAD of the vibrations of the structure supporting the track was sensitive to such damage. As such this is the statistical feature that the motes were made to monitor. The only extraneous factor that was allowed to vary in the experiment was train weight. This factor we also found (through preliminary experimentation) to affect the AAD of the support vibrations. In particular, we found that the AAD of the vibrations increased linearly with cargo (mass added to the train) weight. The other significant factors, such as train speed and air temperature were held constant.

Regarding our setup (referring to Figure 8), Mote One was connected to the base station via a USB interface. Mote Two wirelessly transmitted to Mote One. In our experiment, we first sent an engine attached to an empty gondola car around a healthy track at a speed of 0.18rev/s. Each of the motes collected vibration data, calculated the AAD of its data, and mapped the AAD value to an index value. Overall, the tuple produced by the two index values was found to be healthy. This was the correct result, as no damage had been done to the track. Next, two weights totaling 3.26 kg were placed in the gondola car and the train was again sent around a healthy track at a speed of 0.18rev/s. Here, for each of the motes, the vibration levels and their associated index values were higher than they were for the run with the empty gondola car. Again, the system determined that the index values were such to produce a healthy tuple. This was the correct result, as no damage had yet been done to the track. After this, to simulate damage, a couple of screws were taken out of the track by Mote One. The weights were taken out of the gondola car and the train was then sent around the track at the same speed as the previous two runs. Here, Mote Two responded as it did when the train with the empty gondola car was sent around the track with the screws tightened. The AAD of the vibrations corresponded to the same index value. On the other hand, the vibrations at Mote One changed and the AAD value corresponded to a different index value than it did for the healthy run with the empty gondola car. As a result of this, a tuple other than a healthy tuple was produced and the system was able to detect that damage had occurred.

7. Conclusion and Future Work

Largely as a result of environmental and stimuli-related variability, WSNs have not seen widespread commercial usage in SHM applications [1]. Most deployments have been of an experimental nature [28, 29]. In this paper, we present a dynamic pattern matching technique that can adequately deal with environmental and stimuli-related variability in many cases of SHM. We described our method in general and discussed a case study involving railroad monitoring. Simulations reveal that, for our railroad application, we are able to cut energy consumption due to transmitting by 50% by using a distributed architecture over a centralized architecture. Future work includes an investigation of the tradeoffs between energy and accuracy, and the possibility of using energy harvesting to power the system.

References

1. Chesné, S. and Deraemaeker, A.: 'Damage Localization Using Transmissibility Functions: A Critical Review', *Mechanical Systems and Signal Processing*, 2013, 38, (2), pp. 569-584
2. Farrar, C.R., Doebling, S.W., Cornwell, P.J., and Straser, E.G.: 'Variability of Modal Parameters Measured on the Alamosa Canyon Bridge', IMAC-XV: A Conference on Structural Dynamics, 1997
3. Peeters, B., Maeck, J., and Roeck, G.D.: 'Vibration-Based Damage Detection in Civil Engineering: Excitation Sources and Temperature Effects', *Smart Materials and Structures*, 2001, 10, (3)
4. Lamsa, V. and Raiko, T.: 'Novelty Detection by Nonlinear Factor Analysis for Structural Health Monitoring', in, 2010 IEEE International Workshop on Machine Learning for Signal Processing
5. Deraemaeker, A., Reynders, E., De Roeck, G., and Kullaa, J.: 'Vibration-Based Structural Health Monitoring Using Output-Only Measurements under Changing Environment', *Mechanical Systems and Signal Processing*, 2008, 22, (1), pp. 34-56
6. Hongyang, Z., Junqi, G., Xiaobo, X., *et al.*: 'Environmental Effect Removal Based Structural Health Monitoring in the Internet of Things', in, Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2013
7. Hu, W.-H., Caetano, E., and Cunha, Á.: 'Structural Health Monitoring of a Stress-Ribbon Footbridge', *Engineering Structures*, 2013, 57, (0), pp. 578-593
8. Kullaa, J.: 'Structural Health Monitoring under Nonlinear Environmental or Operational Influences', *Shock and Vibration*, 2014, 2014
9. González, A.G. and Fassois, S.D.: 'Vibration-Based Statistical Damage Detection for Scale Wind Turbine Blades under Varying Environmental Conditions', in, Surveillance 7, 2013
10. Jin, S.-S. and Jung, H.-J.: 'Vibration-Based Structural Health Monitoring Using Adaptive Statistical Method under Varying Environmental Condition', in, Proceedings of SPIE Vol. 9064, 2014
11. Kamrujjaman Serker, N.H.M., Wu, Z., and Li, S.: 'A Nonphysics-Based Approach for Vibration-Based Structural Health Monitoring under Changing Environmental Conditions', *Structural Health Monitoring*, 2010, 9, (2), pp. 145-158
12. Gorinevsky, D. and Gordon, G.: 'Spatio-Temporal Filter for Structural Health Monitoring', in, American Control Conference, 2006
13. Bocca, M., Toivola, J., Eriksson, L.M., *et al.*: 'Structural Health Monitoring in Wireless Sensor Networks by the Embedded Goertzel Algorithm', in, 2011 IEEE/ACM International Conference on Cyber-Physical Systems
14. Lynch, J.P., Sundararajan, A., Law, K.H., *et al.*: 'Embedding Damage Detection Algorithms in a Wireless Sensing Unit for Operational Power Efficiency', *Smart Materials and Structures*, 2004, 13, (4)
15. Bocca, M., Eriksson, L.M., Mahmood, A., *et al.*: 'A Synchronized Wireless Sensor Network for Experimental Modal Analysis in Structural Health Monitoring', *Computer-Aided Civil and Infrastructure Engineering*, 2011, 26, (7), pp. 483-499

16. Sukun, K., Pakzad, S., Culler, D., *et al.*: 'Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks', in, 6th International Symposium on Information Processing in Sensor Networks, 2007
17. Hackmann, G., Sun, F., Castaneda, N., *et al.*: 'A Holistic Approach to Decentralized Structural Damage Localization Using Wireless Sensor Networks', *Computer Communications*, 2012, 36, (1), pp. 29-41
18. Krishnamurthy, V., Fowler, K., and Sazonov, E.: 'The Effect of Time Synchronization of Wireless Sensors on the Modal Analysis of Structures', *Smart Materials and Structures*, 2008, 17, (5)
19. Scianna, A., Jiang, Z., Christenson, R., and DeWolf, J.: 'Implementation of a Probabilistic Structural Health Monitoring Method on a Highway Bridge', *Advances in Civil Engineering*, 2012, 2012
20. Phares, B., Lu, P., Wipf, T., *et al.*: 'Field Validation of a Statistical-Based Bridge Damage-Detection Algorithm', *Journal of Bridge Engineering*, 2013, 18, (11), pp. 1227-1238
21. Shakya, R.K., Singh, Y.N., and Verma, N.K.: 'Generic Correlation Model for Wireless Sensor Network Applications', *Wireless Sensor Systems, IET*, 2013, 3, (4), pp. 266-276
22. de Jager, D. and Reeve, J.S.: 'Efficient Information Valuation and Costing for Distributed Wireless Sensor Networks', *Wireless Sensor Systems, IET*, 2012, 2, (3), pp. 191-200
23. Hackmann, G., Weijun, G., Guirong, Y., *et al.*: 'Cyber-Physical Codesign of Distributed Structural Health Monitoring with Wireless Sensor Networks', *IEEE Transactions on Parallel and Distributed Systems*, 2014, 25, (1), pp. 63-72
24. Tel, G.: *Introduction to Distributed Algorithms*, (Cambridge University Press, Cambridge, 2000), pp. 232-245
25. Uwase, M.-P., Long, N., Tiberghien, J., *et al.*: Poster Abstract: Outdoors Range Measurements with Zolertia Z1 Motes and Contiki', in Langendoen, K., Hu, W., Ferrari, F., Zimmerling, M., and Mottola, L. (eds.), *Real-World Wireless Sensor Networks*, (Springer International Publishing, 2014)
26. Dunkels, A., Gronvall, B., and Voigt, T.: 'Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors', in, 29th Annual IEEE International Conference on Local Computer Networks, 1997
27. Osterlind, F., Dunkels, A., Eriksson, J., *et al.*: 'Cross-Level Sensor Network Simulation with Cooja', in, IEEE Conference on Local Computer Networks, 2006
28. Whelan, M.J., Gangone, M.V., Janoyan, K.D., and Jha, R.: 'Real-Time Wireless Vibration Monitoring for Operational Modal Analysis of an Integral Abutment Highway Bridge', *Engineering Structures*, 2009, 31, (10), pp. 2224-2235
29. Ceriotti, M., Mottola, L., Picco, G.P., *et al.*.: 'Monitoring Heritage Buildings with Wireless Sensor Networks: The Torre Aquila Deployment', in, International Conference on Information Processing in Sensor Networks, 2009