2012

DATA DRIVEN INTELLIGENT AGENT NETWORKS FOR ADAPTIVE MONITORING AND CONTROL

Wenjia Liu
Michigan Technological University

DATA DRIVEN INTELLIGENT AGENT NETWORKS FOR ADAPTIVE MONITORING AND CONTROL

By

Wenjia Liu

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

(Electrical Engineering)

MICHIGAN TECHNOLOGICAL UNIVERSITY

2012

This dissertation, "DATA DRIVEN INTELLIGENT AGENT NETWORKS FOR ADAPTIVE MONITORING AND CONTROL," is hereby approved in partial fulfillment of the requirements for the Degree of DOCTOR OF PHILOSOPHY IN ELECTRICAL ENGINEERING.

Department of Electrical and Computer Engineering

Signatures:

Dissertation Advisor _____
Dr. Bo Chen

Committee Member _____
Dr. Jindong Tan

Committee Member _____
Dr. R. Andrew Swartz

Committee Member _____
Dr. Ossama Abdelkhalik

Department Chair _____
Dr. Daniel R. Fuhrmann

Date _____

Contents

[1] ©Elsevier. Portions reprinted with permission, from W. Liu and B. Chen, "Optimal control of mobile monitoring agents in immune-inspired wireless monitoring networks", Journal of Network and Computer Applications, vol. 34, no. 6, pp. 1818-1826, 2011.

[2] A portion of this chapter has been submitted to Mechanical Systems and Signal Processing, Elsevier

[3] ©DEStech. Portions reprinted with permission, from W. Liu and B. Chen, "Bio-inspired Computing Algorithms for Adaptive Structural Health Monitoring," Proceedings of the Eighth International Workshop on Structural Health Monitoring, 2011. Lancaster, PA: DEStech Publications, Inc.

[4] ©Wiley. Portions reprinted with permission, from B. Chen and W. Liu, "Mobile Agent Computing Paradigm for Building a Flexible Structural Health Monitoring Sensor Network," Computer-Aided Civil and Infrastructure Engineering, vol. 25, no. 7, pp. 504-516, 2010.

[5] ©Inderscience. Portions reprinted with permission, from B. Chen and W. Liu, "A web-based structural health monitoring sensor network," International Journal of Computer Applications in Technology, Vol. 44, 2012.

[6] ©Inderscience. Portions reprinted with permission, from B. Chen and W. Liu, "A High Computational Power Wireless Sensor Network for Distributed Structural Health Monitoring," International Journal of Sensor Networks, vol. 11, 2012.

[7] ©SAE. Portions reprinted with permission,SAE Paper No. 2012-01-0742 ©2012 SAE International.

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to thank my advisor, Dr. Bo Chen, for her consistent support and encouragement in my Ph.D study and research. I would not have accomplished much of my work without her inspiration and great efforts to explain things clearly for me.

I also would like to thank my committee members: Dr. Jindong Tan, Dr. R. Andrew Swartz and Dr. Ossama Abdelkhalik for their time and knowledge in the process of my pursuit of Ph.D degree.

I thank Mr. Glen E. Archer, for his support in my lab teaching. I thank Dr. Chuanzhi Zang, who shared with me lots of knowledge in modeling and simulation. My sincere thanks also go to my labmates: Wei Chen, Wei Luo, Poowanart Poramapojana, Lei Feng and Eddy Trinklein.

I would thank my parents for their sopporting me throughout my life. I thank my wife, Shuang, for all the help and understanding she gave me.

Abstract

To analyze the characteristics and predict the dynamic behaviors of complex systems over time, comprehensive research to enable the development of systems that can intelligently adapt to the evolving conditions and infer new knowledge with algorithms that are not predesigned is crucially needed. This dissertation research studies the integration of the techniques and methodologies resulted from the fields of pattern recognition, intelligent agents, artificial immune systems, and distributed computing platforms, to create technologies that can more accurately describe and control the dynamics of real-world complex systems. The need for such technologies is emerging in manufacturing, transportation, hazard mitigation, weather and climate prediction, homeland security, and emergency response.

Motivated by the ability of mobile agents to dynamically incorporate additional computational and control algorithms into executing applications, mobile agent technology is employed in this research for the adaptive sensing and monitoring in a wireless sensor network. Mobile agents are software components that can travel from one computing platform to another in a network and carry programs and data states that are needed for performing the assigned tasks. To support the generation, migration, communication, and management of mobile monitoring agents, an embeddable mobile agent system (Mobile-C) is integrated with sensor nodes. Mobile monitoring agents visit distributed sensor nodes,

read real-time sensor data, and perform anomaly detection using the equipped pattern recognition algorithms. The optimal control of agents is achieved by mimicking the adaptive immune response and the application of multi-objective optimization algorithms. The mobile agent approach provides potential to reduce the communication load and energy consumption in monitoring networks.

The major research work of this dissertation project includes: (1) studying effective feature extraction methods for time series measurement data; (2) investigating the impact of the feature extraction methods and dissimilarity measures on the performance of pattern recognition; (3) researching the effects of environmental factors on the performance of pattern recognition; (4) integrating an embeddable mobile agent system with wireless sensor nodes; (5) optimizing agent generation and distribution using artificial immune system concept and multi-objective algorithms; (6) applying mobile agent technology and pattern recognition algorithms for adaptive structural health monitoring and driving cycle pattern recognition; (7) developing a web-based monitoring network to enable the visualization and analysis of real-time sensor data remotely. Techniques and algorithms developed in this dissertation project will contribute to research advances in networked distributed systems operating under changing environments.

Chapter 1

Introduction

Distributed sensing and monitoring systems are playing an important role in manufacturing, civil engineering, transportation systems, chemical process, power grids, and homeland security. As the complexity of the monitored objects and environment increases, the monitoring network needs to provide higher quality of accuracy, reliability, adaptability, and autonomy. To meet these requirments, this dissertation research studies data-driven pattern recognition based on real time sensor data. The effects of feature extraction methods, similarity measures and environmental factors on the performance of pattern recognition are also investigated. To achieve adaptive monitoring, the mobile-agent-based monitoring paradigm is adopted and the network platform is developed. The generation and distribution of the monitoring agents over the network are optimized by the multi-objective optimization algorithms and the concept of artificial immune system.

The major work of this dissertation research includes following. Firstly, this dissertation investigates the time series data representation methods and similarity measures for sensor data feature extraction and pattern recognition. The pattern recognition technique basically deals with two issues: finding out the accurate description of the studied object (feature extraction) and then categorize the object into a pattern based on its description (pattern classification). Feature extraction methods are assessed by examining the distribution of feature vectors in the feature space and the success rate of pattern recognition. The impact of similarity measures on the pattern recognition success rate are also investigated. The test data used in this research are from the System Identification to Monitor Civil Engineering Structures (SIMCES) test [1, 2], which was performed on the Z24 bridge in Europe for almost one year. The analysis results show that both time-series data representation methods and the similarity measures have significant impact on the success rate of pattern recognition.

Secondly, this dissertation research integrates a mobile agent network middleware with a high computational power wireless sensor node. The sensor node has Linux operating system and C numerical libraries. The integrated mobile agent middleware, Mobile-C [3], can host both stationary and mobile agents. Agents can work locally on sensor nodes or travel between sensor nodes. Two-layer approach is utilized for the software design of the sensor node: the open architecture of the upper layer promotes the software reuse and speeds up the development cycle by employing available numerical algorithms in open software packages such as CLAPACK and Numerical Recipes in C. The lower

layser of the sensor node is designed to collect data from different types of sensors to support multi-modality sensing. The web service at sensor node level lowers the power consumption of the sensor node by reducing the amount of data transmitted. Users can also view the result of data processing and damage diagnosis online.

Thirdly, the dissertation studies the optimal control of mobile agents in artificial immune system based (AIS-based) monitoring networks. Artificial immune system (AIS) is an artificial intelligence technique inspired by the principles and features of natural immune systems. AIS can be applied to the scenarios where a parallel, distributed, and adaptive system is desirable. In AIS-based monitoring networks, the monitoring work is achieved by a group of mobile agents which are equipped with pattern recognition algorithms. The optimal control of agents includes agent generation and agent distribution. The optimization of the agent generation is to minimize the response time for the mobile agents to diagnose the damage. The optimization of agent distribution is to increase the probability of damage detection and extend the lifetime of a monitoring network. The optimization algorithms are implemented by utilizing genetic algorithms to find the non-dominated solutions as close to Pareto front as possible.

Finally, mobile agent technology, high computational power monitoring networks, and pattern recognition techniques are applied to structural health monitoring and driving cycle pattern recognition. In mobile-agent-based structural health monitoring application, mobile agents migrate from user's computer to remote sensor nodes which are mounted on a scaled

steel bridge. The agents diagnose if the bridge is in normal pattern or damaged pattern by utilizing the autoregressive (AR) model [4] coefficients of the acceleration data collected by the senor nodes. A diagnose report is then sent back to the remote user by the mobile agents. In the damage localization application, the distance between the feature vectors of the normal pattern data and damaged pattern data is calculated for each sensor. The result shows that the closer the sensor is to the damaged location, the larger this distance value is. This finding indicates that the distance between the feature vectors of different patterns can be used as a measure for the damage localization. In the driving cycle pattern recognition application, four federal driving cycles are used to generate reference feature vectors. Real world driving cycles collected by a mild-hybrid Chevy Malibu driven in the urban and suburban area are used to validate the effectiveness of the reference feature vectors for driving cycle pattern recognition. In the last application, a web-based structural health monitoring network is designed to provide sensor level web services. This web-based monitoring network is able to display remote sensor data in a web browser and perform damage diagnosis.

Chapter 2

Literature Review

A literature research was conducted to understand the state of the art of the dissertation research area. This chapter reports the literature review results in the area of pattern recognition, mobile agents, agent networks, and artificial immune systems.

2.1 Pattern Recognition

Pattern recognition is a technique to assign test objects into categories or classes [5] based on the features these objects possess. Pattern recognition is of importance in many areas, including character recognition [6, 7, 8], machine vision [9], data mining [10, 11, 12], speech recognition [13, 14] and computer aided diagnosis [15]. Pattern recognition

typically contains two stages of work: feature extraction and pattern classification [5]. The first stage selects properties of the studied object which can best describe the object and be used to distinguish itself from other objects. The second stage is to classify the object to one of known classes using a pattern recognition algorithm. The pattern classification algorithm is also called a classifier.

Feature extraction method is often used in pattern recognition and image processing [16, 17, 18, 19]. In image processing, the visual factors are usually adopted as the image features, generally including edge, shape, color and texture [20, 21, 22]. In pattern recognition, the features to be extracted depend on the specific problem. To extract features from time series, a number of methods have been proposed, including Single Value Decomposition (SVD) [23], Discrete Fourier Transformation [24, 25], Discrete Wavelet Transformation [26], Adaptive Piecewise Constant Approximation [27], Discrete Cosine Transformation [23], Chebyshev Polynomials [28], Piecewise Aggregate Approximation [10], and Symbolic Aggregate Approximation [29].

The second stage of pattern recognition is the classification of the test objects. Classification is to categorize the feature vectors of the test objects into corresponding patterns. The algorithm to realize the classification is called a classifier [5, 30]. If all the training data have class information, the classification is called supervised classification. The commonly used supervised learning classification algorithms include: Naive Bayesian classifier [31, 32], decision trees [33, 34], support vector machine [35, 36] and K-Nearest

Neighbor [5, 37]. In unsupervised classification, training data don't have class information. The classification algorithms for unsupervised classification are clustering algorithms, including K-means clustering [38, 39] and Hierarchical clustering [5, 40]. An unsupervised pattern recognition method inspired by artificial immune system is proposed for structural damage classification in [41].

2.2 Mobile Agents and Agent Networks

A mobile agent is the composition of computer program and data which can travel from one computing platform to another. The agent technology becomes popular for the reasons such as: parallel performance of tasks, dynamic adaptation to changing conditions, easy deployment of new program and being able to exchange information. The agent concept has been widely adopted in many areas such as: control system [42, 43], network management [44], information management [45], E-commerce [46, 47]. The existing agent platforms include: MOLE [42], Aglets [48, 49], Concordia [50], Ara [51], TACOMA [52], and Mobile-C [3]. Mobile-C is an embeddable mobile agent system compliant with Foundation for Intelligent Physical Agents (FIPA). The FIPA is an internationally recognized agent standards.

In a mobile agent network, agents can carry data and programs while moving from one computing platform to another. One task can be decomposed into several sub-tasks and

these sub-tasks can be performed by multiple agents. These agents work cooperately and dynamically adapt themselves to the changing environment. Agent coordination mechanism is often designed to enable agents cooperation to optimize the performance of the overall agent network. One important feature of an agent network is its scalability. The number of the network nodes can be easily adjusted based on the task scale.

The agent platform adopted in this work is Mobile-C, a defined above FIPA compliant agent platform for mobile C/C++ agents [3]. Ch [53], an embedded C/C++ interpreter has been integrated into Mobile C platform. One advantage of C/C++ being the agent program language is that the agent can interface with the hardware and low-level software. A large amount of existing C/C++ programs can be reused for the mobile agent programs. All the agents running on this platform are composed of XML code and C/C++ programs. For each agent, the C/C++ program is wrapped by the XML code which specifies the identification information of an agent, including ID number, destination, itinerary and other details. Mobile-C has been successfully applied to some fields, such as: traffic detection and management [54], vision fusion for robotics [55] and flexible automation systems [56].

2.3 Artificial Immune Systems

Artificial immune systems are computational systems inspired by the ideas, processes and theories of the biological immune system [57, 58]. The biological immune system can

detect various antigens, and identify them so as to protect the healthy cells and organs. Immune cells, such as B cells and T cells are major players to recognize and eliminate the antigens. If the receptor of a B cell matches the feature of an invading antigen, this antigen will be bound by the B cell. The infected cells will be killed by the activated T cells. If a new type of antigen invades a healthy organ, B cells whose receptors have high affinity with the antigen will be activated. After a process of clonal selection, additional B cells with similar receptors are generated. The B cells whose receptors have high affinity with the antigens become the memory cells and will stay in the immune system for a relatively long time. The immune system can quickly react to this type of antigen when it appears again. Artificial immune system has found its applications in data analysis [59], computer security [60], unsupervised structural damagepattern recognition [41], misbehavior detection in mobile ad hoc networks [61].

Chapter 3

Theoretical Background of Pattern Recognition

3.1 Concept of Pattern Recognition

Pattern recognition is the scientific study of how to describe and distinguish patterns and reasonably classify these patterns into categories. A pattern is the abstract entity to be recognized or identified. Pattern recognition technique finds its applications in a number of fields, including: character recognition where "pattern" refers to a specific character, computer aided diagnosis where a "pattern" refers to images captured by microscope, internet security where a "pattern" may refer to emails and programs, biological statistics

where a "pattern" may refer to a DNA sequence and speech recognition where a "pattern" may refer to human being speech waveform. There are two types of pattern recognitions: supervised learning and unsupervised learning. In supervised learning, all the categories are known and the training data have class labels. In unsupervised learning, there are no predefined categories.

The process of pattern recognition typically includes following steps: (1) Collect data from the object to be classified; (2) Data preprocessing: such as data normalization (3) Feature extraction: identify a number of features to describe the object. These features form a feature vector. (4) Classification: assign the object to a category using a classification algorithm.

3.2 Feature Extraction

Time series is one of the most commonly used data formats in real world. It is being generated in a tremendous speed from almost every application area. Due to its high dimension, direct processing and storage of time series in its raw format is expensive. "Two key aspects for achieving effectiveness and efficiency when managing time series data are representation methods and similarity measures" [62]. In the last decades, a number of representation methods and similarity measures have been proposed to extract features from time series data for indexing, classification, and clustering. The

objective of feature extraction is to find a representation at a lower dimensionality that preserves the fundamental characteristics of the original time-series data [63]. The time series representation methods can be classified as shape-based method, structure-based (or model-based) method, and dimensionality reduction. For long time series data, model-based and dimensionality reduction methods are more effective.

Model-based time series representation methods extract global features from time series, create feature vectors, and use these feature vectors to measure similarity of time series for classification and clustering. Time series data are usually fitted into models, such as Box Jenkins model or Markov Model, and the parameters of the model are used to form feature vectors. The dimensionality reduction methods are typically based on data transformation. Many dimensionality reduction methods have been reported in the literature, such as Discrete Fourier Transformation (DFT) [24, 25], Single Value Decomposition (SVD) [23], Discrete Wavelet Transformation (DWT) [26], Piecewise Approximation [27], and Chebyshev Polynomials (CHEB) [28].

Similarity measure is important for evaluating feature extraction methods. There are over a dozen distance measures have been reported in the literature for mining and indexing time series. These similarity measures include Euclidean Distance [25], Mahalanobis Distance, Cosine Distance, Standardized Euclidean (Seuclidean) Distance, Correlation Distance, and Dynamic Time Warping (DTW) [64, 65].

3.2.1 Model-based Feature Extraction Methods

In this dissertation, autoregressive model is used to model a time series data. The AR model-based feature extraction method fits time series data into an AR model and uses the coefficients of the AR model as members of the feature vector. To reduce noise effects, the time series data Z are normalized by

$$x_i = \frac{z_i - \mu_i}{\sigma_i} \quad i = 1, 2, \cdots\cdots, n \tag{3.1}$$

where μ_i and σ_i are the mean and standard deviation of the time series Z.

Then, the normalized time series data x can be fitted into an p-order AR model as shown in equation 3.2,

$$x_k = \sum_{i=1}^{p} \alpha_i x_{k-i} + r_k \quad k = p+1, \cdots\cdots, n \tag{3.2}$$

where α_i, $i = 1, 2, ..., p$ are the coefficients of the AR model. The feature vector of the time series data x, $F(X)$, is composed of the coefficients of the AR model as shown in Equation 3.3,

$$F(X) = (\alpha_1,\ \alpha_2, \cdots\cdots, \alpha_p)^T \tag{3.3}$$

3.2.2 Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is one type of discrete transform which transforms a function in the time domain into another in the frequency domain. Given a time series with the length of N, the DFT of is defined to be X consisting of complex numbers as shown in Equation 3.4

$$X_k = \sum_{i=1}^{N} x_{i-1} e^{-j\frac{2\pi}{N}(k-1)(i-1)}, \quad k = 1, 2, \cdots\cdots, N \tag{3.4}$$

To reduce the dimensionality of the time series X into a low dimensional feature space, the first n model frequencies $f_1, \cdots, f_2, \cdots, f_n$ and corresponding signal amplitudes $a_1, a_2, \cdots, a_n$ are used to form feature vectors. The general form of a feature vector can be represented as:

$$F(X) = (f_1, k \times a_1, f_2, k \times a_2, \cdots, f_n, k \times a_n) \tag{3.5}$$

where k is the weighting factor of the amplitudes and n is the number of frequencies to be used.

3.2.3 Discrete Wavelet Transform

Discrete Fourier transform is a powerful and commonly used method for feature extraction. However, discrete Fourier transform does not have time stamp in the DFT coefficients. From the DFT coefficients, the frequency element can be seen but users won't know at which time these frequency elements occur. To overcome this disadvantage, discrete wavelet transform (DWT) is adopted for feature extraction and dimension reduction [26, 66, 67, 68]. The DWT can keep signal properties in both time and frequency domain thus carry more information than DFT coefficients. In [26], an efficient time series matching technique is proposed by using Haar Wavelet Transform. Discrete wavelet transform decomposes a signal into layers of coefficients. These coefficients contain both frequency and time domain information. Discrete wavelet transform has been applied for feature extraction in different fields [66, 67, 68]. Given a time series with the length of n, the discrete wavelet transform (DWT) of x is calculated by passing the time series signal through a series of low pass and high pass filters as shown in equation

$$y_l[n] = x[n] * g[n] = \sum_{k=-\infty}^{\infty} x[k]g[2n-k] \tag{3.6}$$

$$y_h[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[2n-k] \tag{3.7}$$

where $g[n]$ and $h[n]$ are low pass filter and high pass filter, respectively. The outputs of the high pass filter are detail coefficients while the outputs of the low pass filter are approximation coefficients. The approximation coefficients are further decomposed in the next iteration while the detail coefficients are kept as the current level wavelet coefficients. To form feature vectors from wavelet coefficients, feature extraction method proposed in [69] is employed. This feature extraction method consists of two steps: cluster determination and feature extraction. The cluster determination process divides the wavelet coefficients into a number of clusters $c_1, c_2, \cdots\cdots c_k$ and the feature extraction process calculates the feature vector. The elements of a feature vector are Euclidean norms of each cluster $F = (\|c_1\|_2, \|c_2\|_2, \cdots\cdots \|c_k\|_2)$. The clusters $c_1, c_2, \cdots\cdots c_k$ are determined as row vectors such that each cluster contains a significant wavelet coefficient near the midpoint of each cluster.

Figure 3.1 shows the process of cluster determination and feature extraction from the raw data of multiple data patterns. First, the DWT coefficient matrices of raw data from multiple patterns are calculated. The dimensions of these coefficient matrices are same if time series data have same length. To find significant wavelet coefficients, the Central Limit Theorem [70] is applied to the elements of the DWT coefficient matrices to generate a new matrix G as shown in equation 3.8

$$G = \frac{1}{\sigma(R(\sum_{k=1}^{K} \tilde{B_k}))}(\sum_{k=1}^{K} \tilde{B_k} - \mu(R(\sum_{k=1}^{K} \tilde{B_k})) \cdot I) \tag{3.8}$$

where R is an operator to reduce a matrix by its last row, I is a matrix which has the same size as $\tilde{B}_k$ and has all the elements of 1. The members of the G matrix are then compared with a threshold and save the comparison results to the corresponding location in a matrix G_b. The comparison result is 1 when the member of the G matrix is greater than the threshold and 0 when the member of the G matrix is less than the threshold. Pittner and Kamarthi [69] prove that the 1s in the matrix G_b appear at the same locations where the significant wavelet coefficients occur in the matrices $\tilde{B}_k$. Based on the G_b matrix, the clusters are then formed with following rules: (1) each cluster contains one "1" element; (2) if one row contains no "1" element, this row is treated as one cluster. After the boundaries of each cluster are determined from the G_b matrix, the wavelet coefficients in the $\tilde{B}_k$ matrices are grouped into clusters using the cluster boundary information obtained from the G_b matrix. The feature vector of the $\tilde{B}_k$ matrix is calculated with the Euclidean norms of each cluster as shown in Figure 3.1.

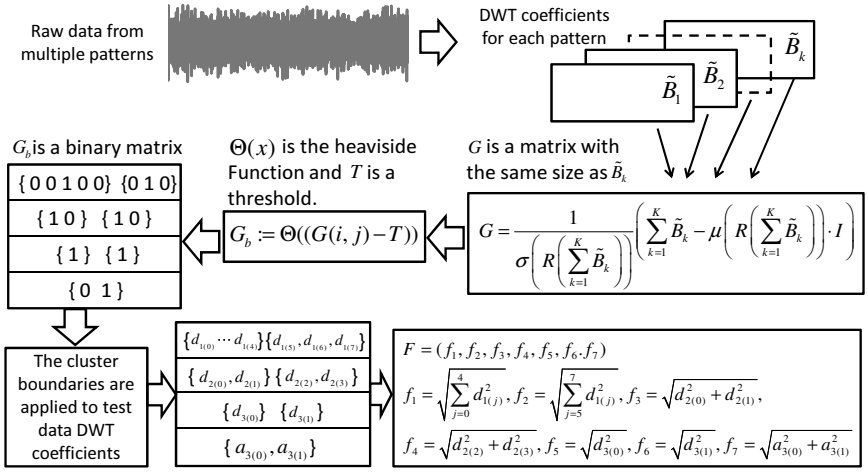


Figure 3.1: Feature extraction from DWT coefficients

3.2.4 Principal Component Analysis

Principal component analysis (PCA) is one of the most important data transform methods that can reduce the dimension of the data. Principal component analysis is also named Karhunen-Loeve transform (KLT). Assume that there are N vectors each of which contains M variables, and we want to compress these vectors, an $M \times N$ matrix to an $L \times N$ matrix. These vectors can be represented as $X_1 \cdots X_N$, and all of them can be represented as column vectors with M rows. The empirical mean values of each dimension is:

$$u[m] = \frac{1}{N} \sum_{i=1}^{N} X[m,n] \tag{3.9}$$

Every row of the raw data is subtracted by the mean value of this row resulting in $B = X - uh$, where h is a row vector with all the elements as 1. The $M \times N$ matrix B represents the deviations from the mean. The $M \times M$ covariance matrix can be calculated by:

$$C = E[B \cdot B^*] = \frac{1}{N} \sum B \cdot B^* \tag{3.10}$$

To find the Karhunen-Loeve transform of matrix X, the eigenvectors and eigenvalues of C are calculated. The eigenvectors of matrix C form an $M \times M$ matrix V and the eigenvalues of C compose an $M \times M$ diagnal matrix D. Sort the eigenvalue matrix D in an order of decreasing eigenvalues. Rearrange the eigenvector matrix V to pair the ith eigenvector

with the eigenvalue. Select the first L columns of V as a subset. The subset matrix is labelled as W. Then, the Karhunen-Loeve transform (Y) of the matrix X can be represented as:

$$Y = W^* \cdot \frac{B}{s \cdot h} \tag{3.11}$$

where s is a vector whose elements are the square root of each element along the diagonal of C.

3.2.5 Singular Value Decomposition

The singular value decomposition has been widely used for rank and dimension reduction in pattern recognition and statistics [5]. SVD is based on the global information of the matrix and has good capability of retaining information, which makes it a good candidate for feature extraction. The singular value decomposition for a $l \times n$ sized matrix X of rank r can be written as:

$$X = [u_0, u_1, ..., u_{r-1}] \begin{bmatrix} \sqrt{\lambda_0} & & & \\ & \sqrt{\lambda_1} & & \\ & & \ddots & \\ & & & \sqrt{\lambda_{r-1}} \end{bmatrix} \begin{bmatrix} v_0^H \\ v_1^H \\ \vdots \\ v_{r-1}^H \end{bmatrix} \tag{3.12}$$

where u_i,v_i are the column vectors of $l \times l$ and $n \times n$ unitary matrix respectively, and λ_i are the nonzero eigenvalues of the matrix $X^H X$. The original matrix X can be approximated by a lower rank matrix by using singular value decomposition. Suppose the lower rank value is k, then the approximate matrix can be written as:

$$X \simeq \hat{X} = [u_0, u_1, ..., u_{k-1}] \begin{bmatrix} \sqrt{\lambda_0} v_0^H \\ \sqrt{\lambda_1} v_1^H \\ \vdots \\ \sqrt{\lambda_{k-1}} v_{k-1}^H \end{bmatrix} = U_k [a_0, a_1, ..., a_{n-1}] \tag{3.13}$$

where u_k is a matrix containing the first k columns of a $l \times l$ unitary matrix and a_n is the product matrix of $\lambda_k^{\frac{1}{2}} V_k^H$. So the relatively high dimensional matrix can be represented by the low dimensional matrix. If $k << r$, the computational burden can be greatly reduced.

3.3 Pattern Classification Methods

Pattern classification is a process to identify a pattern and make a reasonable decision of which category this pattern belongs to. The algorithm of pattern classification is also called classifier. In most cases, when talking about "classifier", the users have already trained the data and own the information of categories. The new coming patterns will be labeled by quantitatively analyzing the parameters of them. The number of the categories is limited, so the training data patterns can be represented as: $(f_1, l_1), (f_2, l_2), \cdots, (f_n, l_n)$, f represents the

quantized characteristics of the training data and l is the label of the pattern. These existing training data can help establish a classifier to evaluate and identify any new observations. The most commonly used classifiers are: k-nearest neighbors, support vector machine, naive Bayes, decision tree, and so on.

3.3.1 K-nearest Neighbor

K-nearest neighbor is an intuitive classification method based on the number of closest training data features in a feature space [5]. The K nearest neighbor algorithm requires the set of training data and the similarity measure to determine the "closeness". Suppose that there are totally N training feature vectors belonging to M number of classes. Among the k nearest neighbors of the test feature vectors, the amount of feature vectors belonging to each pattern is $m_1, m_2, \cdots, m_M$, respectively, and $\sum_{i=1}^{M} m_i = k$. The test data will be assigned to the group with most training features in the k nearest neighbors. If $k = 1$, then the label of the test data feature vector will be determined by the nearest neighbor only. Euclidean distance is often adopted as the similarity measure between features, however, this is not always the case. The selection of similarity measures is problem dependent. The success rate of KNN method for a large k value is relatively higher than that for a small k value.However, large k value destroys the locality of the estimation and increases the computation load. Another factor that affects the performance of the KNN method is the process of calculating the distances of all the feature vectors in a feature space and

searching for the nearest neighbors. KNN method is easy to implemented, however, as a lazy learning algorithm, it has large storage requirements.

3.3.2 Support Vector Machine

A support vector machine is one type of supervised learning methods for data analysis and classification. For a two-class classification problem, an SVM can categorize the input data into one of the two possible classes. Suppose that the test data are p-dimensional and belonging to two classes. If there is a $(p-1)$ dimensional hyperplane which can separate these test data correctly, this hyperplane can be regarded as a linear classifier [5]. The simplest case is that one straight line can divide a 2D plane into two parts and each part can contain the points of one pattern. There are multiple ways to draw this line to divide the plane in order to separate the points into two groups, however, there is a best approach which can make the line having the longest distance with points of each pattern. A linear support vector machine is to find a high dimensional hyperplane that can separate test data vectors. Technically, one hyperplane can only be able to divide the test data into two groups. If there are more than two patterns, the separation can be done in the form of one and the rest or in the tree structure. Let $x_i, i = 1, 2, \cdots, N$ be the P-dimensional feature vectors of the training data. The group of data can be represented as:

$$TD = (x_i, y_i) | x_i \in \mathscr{R}^P, y_i \in (-1, 1)_{i=1}^n \tag{3.14}$$

The hyperplane $g(x)$ can be represented as [5]:

$$g(x) = \omega^T x + \omega_0 = 0 \tag{3.15}$$

where ω is a weight vector and ω_0 is a threshold. To find out the optimal hyperplane, the optimization problem of finding the hyperplane is summarized as:

$$minimize\ J(\omega, \omega_0) \equiv \frac{1}{2}\|\omega\|^2 \tag{3.16}$$

$$subject\ to\ y_i(\omega^T x_i + \omega_0) \geq 1,\ i = 1, 2, \cdots, N \tag{3.17}$$

The solution of this optimization problem is:

$$\omega = \sum_{i=1}^{N} \lambda_i y_i x_i \tag{3.18}$$

where λ_i are Lagrange multipliers. The parameter ω are used to generate the feature vectors which are called support vectors lying on the margin of the hyperplane.

3.3.3 Naive Bayes

The naive Bayes classifier can predict the probability of a test pattern belonging to a particular category. The classifier is based on Bayes' theorem and assumes that the effects of all the features on a given pattern are conditionally independent. This assumption simplifies the computation and thus the classifier is considered "naive". Suppose that $P(x|y)$ is the conditional probability of x occurring given the condition of y, then this probability can indicate how much x can contribute to the judgment of the happening of y. The function for the Naive Bayes classifier is:

$$Y_{predict} = argmax\ P(Y = y)\prod_{i=1}^{n} P(X_i = x_i|Y = y) \tag{3.19}$$

In real world test, the product of the small probabilities may cause float point overflow, so log function can be used to replace the multiplication. Since log function is monotonic, the classification result will be the same.

$$Y_{predict} = argmax\ \{log(P(Y = y)) + \sum_{i=1}^{n} log(P(X_i = x_i|Y = y))\} \tag{3.20}$$

In naive Bayes classifier, the probability of each category and the probability of every feature occuring in each category are all involved in the classification function. This classification method is simple but works fine in many scenarios.

3.3.4 Decision Tree

A decision tree is one of the most popular methods for classification which constructs a multistage tree-like model for decision making. In each stage, one decision is made based on the criterion on that stage and a class keeps being rejected until it arrives at a pattern whose criterion the class can satisfy. As a nonlinear classification method, decision tree works well with the scenarios when there are many classes. Being a graphical tool, decision trees makes the problem visable and gives users an intuitive understanding. There are three types of nodes in a decision tree: (1) decision node, which is commonly represented by a square. Decision nodes represent the controllable factors and decisions can be made; (2) event or chance node, which is represented by a circle. This kind of node represents the uncontrollable factors: some events may happen, but they are beyond the decision maker's control ability. (3) terminal node, which is represented by a triangle. The terminal node represents the decision making result of a combination of decisions and events. The size of the decision tree needs to be carefully determined: it should be large enough but not too large to lose the generalization [5]. A drawback of the decision tree classifier is it's sensitive to the change of the training data. A subtle change of the training data may generate a quite different decision tree.

3.3.5 Bio-inspired Pattern Recognition

3.3.5.1 Neural Network Classification

The neural network is a paradigm for data and information analysis inspired by the biological neural network. It's popular due to the ability of exploring information from the complicated raw data and adaptive learning. The neural network is composed of a number of neurons which are interconnecting functional units. Each neuron has multiple inputs and multiple outputs and can work in two modes: training mode and using mode. The neurons are trained in the training mode by the input training patterns and learns the rules between input patterns and output result. When working in the using mode, the neurons identify the input test pattern and try to find out the corresponding output pattern if there is. If the test patterns are not the same as the input patterns in the training process, the firing rule is adopted to determine the output. The firing rule is an decision making strategy in the neural network which increases the network flexibility. For any unknown input pattern, the network will find a known input pattern which has the highest similarity with the unknown pattern, then the firing rule decides if the output of the known pattern is used as the output of the unknown pattern. A neural network basically consists of three layers: input layer, hidden layer and output layer. The feed-forward neural networks only allow the signal flow to strictly goes from the input to output, while in the feedback network, signal can travel

in both directions which makes the feedback neural network powerful but complicated. Figure 3.2 illustrates the feed-forward and feedback networks [71].

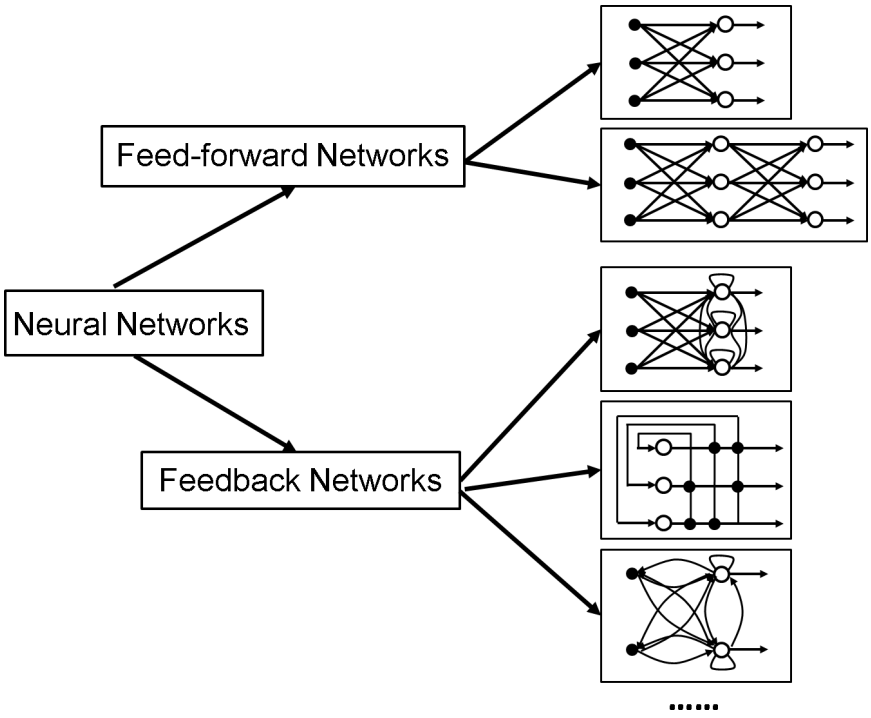


Figure 3.2: Structure of a simple neural network

3.3.5.2 Artificial Immune Pattern Recognition (AIPR)

The desirable characteristics of the immune system, such as adaptation, evolution, and fault tolerance, have inspired the applications of immune concept for supervised and unsupervised structural damage detection and classification [41, 72]. In supervised structural damage classification [41], the pattern representative feature vectors for each data pattern are generated by training data with appropriate pattern labels. By contrast,

for unsupervised structural damage pattern recognition algorithms [72], the pattern label information of the training data is not available. Figure 3.3 shows the concept of artificial immune pattern recognition (AIPR) based structural damage pattern recognition. The damage classification is to classify the detected damage to one of the possible damage patterns. The AIPR-based damage pattern recognition is achieved by establishing representative feature vectors for each damage pattern in a training process. The training process of the AIPR algorithm is inspired by the clonal selection strategy of the natural immune system. The initial representative feature vectors are generated by the random selection of feature vectors from the training data. The rest of feature vectors are used to stimulate the evolution of representative feature vectors using clonal selection principle.

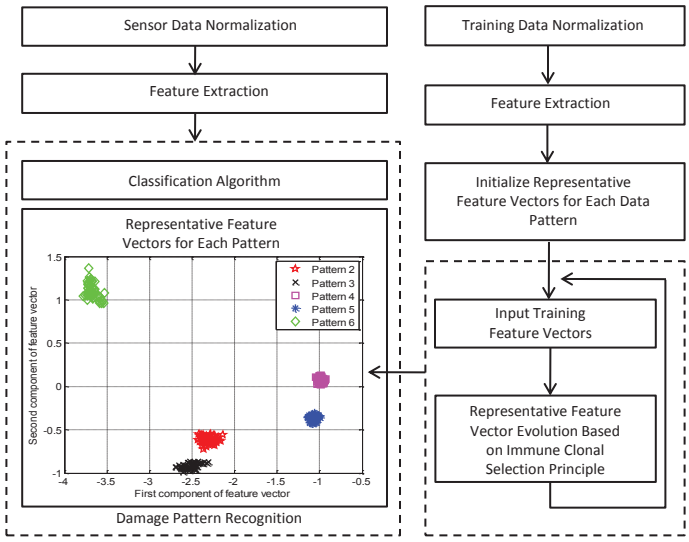


Figure 3.3: AIPR-based pattern classification

Let $f_r = (f_{r1}, f_{r2}, \cdots, f_{rp})^T$ and $f_t = (f_{t1}, f_{t2}, \cdots, f_{tp})^T$ denote a representative feature vector and a training feature vector, respectively. The affinity between these two feature

vectors is defined as

$$aff(f_r, f_t) = 1 - \frac{1}{2}\sqrt{\sum_{i=1}^{p}(f_{ri} - f_{ti})^2} \tag{3.21}$$

The number that a representative feature vector will be cloned, $CloneNum$, depends on its affinity with the training feature vector,

$$CloneNum = round(CR * aff(f_r, f_t)) \tag{3.22}$$

where CR is the clonal rate. The cloned representative feature vectors undergo a process of maturation which increases the diversity of the representative feature vectors using the formula below.

$$f_{r,mutated} = f_r + MV \times \phi \tag{3.23}$$

where $f_{r,mutated}$ is the mutated representative feature vector and MV is the mutation value. The vector $\phi = (\phi_1, \phi_2, \cdots, \phi_p)^T$ is a random vector. Each element in ϕ is defined by $\phi_i \in N(0, \sigma^2)$, where $N(0, \sigma^2)$ is a normal random variable with the standard deviation of σ. The clonal selection principle allows the AIPR algorithm to mutate its representative feature vectors towards matching the training data. The AIPR algorithm can study and store different data patterns by creating representative feature vector clusters for input data patterns.

3.4 Unsupervised Clustering Methods

Unsupervised clustering refers to the process of partitioning unlabeled data into clusters. The cluster is defined as a set of similar objects which are grouped together. If all the objects are represented by feature vectors in a feature space, the distance between any two feature vectors in the same cluster is shorter than the distance from any feature vector in this cluster to any other feature vectors which is not in this cluster [73]. Clustering technique is regarded as unsupervised learning and can be divided into several major categories [5]: sequential algorithms, hierarchical clustering algorithms, clustering algorithms based on cost function optimization and other special clustering techniques. The most popular clustering approaches are k-means clustering, hierarchical clustering and fuzzy clustering.

3.4.1 K-means Clustering

If there are n feature vectors $\{x_i\}_{i=1}^{n}$ in the feature space to be partitioned into k clusters $C_1, C_2, \cdots, C_k$ ($k \leq n$), the centroid of each cluster can be represented as:

$$\mu_i = \frac{1}{n_i} \sum_{j \in C_i} x_j \tag{3.24}$$

where n_i is the number of elements in cluster C_i. The initial clusters are usually randomly selected. If Euclidean distance is used for similarity measure, the sum of the squared error is:

$$E = \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\| \tag{3.25}$$

where μ_i is the mean of the points in the cluster i. So the clustering problem turns to be seeking the minimum value of the error.

Generally, the satisfying partitioning is achieved by iterative calculation [74]: Assign each pattern to the cluster whose centroid this pattern is closest to:

$$C_i^{(t)} = \{x_j : \|x_j - \mu_i^{(t)}\| \leq \|x_j - \mu_{i'}^{(t)}\|, for\ all\ i' = 1, 2, \cdots, k\} \tag{3.26}$$

Calculate the new means of the clusters for the next iteration:

$$\mu_i^{t+1} = \frac{1}{|C_i^{(t)}|} \sum_{x_j \in C_i^{(t)}} x_j \tag{3.27}$$

where t is the iteration index. The iteration stops when there is no change of the pattern assignment.

3.4.2 Hierarchical Clustering

Hierarchical clustering is a technique which can construct a hierarchy of data clusters. Hierarchical clustering includes agglomerative algorithms (bottom up) and division algorithms (top down). In agglomerative algorithms, if there are n sample patterns, there will be totally n levels in the hierarchy and each sample pattern is in a cluster at the beginning. So there are n clusters in the first level of the hierarchy and at each level, two closest clusters are merged, until all the patterns are merged into one single cluster. The division algorithms run are in the opposite direction: the hierarchy starts from one cluster and more clusters are produced at each step by splitting single clusters.

There are multiple ways to calculate the distance between two clusters A and B. One example is to calculate the minimum distance between two points from the two clusters:

$$D(A,B) = min\ distance(a_i,b_j),\ a_i \in A, b_j \in B \tag{3.28}$$

The other example is to calculate the maximum distance between two points from the two clusters:

$$D(A,B) = max\ distance(a_i,b_j),\ a_i \in A, b_j \in B \tag{3.29}$$

3.4.3 Fuzzy Clustering

Conventionally clustering methods classify a studied object exclusively to a single category [5, 75]. However, in some real world problems, this hard clustering technique is insufficient. With fuzzy clustering technique, clusters with vague boundaries can be constructed and one object can belong to more than one clusters to some degree. Suppose that there are n objects $X = \{x_1, x_2, \cdots, x_n\}$ and K number of clusters. The degree of describing how much an object belongs to a cluster can be denoted as:

$$u_{ik} \equiv \mu_k(x_i),\ i = 1, \cdots, n,\ k = 1, \cdots, K \tag{3.30}$$

where

$$u_{ik} \in [0,1], \forall i,k;\ \sum_{k=1}^{K} u_{ik} = 1,\ \forall i \tag{3.31}$$

Fuzzy C-Means is a fuzzy clustering method which minimizes the cost function [75]:

$$J(U, v_1, \cdots, v_K) = \sum_{i=1}^{n} \sum_{k=1}^{K} (u_{ik})^m d^2(x_i, v_k) \tag{3.32}$$

where v_k denotes the centroid of the cluster k and $d^2(x_i, v_k)$ is the square of the Euclidean distance between x_i and v_k. m ranges within $(1, \infty)$ and indicates the degree of fuzziness.

The solutions for the above equation are [75]:

$$u_{ik} = \frac{1}{\sum_{l=1}^{K} \{d(x_i, v_k)/d(x_i, v_i)\}^{\frac{2}{m-1}}}, \; v_k = \frac{\sum_{i=1}^{n} (u_{ik})^m x_i}{\sum_{i=1}^{n} (u_{ik})^m} \tag{3.33}$$

Chapter 4

Mobile Agent and AIS-Based Monitoring Networks

4.1 Mobile-Agent-based Monitoring Networks

As introduced in Chapter 2, mobile agents can transport data and program from one computing node to another and perform the assigned task on the destination platform. This section presents a mobile agent approach for building a monitoring network to reduce data transmission and enhance the flexibility of distributed monitoring systems. Taking advantage of the mobility of a mobile agent system, the presented monitoring network allows moving diagnosis programs to data sources and performing abnormal condition

diagnosis locally as shown in Figure 4.1. The distributed sensor nodes can dynamically accept mobile agents for the deployment of new abnormal condition diagnosis algorithms and sensing strategies in response to the changes of monitoring conditions. In a mobile agent-based monitoring network, mobile agents can be sent to the hardware platforms in the network by the remote user. Mobile agents can travel from one hardware platform to another with program and its execution state. After execution of the assigned task, the results are sent back to the users. To identify each agent, an identification number accompanies the mobile agent throughout its lifetime. To support mobile agent generation,

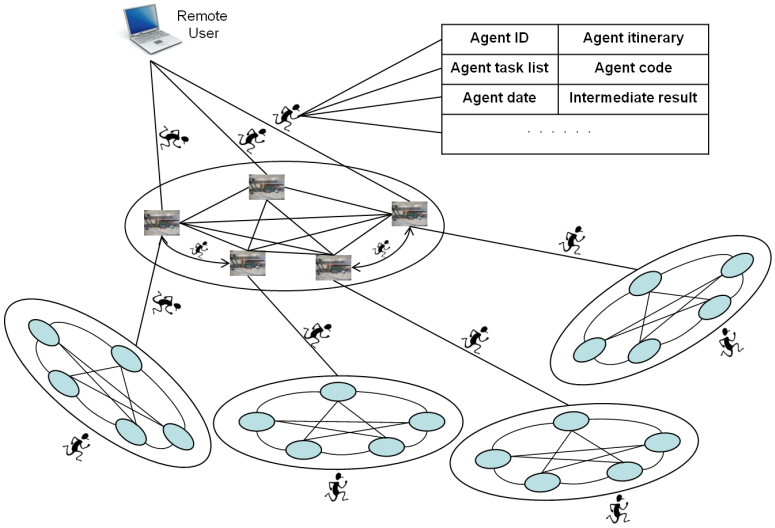


Figure 4.1: A mobile agent-based monitoring network

migration, execution, and management, the presented mobile agent based sensor network platform is developed based on Mobile-C [3, 41, 76]. In a mobile-agent-based monitoring network, mobile agents of different types could be generated and sent to hardware platforms according to users' requests. These agents will patrol over the network and perform the assigned tasks as requested.

4.2 AIS-based Monitoring Networks

The main features of the immune system include adaptive immune response to the invading pathogens and pattern recognition capabilities. When an invading pathogen is detected by the host, corresponding biological response will happen on different levels of biological organization [77]. Immune cells, such as B cells, T cells, and antigen presenting cells, will be activated [78]. In the process of adaptive immune response, the amount of B cells can increase due to one kind of specific antigen and the newly created B cells can generate more antibodies to bind with the intruders [79].

The advantages of the immune system, such as adaptation, evolution, and fault tolerance, have inspired the applications of immune concept for monitoring networks. In AIS-based monitoring networks, a mobile monitoring agent is selected for cloning when it detects abnormal condition in a sensor unit. Multiple copies of this type of monitoring agents will be created. As a result, the type and amount of mobile monitoring agents are adapted to detected abnormal pattern. The framework of an AIS-based monitoring network consists of (1) an agent-based network middleware (mobile agent system) [3, 80] to support the generation, execution, migration of mobile monitoring agents; (2) clonal selection algorithm for the cloning of mobile monitoring agents who detect abnormal condition in distributed sensor units; (3) knowledge base for keeping updated representative feature vectors (memory cells) for normal and abnormal condition patterns

and performing confirmation of abnormal condition detected by a mobile monitoring agent; (4) agent interaction protocols, for example, agent communication protocols amongst mobile monitoring agents, knowledge base agent, and the clonal selection agent for cloning a mobile monitoring agent.

4.3 Mobile Agent Network Platform

To integrate a mobile agent system with sensor nodes and provide high computational power for anomaly detection, an embedded computer-based wireless sensor node integrated with C numerical libraries is developed to form a mobile-agent-based network.

4.3.1 Sensor Node Design

4.3.1.1 Hardware Design

Sensor nodes are building blocks of wireless sensor networks. For the SHM sensor networks, the desirable characteristics of sensor nodes are as follows. First, high computational power sensor nodes are highly recommended. Local data processing can reduce the raw data transmission over a network. The reduction of data transmission can save network bandwidth and energy [81]. Second, open software implementation

is desirable to promote software reuse. The open software architecture allows user communities to participate in improving node functionalities and developing new software. Third, multimodality sensors help to achieve a better assessment of the structural state from a comprehensive view of the structure. Finally, reprogrammable sensors are welcome to increase the adaptability and support the multitasking purposes.

Having the aforementioned node design criteria in mind, we chose a finger size embedded computer called Gumstix [82] as sensor node computing platform. The sensor node consists of three boards: The sensing board lies at the bottom; the Gumstix board is located in the middle; and a wireless communication board sits at the top. Three boards are connected together through predesigned connectors. The Gumstix board communicates with the sensing board through I^2C bus, and connects to the wireless communication board through a parallel port. The volume of the sensor node is about $4 \times 2.4 \times 0.65\, in^3$.

4.3.1.2 Software Design

Targeting for embedded applications, Gumstix does not include an on-board compiler. The Gumstix application software is usually developed on a host computer using a cross-compiler, and then, downloaded to the Gumstix. After setting up the Gumstix, Gumstix can program the sensor board microcontroller through I^2C-load and in system programmer (UISP).

To facilitate the implementation of damage diagnosis algorithms on sensor nodes, a number of numerical libraries are integrated into sensor nodes. Thanks to the open source software packages Ch, CLAPACK [83], and Numerical Recipes in C [84], which make it easy to perform damage diagnosis on sensor nodes and build an open software architecture. Ch is an embeddable C/C++ interpreter which supports all the standard libraries and features of the ISO C90 Standard [53].

The lower layer embedded software manages the peripherals of ATmega128L, such as UART, SPI, I^2C, and on-chip ADC. Impedance measurement uses I^2C module to transmit data and send commands, while passive sensing module communicates with the microcontroller via SPI bus. The passive data acquisition is handled in a timer interrupter processing module. The active sensing and the humidity and temperature data are processed in the microcontroller main control loop.

To support the execution and migration of mobile monitoring agents, an embeddable mobile agent system, mobile-C, is integrated with the sensor unit as shown in Figure 4.2. Mobile-C is an IEEE FIPA [85] compliant mobile agent system supporting mobile C/C++ agents. It has a small footprint and is easy to be integrated with resource-constrained systems, such as sensor networks. The Mobile-C in sensor nodes can host both stationary agents and mobile agents. Stationary agents are those staying in the sensor nodes where they are created, such as data acquisition agents and regional or central management agents. Mobile agents are those created during the system operation and able to move to different

sensor nodes in a network. The agencies are the main components of the Mobile-C

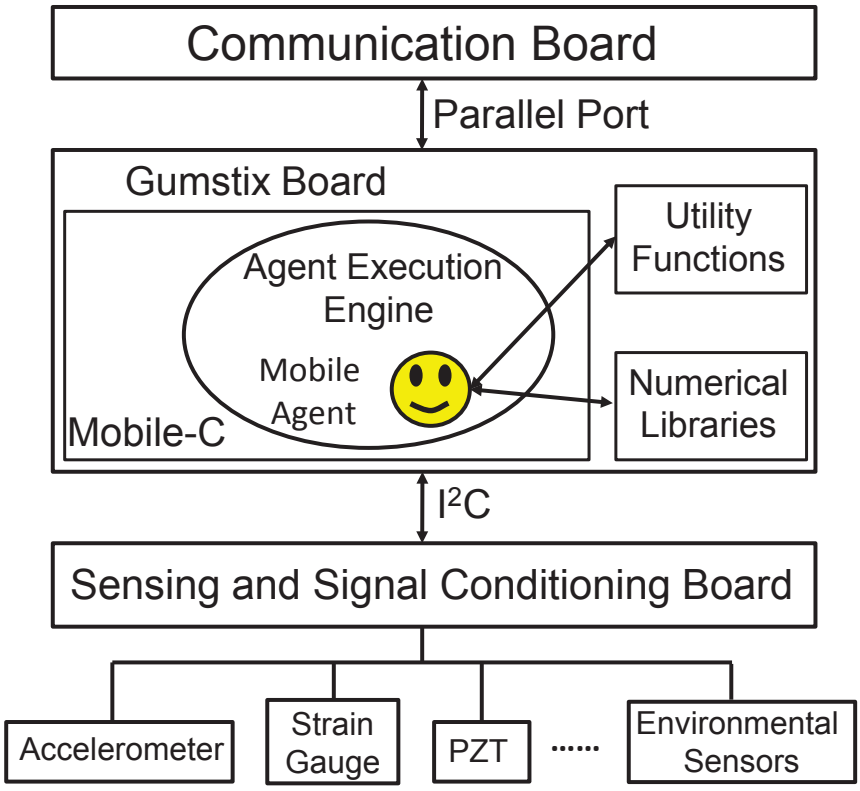


Figure 4.2: A sensor node integrated with a mobile agent middleware

agent system and the runtime environment of both stationary and mobile agents [3]. Ch [53], a C/C++ interpreter, is adopted as the agent program execution engine in Mobile-C. The advantages of using C/C++ as the agent language include [3]: (1) C/C++ is highly structured language with many powerful function sets; (2) C/C++ has been popular for years, so a large number of existing programs can be reused in Mobile-C; (3) C language is able to interface with hardware. Because Ch is not able to capture and record the execution state of a process, so the agent migration in Mobile-C is labeled as weak migration which only allows the data state to be transferred between hosts. In Mobile-C, XML language

carries the responsibility of conveying agents which are composed of data states and C/C++ code and the migration messages are encoded into XML code. The task for an mobile agent can be divided into subtasks which are saved into the task list. The task list can be updated dynamically due to the changing user requirement. The subtasks can be performed on different hosts, however, once the subtask begins to be executed on one host, the agent has to stay on this host till the ongoing subtask is finished.

4.3.1.3 Lifetime Evaluation of Sensor Nodes

For energy efficiency, the sensor node is designed to operate in two different configurations: with Gumstix and WiFi boards (three boards) or sensor board only. For agent or web-based applications, three boards configuration is required. For simple structural data collection, only one sensor board is needed. In a typical application, two-level network architecture can be employed using these two configurations. The upper-level sensor nodes can use configuration 1 (three boards) while the lower-level sensor nodes can use configuration 2 (senor board only).

Figure 4.3 shows the lifetime of the sensor node (configuration 2) if the sensor node is powered by four 1.2V 2900mAh AA batteries. The sensor board performs passive and active sensing periodically. Each passive sensing event collects 7500*4*2 byte data and lasts one minute. IC chips on the sensor board including microcontroller, AD converter, active sensing signal generator and response analyzer, SRAM, SHT15 sensor, and ZigBee

module, can work in either active or power saving mode. The power supply voltage and current of these chips in the active/power saving mode are listed in Table 4.1. When the sensor board is performing passive/active sensing, the corresponding chips are in the active mode. These chips enter power saving mode when the board is not collecting data. The term "duty cycle" represents the ratio of the duration of active mode to a period of 10 minutes. The duty cycle is 10% if one passive sensing event happened within 10 minutes. Figure 4.3 shows that the lifetime of the sensor board is 863 hours when the duty cycle is 10% and the sensor board does not perform active sensing. The lifetime of the sensor board reduces to 166 hours when the sensor board performs continuous passing sensing. The impact of the active sensing to the lifetime of the sensor board is also shown in Figure 4.3. If the sensor board performs active sensing once every hour, the lifetime of the sensor node decreases to 717 hours at 10% duty cycle and 119 hours for continuous passing sensing. In configuration 1 (three boards), the Gumstix and WiFi boards work with 5V DC power supply. The Gumstix board draws 270 mA in sensing mode and 60 mA in power saving mode. The current of the WiFi board is 80 mA. The lifetime of the sensor node in configuration 1 is 19 hours at 10% duty cycle and 7.6 hours for continuous passing sensing.

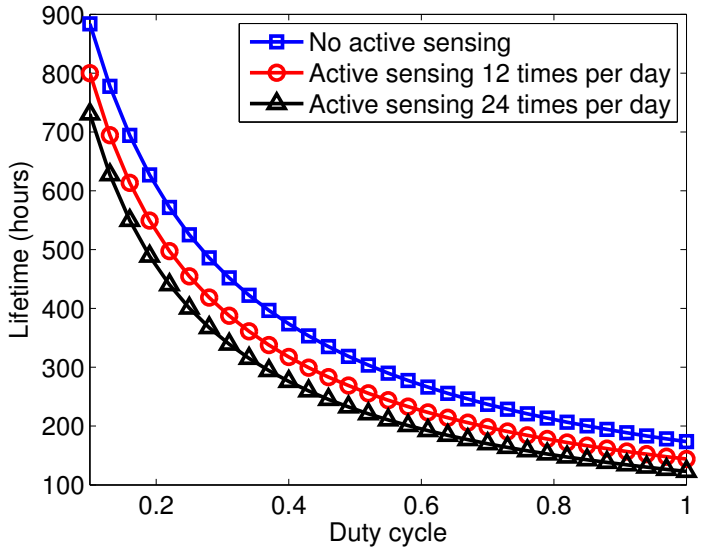


Figure 4.3: The lifetime of the sensor board vs the duty cycle of the data collection

Table 4.1
The power supply voltage and current of chips on the sensor node

Component	Voltage (V)	Current (mA) Active/Power saving mode
ATmega128	3.3	5/2
ADC (QF4A512)	3.3	17/0.6
Active sensing chip (AD5933)	3.3	10/0.0007
SRAM (CY62138CV30LL)	3.3	1.5/0.0002
Temperature and humidity (SHT15)	3.3	0.55/0.0003
ZigBee module	3.3	35/0.001
Accelerometer	3.3	0.95

Chapter 5

Optimal Control of Agent Networks[1]

This chapter presents multi-objective optimization algorithms for the optimal control of mobile monitoring agents in artificial-immune-system-based monitoring networks. To minimize the response time for the generated monitoring agents to diagnose abnormal condition and maximize the average affinity of the feature vectors of monitoring agents and the abnormal sensor data feature vectors, a multi-objective genetic algorithm is developed to find appropriate number of agent clones and the mutation rate for the cloned monitoring agents. In the distribution optimization of mobile monitoring agents, sensor data feature vector and the remaining battery capacity of sensor nodes are taken into consideration to increase the detection probability and extend the lifetime of the monitoring network.

[1] ©Elsevier. Portions reprinted with permission, from W. Liu and B. Chen, "Optimal control of mobile monitoring agents in immune-inspired wireless monitoring networks", Journal of Network and Computer Applications, vol. 34, no. 6, pp. 1818-1826, 2011.

5.1 Genetic Algorithms and Multi-Objective Optimization

Genetic algorithm is a population-based optimization algorithm which maintains a population of candidate solutions. Due to its population-based nature, it belongs to the family of global optimization algorithms. Genetic algorithm uses techniques inspired by natural evolution to search the solutions for the optimization problems. The solution search process consists of steps such as mutation and crossover which are inspired by the natural evolution. The genetic algorithm starts searching from a group of initial solution population. The initial solutions are randomly generated. Each solutioni is encoded into a chromosome. The data structure of a chromosome is an array. The first generation of solutions are evaluated and sorted by their performance. Then several top solutions are picked to go through the process of crossover and mutation. The generated new solutions will be evaluated with their parent solutions. The algorithm repeats above steps until a stopping criterion is met.

Genetic algorithms have been widely applied to many optimization problems. In a sensor network for agriculture application [86], genetic algorithm is adopted to design wireless sensor network topologies by optimizing parameters which are related to the power consumption of the sensors. Parameters of network connectivity and application

requirements are also considered so an integrated network is designed. Khanna et al. [87] design a reduced-complexity genetic algorithm to minimize the power consumption of the sensor system while maximize the sensing coverage. The genetic algorithm runs periodically to assign functions to the randomly deployed sensor nodes. In [88], a genetic algorithm is applied for data dissemination by creating energy-efficient clusters. The authors proposed an intelligent hierarchical clustering protocol which has better performance than the traditional cluster-based protocols. Igor et al. [89] introduce basic and advanced genetic algorithm implementations and applications in information fusion. Genetic algorithm is also employed in the work of node deployment for wireless sensor networks [90]. The GA system decides which sensor nodes should be active and which sensor node should work as the cluster head in the network. To maximize network lifetime, a regression model [91] is applied to find out an energy-efficient configuration for the genetic algorithm to optimize resource allocation. The reason genetic algorithm is adopted in this work is it can quickly search a large group of solution candidates and handle multiple constraints and objectives.

Multi-objective optimization method is used to control agent generation and distribution in this study. A multi-objective optimization problem has a number of objective functions that need to be minimized or maximized under a number of constraints. The general form

of a multi-objective optimization is described below [92].

$$
\begin{aligned}
&Minimize/maximize\ f_m(X), m = 1,2,\cdots,M; \\
&subject\ to\ g_j(X) \geq 0, j = 1,2,\cdots,J; \\
&h_k(X) = 0, k = 1,2,\cdots,K; \\
&x_i^{(L)} \leq x_i \leq x_i^{(U)}, i = 1,2,\cdots,n
\end{aligned}
\tag{5.1}
$$

where $f_m(x)$ are M number of objective functions; $g_j(x)$ and $h_k(x)$ are inequality and equality constrains; and the n solutions $X = (x_1, x_2, \cdots, x_n)^T$ are restricted by lower and upper boundaries $x_i^{(L)}$ and $x_i^{(U)}$.

There exist a number of multi-objective optimization implementations. A fast and elitist multi-objective genetic algorithm, NSGA-II [93], is one of those algorithms which has been applied to many applications. NSGA-II employs non-dominated sorting incorporating elitism. The goals of NSGA-II algorithm are to find a population of solutions that are as close as possible to the Pareto-optimal front and as diverse as possible. The NSGA-II algorithm consists of four steps. The first step combines parent and offspring, and performs a non-dominated sorting to identify different front for each solution. The first front is a non-dominated set in the current population and the second front is dominated by the individuals in the first front, and so on. In the second and third steps, a new parent set is

formed based on the rank of front level and the crowding distances. The solutions with higher order of front levels will firstly be selected into new parent set. The selection of solutions at same front level is based on crowding distances. The solutions with large crowding distances will be included to increase the diversity of the solution population. In the forth step, the selected parents generate offspring by using the crowed tournament selection, crossover, and mutation operations.

5.2 Optimization of Agent Generation

5.2.1 Agent Generation in an AIS-based Monitoring Network

In AIS-based monitoring networks, adaptive immune response is a mechanism to manage the amount and type of monitoring agents through clonal selection. Figure 5.1 shows the clonal selection process in which the activated mobile monitoring agent is cloned and mutated. The cloned mobile monitoring agents are sent to sensor nodes close to the location where the abnormal condition is detected to find out the abnormal condition affected area. Assume that the local communication uses ZigBee and the long distance communication uses WiFi. The cloned mobile monitoring agents will visit N number of sensor nodes for further abnormal condition diagnosis. The number of cloned monitoring agents, n, depends on the clonal rate CR and the affinity between the feature vector of the mobile monitoring

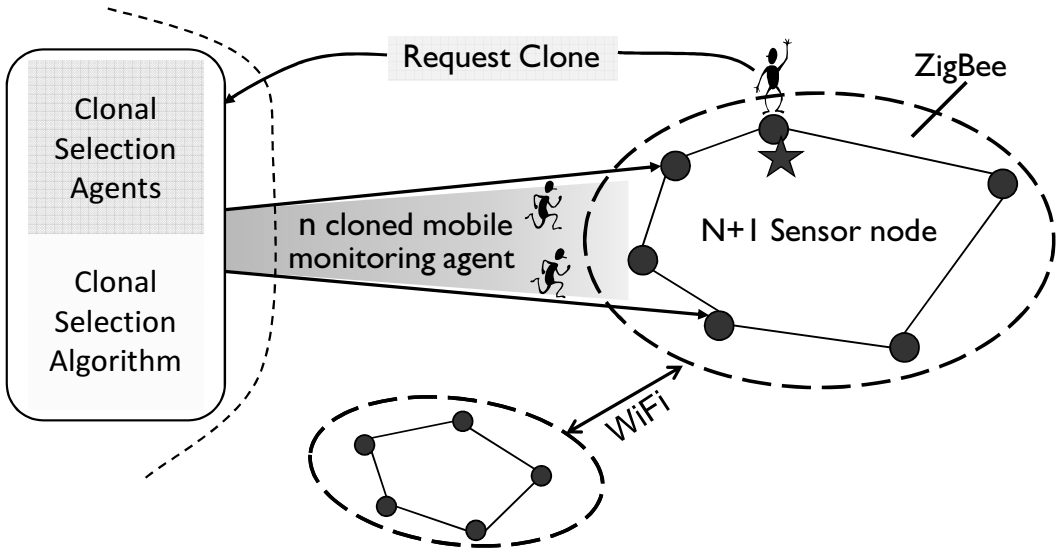


Figure 5.1: Artificial immune response

agent and the sensor data feature vector where the abnormal condition is detected. The number of cloned agents can be calculated by equation 5.2

$$n = round(CR * aff(\beta, \alpha)) \tag{5.2}$$

where the CR is an integer value used to control the number of agent clones allowed for the activated mobile monitoring agent. Let $ma.f = \beta = (\beta_1, \beta_2, \cdots \beta_p)$ and $sd.f = \alpha = (\alpha_1, \alpha_2, \cdots \alpha_p)^T$ denote the feature vector of a mobile monitoring agent and the feature vector of the sensor data, respectively. The affinity between the feature vector of the mobile monitoring agent and the sensor data feature vector is defined as

$$aff(\beta, \alpha) = 1 - \frac{1}{2} \times dist(\beta, \alpha), \tag{5.3}$$

where $dist(\beta, \alpha)$ is the Euclidian distance between vectors α and β as shown in Equation 5.4.

$$dist(\beta, \alpha) = \sqrt{\sum_{i=1}^{p} (\beta_i - \alpha_i)^2} \tag{5.4}$$

The cloned monitoring agents are mutated to increase the diversity of the generated mobile monitoring agents. The mutation is performed by mutating the feature vectors of the cloned monitoring agents. Let $(ma_{mutated}).f = \gamma = (\gamma_1, \gamma_2, \cdots \gamma_p)^T$ denote the feature vector of a mutated monitoring agent. The mutation is performed as shown in equation 5.5

$$\gamma = \beta + MV * (\phi_1, \phi_2, \cdots \phi_p)^T \tag{5.5}$$

where MV is the mutation value; and ϕ_i is a normal random value in the range of $[-1, 1]$. The goal of the presented multi-objective optimization algorithm is to find appropriate values of CR and MV with following objective functions.

1). Minimize response time: the response time is defined as the time needed for the cloned mobile monitoring agents to further diagnose structural abnormal condition at the N number of sensor nodes close to the location where the abnormal condition is detected. The response time includes both mobile agent transmitting time and abnormal condition

diagnosis computational time

$$T = nsT_w + T_c + [\frac{(N - N\%n)}{n} - 1] * ST_z + [\frac{(N - N\%n)}{n} - 1] * T_c + \lceil \frac{N\%n}{n} \rceil * [ST_z + T_c] + T_{ga} \tag{5.6}$$

where T_c is the computational time for a mobile agent to execute abnormal condition diagnosis program in a sensor node. T_w and T_z are the time needed for transmitting one byte of data in WiFi and ZigBee networks. T_{ga} is the time need for the genetic algorithm to find non-dominated solutions. S is the size of a mobile agent in bytes. The number of cloned mobile monitoring agent is n. Modulus function $N\%n$ finds the remainder of the division N/n. Since $n < N$, n number of cloned agents needs to move $[\frac{(N-N\%n)}{n} - 1]$ times in the ZigBee network and part of the cloned agents needs one more move to cover N number of sensor nodes.

2). Maximize average affinity level (minimize average distance) among the feature vectors of cloned mobile monitoring agents and the sensor data feature vector:

$$D = \frac{1}{n} \sum_{i=1}^{n} dist((ma_{i,mutated}.f, sd.f)) = \frac{1}{n} \sum_{i=1}^{n} dist(\gamma_i, \alpha) \tag{5.7}$$

where $dist((ma_{i,mutated}.f, sd.f))$ is the Euclidean distance between the feature vectors of the ith cloned mobile monitoring agent and the sensor data feature vector.

3). The optimization is subjected to following constraints:

$$n \leq N;\ 0 < CR \leq 30;\ 0 < MV < 1 \tag{5.8}$$

The range of CR is determined from previous research results. When the value of CR is less than 30, the number of memory cells is within a reasonable range.

5.2.2 Solutions for the Optimization of Agent Generation

To find out non-dominated solutions for defined objective functions, the fast and elitist multi-objective genetic algorithm, NSGA-II, is used for the simulation of this optimization problem. The values of simulation parameters are listed in Table 5.1. The non-dominated solutions found by the NSGA-II algorithm are shown in Figure 5.2. The corresponding solution values form a decision space as shown in Figure 5.3. From the decision space, we can see that when the mutation value MV is high, the value of clonal rate CR is relative small. When the MV value is small, the CR value is high. Because these solutions are non-dominated, users can decide which solution will be adopted based on their specific needs.

Table 5.1
Description of progressive abnormal condition tests

Simulation parameters	Value	Unit
Population size	40	
Number of generations	2000	
Crossover probability	0.6	
Mutation probability	0.5	
T_c–computational time for a mobile monitoring agent to execute abnormal condition diagnosis program	55	s
T_w–time needed for transmitting one byte of data in WiFi network	0.727	μs
T_z–time needed for transmitting one byte of data in ZigBee network	0.032	ms
T_{ga} – –time needed for the genetic algorithm to find non-dominated solutions	4	s
The size of a mobile monitoring agent in bytes,S	1900	byte

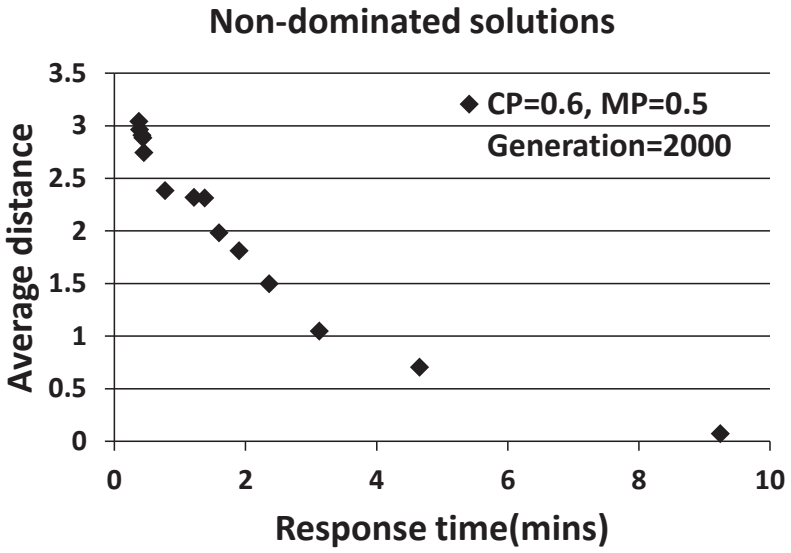


Figure 5.2: Non-dominated solutions

To find out good solutions for the system, the impact of algorithm parameters on the system performance is investigated. Two criteria are usually used to evaluate the goodness of solutions [92]: (1) the non-dominant solutions should be as close to the Pareto-front as possible; (2) the non-dominated solutions are as diverse as possible. The diversity helps

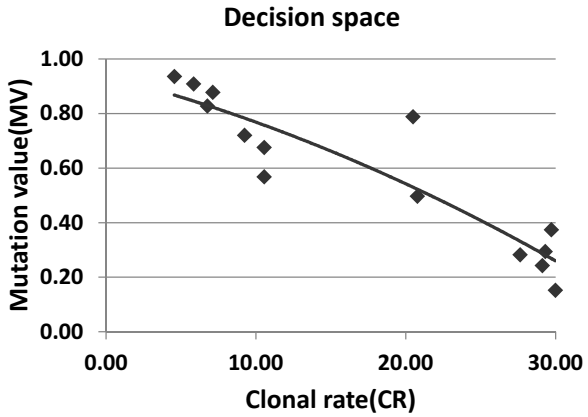


Figure 5.3: Decision space

to achieve more uniform distribution of non-dominated solutions along the Pareto front. Figure 5.4 shows that non-dominated solutions move towards the origin as the number of generation increases. This means that the larger the number of generations, the closer the non-dominated solutions to the Pareto-optimal solutions. The distance between consecutive solutions can be used to evaluate how well the solutions are uniformly spaced [92]. The spacing metric S is defined as follows.

$$S = \sqrt{\frac{1}{|Q|}\sum_{i=1}^{|Q|}(d_i - \overline{d})^2} \tag{5.9}$$

Where Q is the number of non-dominated solutions, $d_i = min_{k \in Q \wedge k \neq i} \sum_{m=1}^{M} |f_m^i - f_m^k|$ and $\overline{d}$ is the mean value of the above distance measure $\overline{d} = \sum_{i=1}^{|Q|} d_i / |Q|$. In the equation of d_i, M represents the number of objective functions. In our application, f_1 represents response time and f_2 represents average affinity level. A smaller value of S indicates more uniform spacing of non-dominated solutions. Figure 5.5 shows that the spacing decreases when the number of generations increases. When the number of generations is above 1200, the value

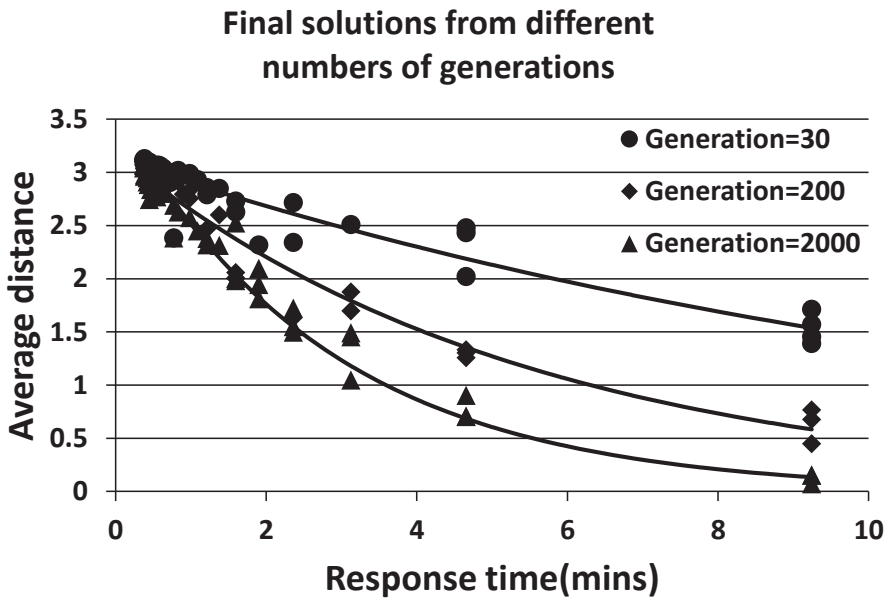


Figure 5.4: Final solutions with different number of generations

of *S* remains at similar level. In addition to number of generations, the mutation rate of the genetic algorithm also impacts the spacing of the solutions. Comparing with Figure 5.6 and Figure 5.7, the larger value of the mutation rate has better solution spacing. This is because the larger mutation probability improves the diversity of solutions.

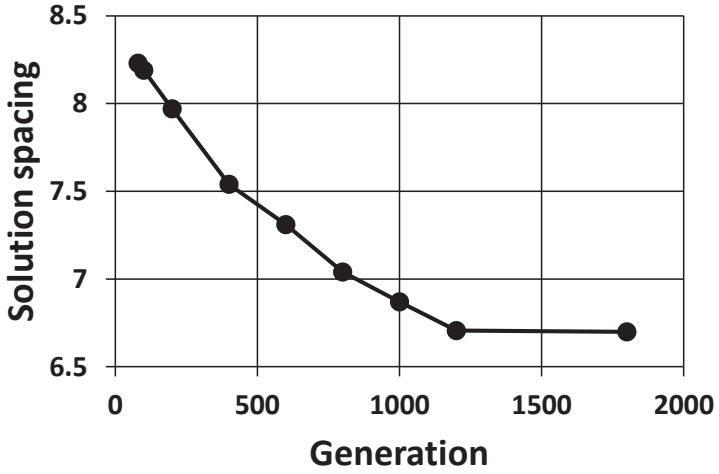


Figure 5.5: Spacing of non-dominated solutions vs. number of generations

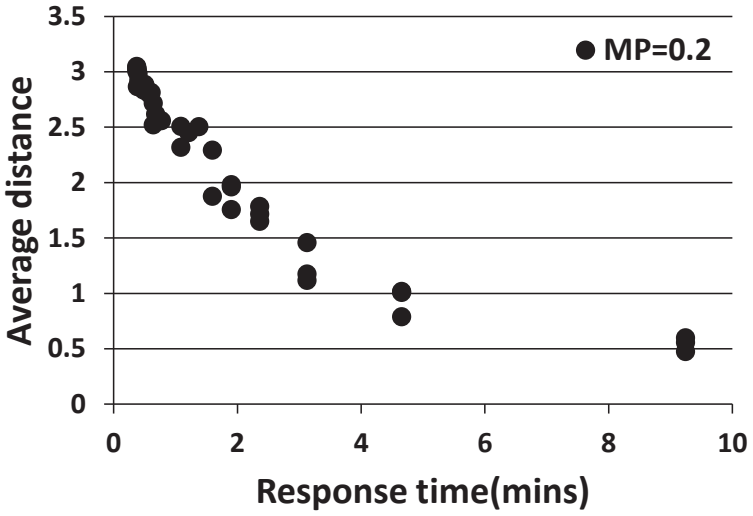


Figure 5.6: Final solutions when mutation probability is 0.2

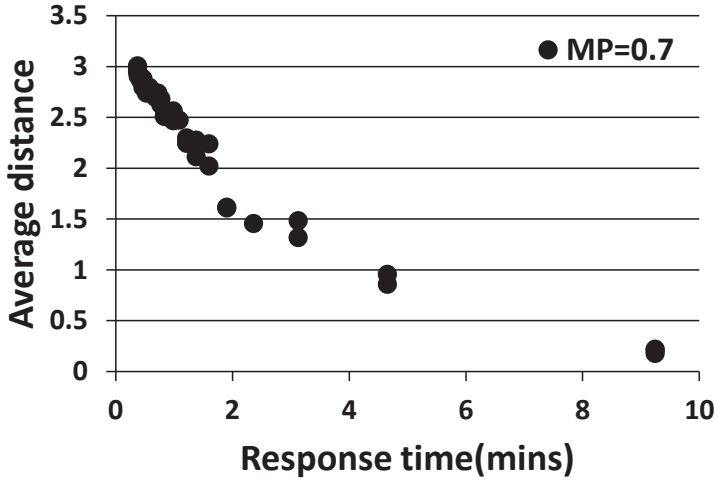


Figure 5.7: Final solutions when mutation probability is 0.7

5.3 Optimization of Agent Distribution

Optimization of agent distribution is to find out which sensor node is the best candidate for a mobile monitoring agent to visit so that the abnormal condition detection probability is high

and monitoring network lifetime can be extended. Due to energy constraints, each sensor node has its own lifespan. The remaining battery capacity in each sensor node impacts the lifetime of the entire monitoring network. The goal of the optimization algorithm is to distribute mobile monitoring agents to the node where they are needed for abnormal condition detection and prolong the network lifetime. All the sensor nodes keep collecting data from surroundings at a fixed frequency. Figure 5.8 shows the basic concept of the presented GA algorithm for optimizing agent distribution. Assume that there is more than one sensor node in the region where the abnormal condition is occurred. Based on the fitness values (the distances amongst the sensor data feature vectors and the feature vectors of a mobile monitoring agent), the GA algorithm determines on which sensor node the mobile agent has the highest probability to detect anomaly. The algorithm also takes the remaining battery capacity of each sensor node into consideration for the agent distribution decision. Sensor nodes with low remaining battery capacities have low privilege to accept a mobile monitoring agent. This distribution strategy extends the overall lifespan of a monitoring network.

5.3.1 Genetic Algorithms for Energy Balanced Agent Distribution

A GA-based algorithm is developed to dynamically search a destination sensor node for a mobile monitoring agent based on the remaining battery capacity of sensor nodes and the sensor data feature vectors. The selected sensor node satisfies two criteria: (1) the

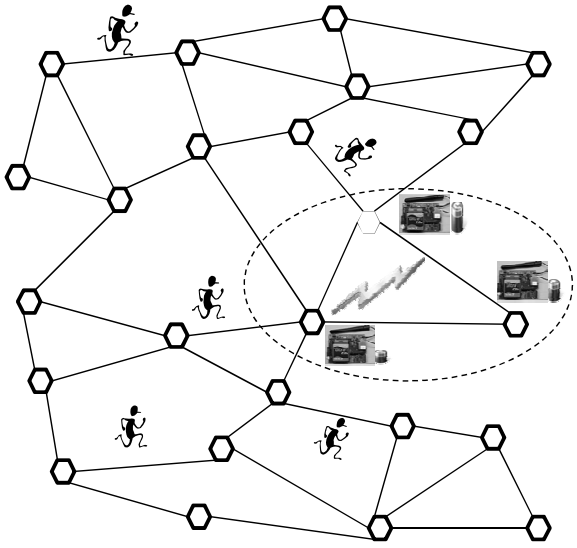


Figure 5.8: Agent distribution based on affinity and remaining battery capacity of sensor nodes

Euclidean distance between the sensor data feature vector and the feature vectors of the mobile monitoring agent is small; (2) the remaining battery capacity of the sensor node is higher than a pre-defined threshold. Figure 5.9 shows a GA-based algorithm for a mobile monitoring agent to search a candidate sensor node to visit. The sensor data feature vectors and the remaining battery capacities of sensor nodes are used to choose an appropriate sensor node. The output of the genetic algorithm is the ID of selected sensor node. Sensor IDs are encoded into chromosomes in binary format. In the experimentation discussed in this dissertation, 8 digits of binary code are used to represent the sensor IDs. Initial population with 10 individuals is randomly picked up in the sensor node population pool. Each individual is an 8-bit binary string. The fitness function of the individuals is calculated based on the Euclidian distance between the sensor data feature vector and the feature vector of a mobile monitoring agent. Let $\beta_i = (\beta_{i1}, \beta_{i2}, \cdots \beta_{ip})^T$ denote the feature vector

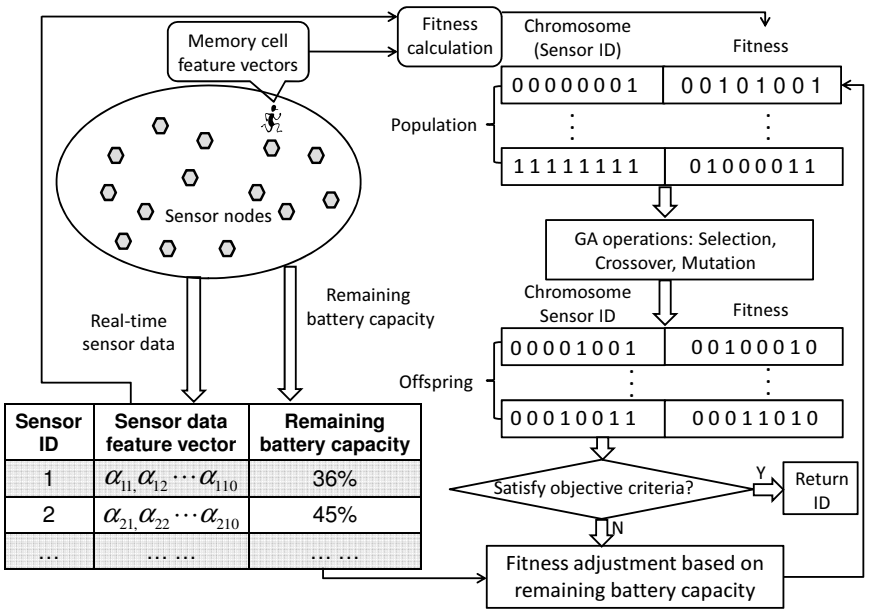


Figure 5.9: Genetic algorithm for searching a suitable sensor node

of the *i*th mobile monitoring agent and $\alpha_j = (\alpha_{j1}, \alpha_{j2}, \cdots \alpha_{jp})^T$ the sensor data feature vector of the j th sensor node. The Euclidian distance between these two feature vectors can be calculated by Equation 5.10.

$$dist(\beta_i - \alpha_j) = \sqrt{(\beta_{i1} - \alpha_{j1})^2 + (\beta_{i2} - \alpha_{j2})^2 + \cdots + (\beta_{ip} - \alpha_{jp})^2} \tag{5.10}$$

Based on fitness values, several sensor individuals with high fitness to the feature vectors of the mobile monitoring agent are selected. The selected sensor individuals subject to two GA operations: crossover and mutation, to generate offspring of the parent generation. The impact of crossover probability and mutation rate on the number of iterations to find a destination sensor node and the ration of good solutions is discussed later. To balance energy consumption among sensor nodes in a monitoring network, a fitness punishment

strategy is designed in the genetic algorithm. If a sensor node whose remaining battery capacity is below a pre-defined threshold, the fitness value of the sensor node is reduced by multiplying a penalty factor. This punishment greatly impacts the fitness value of the sensor node; as a result, the sensor node will most likely be eliminated from the population of the new generation.

5.3.2 The Remaining Energy Model of a Sensor Node

For battery-powered sensor nodes, the remaining battery capacity of a sensor node can be calculated by equation 5.11 [94]:

$$RC = SOC \cdot SOH \cdot DC \tag{5.11}$$

where *SOC* (state of charge) indicates how much energy left in a battery; *SOH* (state of health) is the ratio of the full charge capacity of a battery to its design capacity (*DC*). The equations for calculating *SOC*, *SOH*, and *DC* are shown in equations 5.12, 5.15, and 5.16 [94]. The SOC is defined below.

$$SOC = 1 - \frac{[\frac{1}{b_1} - (\frac{1}{b_1} - SOH^{b_2} \cdot DC^{b_2}) exp(\frac{\Delta v_m - \Delta v}{\lambda})]^{\frac{1}{b_2}}}{SOH \cdot DC} \tag{5.12}$$

where $\Delta v = VOC_{init} - v$ and $\Delta v_m = VOC_{init} - v_{cut-off}$, VOC_{init} indicates initial open-circuit voltage; $v_{cut-off}$ indicates a voltage threshold below which the end of discharge is

considered being reached; $\lambda = (2RT)/(nF)$, where R indicates gas constant; T indicates current temperature; $n$indicates number of electronics transferred; and F indicates Faraday's constant; In equation 5.12, b_1 ,b_2 are functions of discharge rate and temperature, respectively [94],

$$b_1(i,T) = d_{11}exp(\frac{d_{12}}{T}) + d_{13} \tag{5.13}$$

$$b_2(i,T) = (\frac{d_{21}}{T + d_{22}}) + d_{23} \tag{5.14}$$

where $d_{jk}(i) = \sum_{z=0}^{4} m_z(d_{jk}) \cdot i^z, j = 1,2, k = 1,2,3$, and $m_z(d_{jk})$ are constant coefficients based on curve fitting. The values of SOH and DC can be calculated by equations 5.15 and 5.16,

$$SOH = \{\frac{1 - exp(\frac{r_n \cdot i - \Delta v_m}{\lambda})}{1 - exp(\frac{r_0 \cdot i - \Delta v_m}{\lambda})}\}^{\frac{1}{b_2}} \tag{5.15}$$

$$DC = \{\frac{1}{b_1}[1 - exp(\frac{r_0 \cdot i - \Delta v_m}{\lambda})]\}^{\frac{1}{b_2}} \tag{5.16}$$

where $r_n = r(i,T,n_c,T')$ represents the effect of the Ohmic overpotential and electrode reaction potential when the discharged current is constant; i indicates current; T indicates

the current temperature; T' indicates the temperature that the battery has experienced in previous cycle; and n_c is the number of electrons transferred during the process of battery charging; if $n_c = 0$,$r_n = r_0$. Based on above equations, the remaining capacity of a battery can be calculated with three measurable variables: battery voltage v, current i, and temperature T.

5.3.3 Effects of Fitness Punishment Strategy on the Network Lifetime

To analyze the impact of the fitness punishment strategy based on remaining battery energy on the network lifetime, a simulation is performed to find out appropriate penalty factor values and remaining energy threshold. Assume that 255 sensor nodes are deployed evenly in a $15 \times 15m^2$ area. Each sensor node collects measurement data from sensors. The sensor network periodically dispatches mobile monitoring agents to the network for distributed anomaly detection. The total energy consumption of the network includes sensing energy and mobile agent deployment energy E_a. The energy consumed for deploying a mobile agent consists of mobile agent transmitting energy and computation energy. In our monitoring network, collecting 2000 data bytes of acceleration data for abnormal condition diagnosis consumes 1.296 J. The energy consumption for transmitting a mobile agent to a sensor node and performing abnormal condition diagnosis consumes 183.75 J. Since the agent deployment energy is much higher than sensing energy, only agent deployment energy consumption is considered in this simulation. Figure 5.10 shows the percentage of

network lifetime extension vs. the values of penalty factor and the energy threshold for applying fitness punishment.

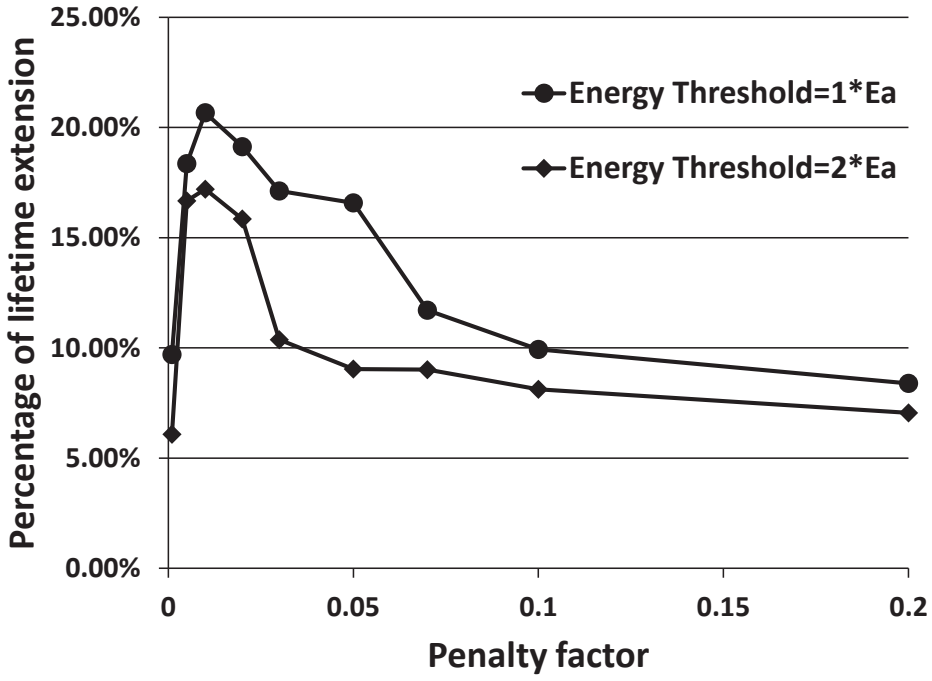


Figure 5.10: Percentage of lifetime extension vs. penalty factors

5.3.4 The Impact of Genetic Algorithm Parameters on the Good Solution Ratio and the Number of Iterations

The presented GA algorithm is implemented by the Genetic Algorithm Toolbox for Matlab from University of Sheffield [95].

To investigate the impact of parameters on the performance of the algorithm, different

values of mutation probability and crossover rate are used to test the ration of good solutions and the convergence speed. The good solution is the sensor ID the data collected by which have the highest affinity with the mobile agent and the remaining energy is above the predefined threshold. Figure 5.11 shows the impact of the value of mutation probability on the good solution ratio with a fixed crossover rate. The horizontal axis represents the number of generations and the vertical axis represents the good solution ratio. Good solution ratio is defined as the ratio of the number of good solutions to the size of population. The crossover rate used in this test is equal to 0.7. From Figure 5.11, we can see that the ratio of good solutions increases quickly in the first 30 generations, but, no significant change after that.

Figure 5.12 shows the average number of iterations the genetic algorithm takes to find the best candidate sensor node. The horizontal axis represents the mutation probability and the vertical axis represents the average number of iterations from 12 trials. The curves with different colors are corresponding to different values of crossover rate. From Figure 5.12, we can see that the number of iterations increases significantly when the mutation probability varies from 0.04 to 0.06. For each crossover rate, the average number of iterations increases as the mutation probability increases.

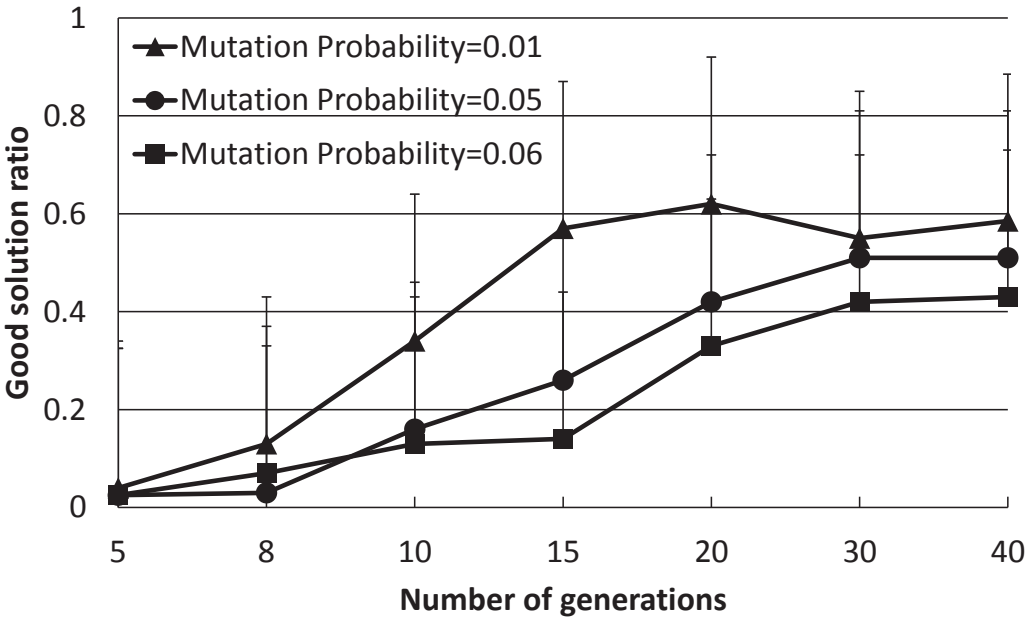


Figure 5.11: Good solution ratio vs. mutation probability

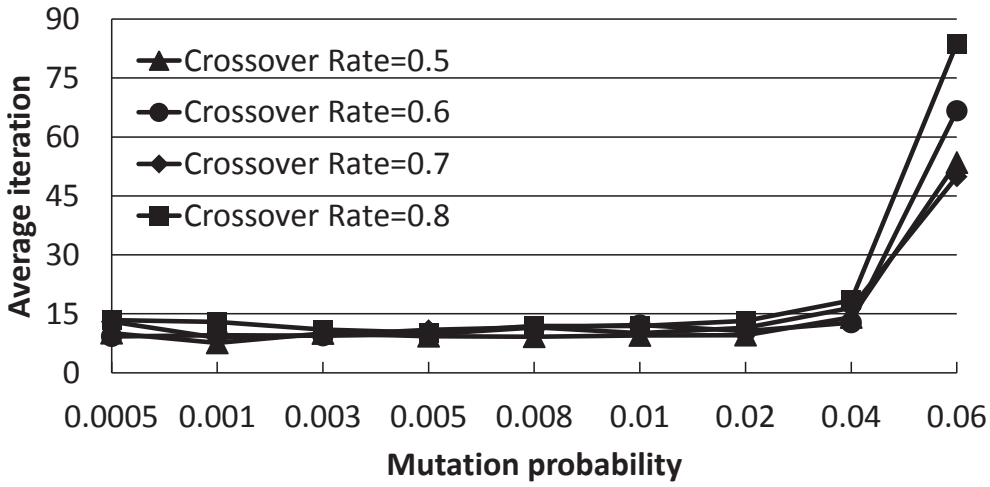


Figure 5.12: Average iterations based on crossover rate and mutation rate

Chapter 6

Study the Effects of Similarity Measures, the Length of Time Series Data, and the Environmental Factors on the Performance of Pattern Recognition [1] [2]

This chapter examines several time series representation methods and similarity measures for structural damage feature extraction and pattern recognition. The study result is a good reference for selecting effective feature extraction and similarity measure methods.

[1] A portion of this chapter has been submitted to Mechanical Systems and Signal Processing, Elsevier

[2] ©DEStech. Portions reprinted with permission, from W. Liu and B. Chen, "Bio-inspired Computing Algorithms for Adaptive Structural Health Monitoring," Proceedings of the Eighth International Workshop on Structural Health Monitoring, 2011. Lancaster, PA: DEStech Publications, Inc.

Both model-based and dimensionality reduction representation methods are investigated. The computation of the presented time series representation algorithms can be handled by the sensor node level software. Pattern-recognition-based structural damage detection and classification are based on the similarity measure of damage feature vectors with normal feature vectors. The goal of the feature extraction is to select features which will separate damage feature vectors and normal feature vectors in the feature space. This separation will allow us to distinguish damage and normal patterns. The performance of feature extraction methods and similarity measures are evaluated utilizing acceleration data collected from the Z24 Bridge as part of the System Identification to Monitor Civil Engineering Structures (SIMCES) project [2].

6.1 Z24 Bridge Test Data

The Z24 Bridge, located in Switzerland, was monitored for almost one year from November 1997 to September 1998 as the SIMCES project. The data collected from the Z24 Bridge consists of two parts: environmental monitoring system (EMS) measurements and progressive damage test (PDT). During EMS test, both environmental and vibration data are collected. The environmental parameters such as wind characteristic, humidity, air temperature, soil temperature, and temperatures of multiple spots on the bridge were monitored. In the PDT test, acceleration data were recorded from over 150 accelerometers under normal and various damage scenarios as shown in Figure 6.1. Table 6.1 provides

Table 6.1
Description of progressive damage tests

Test	Description	Test	Description
Pattern 1	No Damage (missing/corrupted data)	Pattern 9	Concrete Spalling: 12m2
Pattern 2	No Damage, Pier Hinge Added (Baseline)	Pattern 10	Concrete Spalling: 24m2
Pattern 3	Pier 3 Settlement: 20mm	Pattern 11	Landslide at Abutment
Pattern 4	Pier 3 Settlement: 40mm	Pattern 12	Concrete Hinge Failure
Pattern 5	Pier 3 Settlement: 80mm	Pattern 13	Anchor Head Failure (2)
Pattern 6	Pier 3 Settlement: 95mm	Pattern 14	Anchor Head Failure (4)
Pattern 7	Pier 3 Foundation Tilt	Pattern 15	Tendon Wire Failure (54/2)
Pattern 8	No damage, Pier 3 Restored	Pattern 16	Tendon Wire Failure (100/4)

a list of damage scenarios including same damage type (pier settelment) with different damage severity and different damage modalities. Data from this project has been used in a number of published studies [96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106].

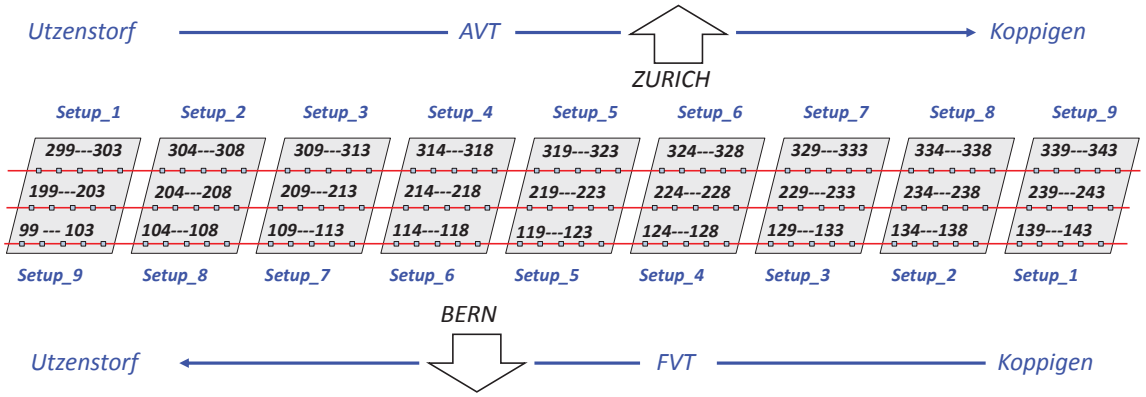


Figure 6.1: Measurement set-up for vibration test on Z24 bridge

In the presented study, the data collected from these damage scenarios are divided into training and test subsets. The training subset was used to generate representative feature

vectors for damage patterns, and the test subset was used to find the success rate of the pattern recognition.

6.2 Effects of Feature Extraction Methods, Similarity Measures, and the Length of Time Series Data on the Performance of Pattern Recognition

Performance evaluation was conducted to test the effectiveness of the feature extraction methods and the impact of similarity measures and the length of time series. Two test scenarios were designed: (1) same type of structural damage with different extents (Pattern 2 - Pattern 6 in Table 6.1) and (2) different damage modalities (Patterns 6, 10, 11, 12, 14, and 16 in Table 6.1). I adopted sensor data collected by sensor node 232 in the forced vibration test. Data points in the sensor data files were divided into two groups: training data and test data. Features vectors generated from the training data were use to find the representative feature vectors for each damage pattern using K-means method. The feature vectors created from test data were used to test the effectiveness of feature extraction methods for damage pattern recognition using K-nearest neighbor (KNN-1) classification method. The success rate in this context refers to the ratio of the number of test feature vectors which are correctly identified to the number of total test feature vectors.

To find good similarity measures for structural damage pattern recognition, a number of commonly used similarity measures are evaluated using Z24 bridge data sets. These similarity measures are able to represent the distance between feature vectors in the feature space and help classify the feature vectors. The tested similarity measures include Euclidean distance, L-infinity (Maximum) norm, Mahalanobis distance, Cosine distance, Standardized Euclidean (Seuclidean) distance, and Correlation distance. Let X and Y are two feature vectors with dimension n: $X = (x_1, x_2, \cdots\cdots, x_n)^T$ and $Y = (y_1, y_2, \cdots\cdots, y_n)^T$. The definitions of these similarity measures are given below:

• *Manhattan distance* :

$$d_{XY} = \sum_{i=1}^{n} |x_i - y_i| \tag{6.1}$$

• *Euclidean distance* :

$$D = \sqrt{(X-Y)(X-Y)^T} \tag{6.2}$$

• $L-infinity$:

$$d_{XY} = max(|X_i - Y_i|), i \in n \tag{6.3}$$

• The Mahalanobis distance of a multivariate vector $X = (x_1, x_2, \cdots\cdots, x_n)^T$ from a group

of values with mean $\mu = (\mu_1, \mu_2, \cdots\cdots, \mu_n)^T$ and covariance matrix S is defined as:

$$D = \sqrt{(X-\mu)^T S^{-1}(X-\mu)} \tag{6.4}$$

• *Cosine distance* :

$$d_{XY} = 1 - \frac{XY^T}{(XX^T)^{\frac{1}{2}}(YY^T)^{\frac{1}{2}}} \tag{6.5}$$

• *Standardized Euclidean*$(Seuclidean)$ *distance* :

$$d^2_{XY} = (X-Y)D^{-1}(X-Y)^T \tag{6.6}$$

where D is a diagonal matrix with diagonal elements given by v_j^2, which denotes the variance of the jth-feature over all the features vectors contained by X and Y.

• *Correlation distance* :

$$d_{XY} = 1 - \frac{(X-\bar{X})(Y-\bar{Y})^T}{((X-\bar{X})(X-\bar{X})^T)^{\frac{1}{2}}((Y-\bar{Y})(Y-\bar{Y})^T)^{\frac{1}{2}}} \tag{6.7}$$

where $\bar{X} = \frac{1}{p}\sum_j X_j, \bar{Y} = \frac{1}{p}\sum_j Y_j$

6.2.1 Effects of Similarity Measures and the Length of Time Series Data Using AR-based Feature Extraction

To test the performance of feature extraction methods, the Z24 Bridge data sets described in Section 6.1 are used. In the Z24 bridge data sets, each sensor data file contains 65536 acceleration data points. For the feature extraction, the first 4999 data points are abandoned to avoid using unstable sensor data which may occur in the beginning of each test. The rest of sensor data in the data file are used to form data series. The length of the time series data are selected as: 100, 200, 300, 500, 700, 1000, 1500, 2000, 3000, and 5000.

The success rate of classifying test data to corresponding damage patterns using AR-based feature extraction method was evaluated for different damage modalities and progressive damage patterns. Figure 6.2 shows the average success rate of pattern recognition in first scenario (Pattern 2 - Pattern 6 in Table 6.1) using similarity measures defined above. Five data patterns defined in the first scenario are No Damage, Pier 3 Settlement-20mm, Pier 3 Settlement-40mm, Pier 3 Settlement-80mm, and Pier 3 Settlement-95mm. The x axis stands for the length of time series for feature extraction; the y axis stands for the type of similarity measures; and the z axis is the average success rate of AR-based feature extraction method. From Figure 6.2 we can see that the Mahalanobis distance outperforms over other similarity measures. For each similarity measure, the success rate increases as the length of time series gets longer. Figure 6.3 shows the average success rate of pattern

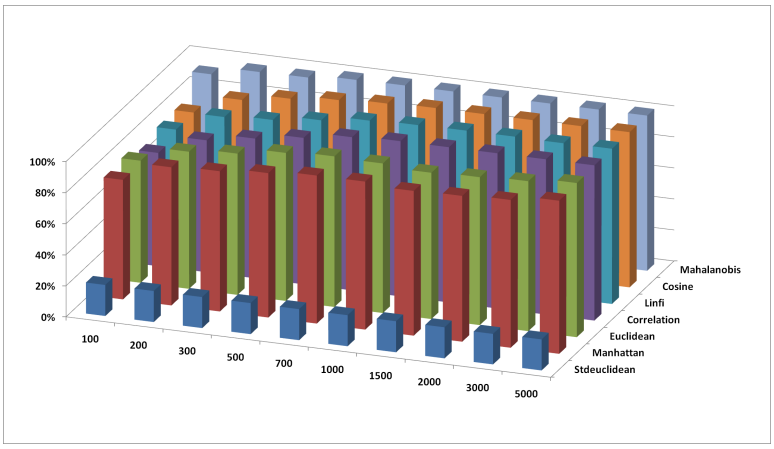


Figure 6.2: Average success rate of pattern recognition with different similarity measures and the length of time series (first scenario).

recognition for Patterns 6, 10, 11, 12, 14, and 16 in Table 6.1 (pier 3 settlement-95 mm, concrete spalling-24 m^2, landslide at abutment, concrete hinge failure, anchor head failure, and tendon wire failure). Figure 6.3 presents similar trends as Figure 6.2 with regard to the effects of similarity measures on the pattern recognition success rate for AR-based feature extraction method. The success rate in Figure 6.3, however, is generally lower than that in Figure 6.2. This is due to the separation of feature vectors in first scenario is better than that of the second scenario. This can be observed from Figure 6.4 and Figure 6.5.

The impact of the length of time series on pattern recognition success rate for individual pattern using Mahalanobis distance as similarity measure was also investigated. Figure 6.6 shows the success rates of pattern recognition for the patterns in the first scenario and Figure 6.7 shows the success rates of pattern recognition for the patterns in second scenario. Figure 6.6 indicates that the success rate increases as the length of time series increases. In addition, the severity of damage affects the pattern recognition success rate also. Pattern 6,

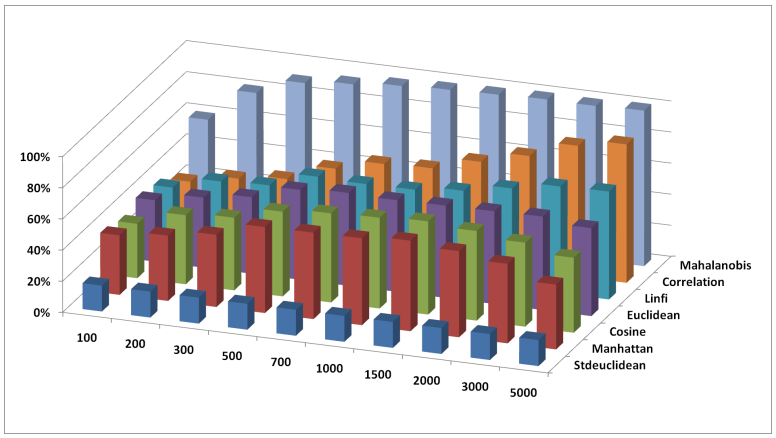


Figure 6.3: Average success rate of pattern recognition with different similarity measures and the length of time series (second scenario).

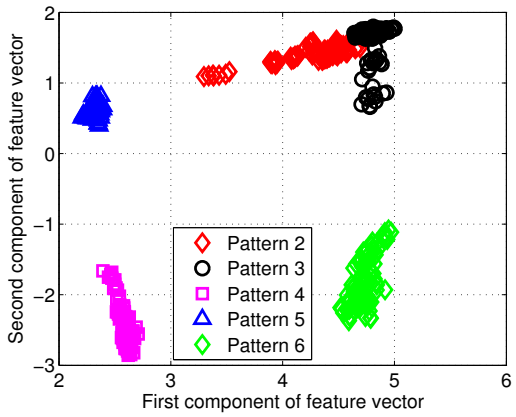


Figure 6.4: The feature vectors of data patterns in first scenario from sensor node 232 using Mahalanobis distance

with largest settlement, has the highest pattern recognition success rate. Figure 6.7 shows the success rate of pattern recognition performed on different damage modalities. Similarly, the success rates go up as the lengths of time series increases.

The success rate is also affected by the separation of feature vectors in the feature space. Figure 6.4 and Figure 6.5 show the distribution of feature vectors in the first and second scenarios. The length of time series is 5000 in both plots. To display high dimensional

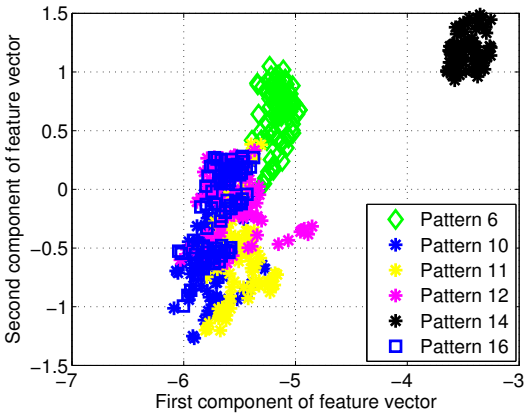


Figure 6.5: The feature vectors of data patterns in second scenario from sensor node 232 using Mahalanobis distance

feature vectors in 3D space, 20-dimensional feature vectors are reduced to 3-dimentsional feature vectors using principal component analysis. In Figure 6.4, the feature vectors of pattern 6 are located far away from feature vectors of other patterns. As a result, pattern 6 is easy to be recognized. The success rate of pattern 6 is the highest one comparing with other patterns. Figure 6.5 shows the distribution of the feature vectors from different damage modalities. As we can be seen from Figure 6.5, feature vectors of pattern 14 is located far away from feature vectors of other patterns, so the success rates of pattern 14 is higher than the success rate of other patterns. In general, the separation of feature vectors in first scenario is better than that of the second scenario. The overall success rate in first scenario is also higher than that of the second scenario.

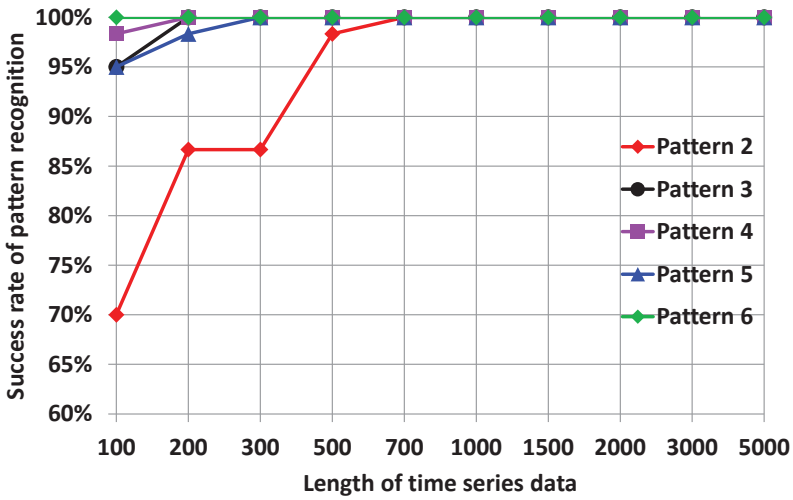


Figure 6.6: Success rate of pattern recognition for the patterns in the first scenario (pattern 2-6) using Mahalanobis distance

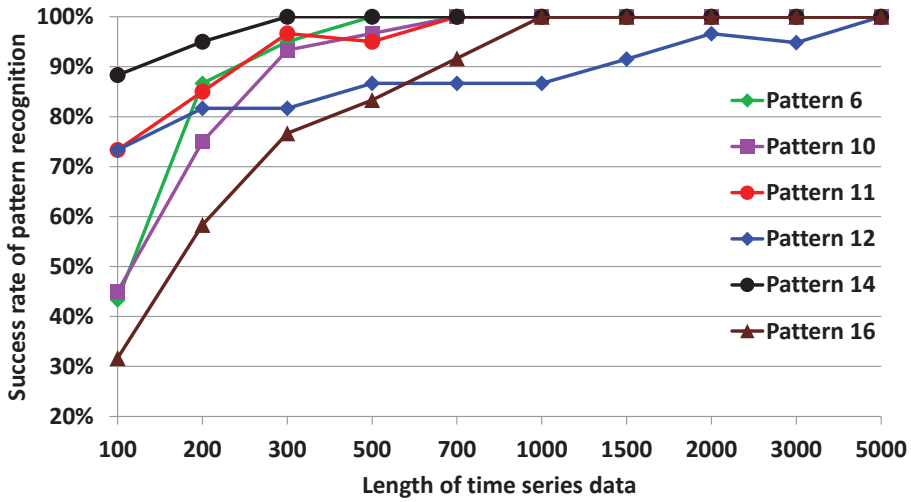


Figure 6.7: Success rate of pattern recognition for for the patterns in the second scenario (patterns 6, 10, 11, 12, 14, and 16) using Mahalanobis distance

6.2.2 Effects of Similarity Measures and the Length of Time Series Data Using DFT-based Feature Extraction

Figure 6.8 and Figure 6.9 show the average success rate of DFT-based feature extraction method for structural damage pattern recognition with different similarity measures. In general, the success rate of DFT-based feature extraction is lower than that of AR-based feature extraction method. Compare with two test scenarios, the first scenario has relatively high success rate. In both test scenarios, the dissimilarity measure - Mahalanobis distance again showing better performance than other similarity measures. Figure 6.10 and Figure

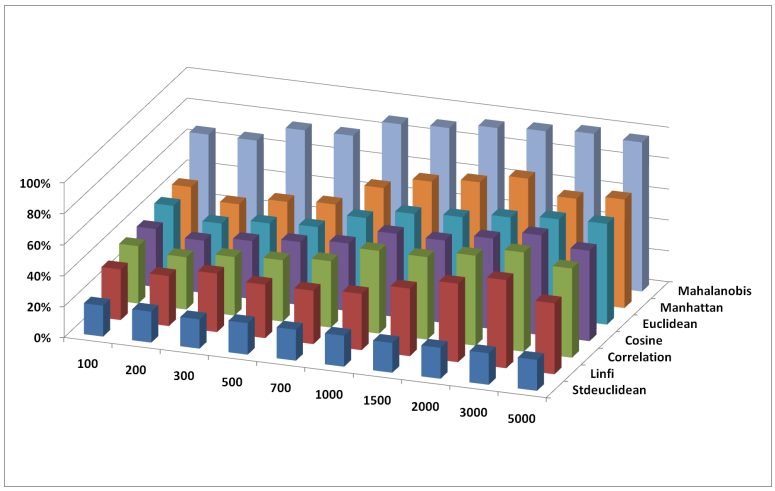


Figure 6.8: Average success rate of pattern recognition using DFT-based feature extraction in first scenario

6.11 show the success rates of pattern recognition for each damage pattern with different lengths of time series. The similarity measure used in the tests is the Mahalanobis distance. For most damage patterns, the success rate increases as the length of time series increases.

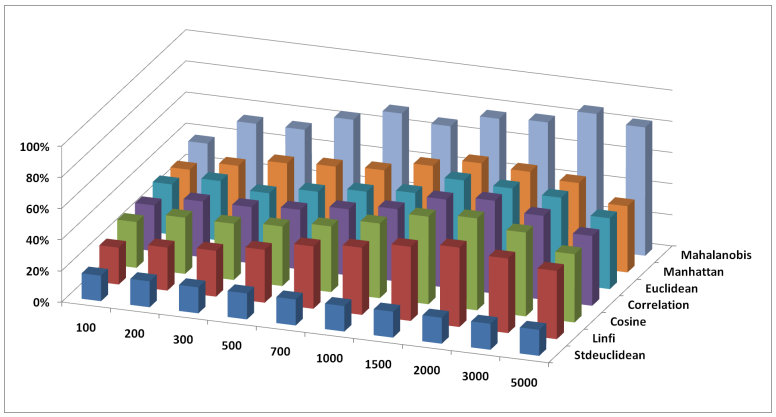


Figure 6.9: Average success rate of pattern recognition using DFT-based feature extraction in second scenario

In general, the success rate of pattern recognition in first scenario is better than that of the second scenario. Figure 6.12 and Figure 6.13 show the distribution of the feature vectors in two scenarios using DFT-based feature extraction method. The length of time series is 5000 in both plots. The separation of feature vectors using AR-based feature extraction method (Figure 6.4 and Figure 6.5) is better than that of the DFT-based feature extraction method (Figure 6.12 and Figure 6.13). As a result, the success rates of pattern recognition using AR-based feature extraction (Figure 6.6 and Figure 6.7) are higher than that of the DFT-based feature extraction (Figure 6.10 and Figure 6.11).

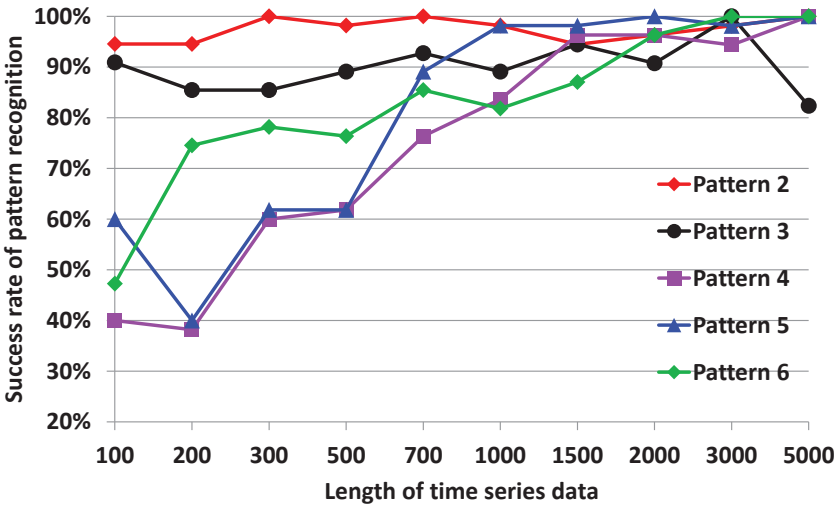


Figure 6.10: Success rate of pattern recognition using DFT-based feature extraction for the patterns in the first scenario

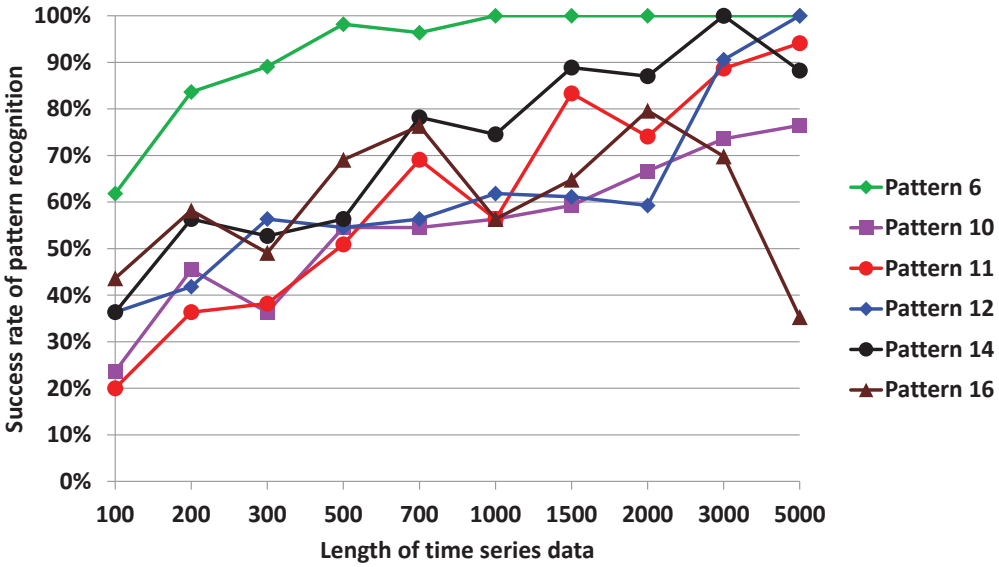


Figure 6.11: Success rate of pattern recognition using DFT-based feature extraction for the patterns in the second scenario

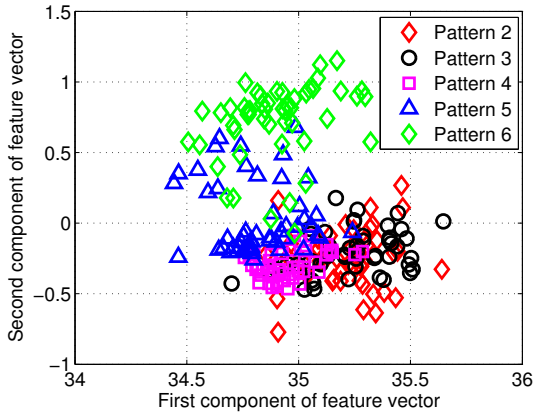


Figure 6.12: The feature vectors of data patterns in first scenario from sensor node 232

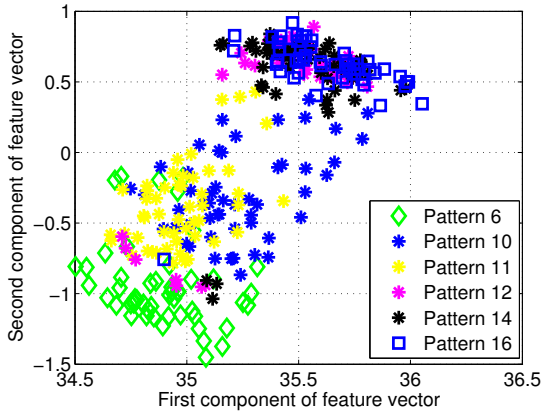


Figure 6.13: The feature vectors of data patterns in second scenario from sensor node 232

6.2.3 Effects of Similarity Measures and the Length of Time Series Data Using DWT-based Feature Extraction

Figure 6.14 and Figure 6.15 show the average success rate of DWT-based feature extraction method for structural damage pattern recognition with different similarity measures. In

general, the success rate of DWT-based feature extraction is lower than that of AR-based feature extraction method but higher than that of the DFT-based feature extraction method. Compare with two test scenarios, the first scenario has relatively high success rate. In both test scenarios, the dissimilarity measure - Mahalanobis distance again showing better performance than other similarity measures. Figure 6.16 and Figure 6.17 show the success

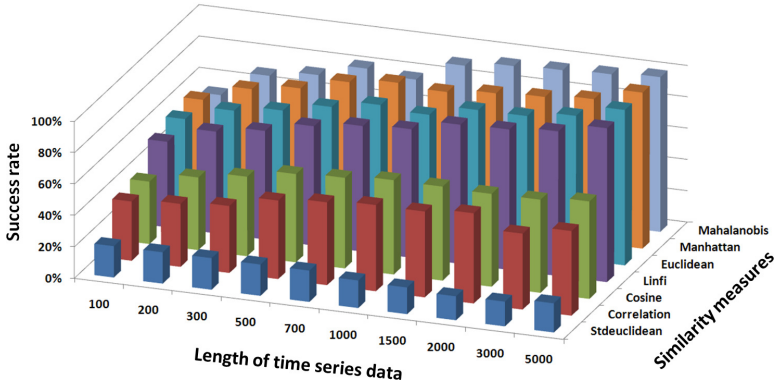


Figure 6.14: Average success rate of pattern recognition using DWT-based feature extraction in first scenario

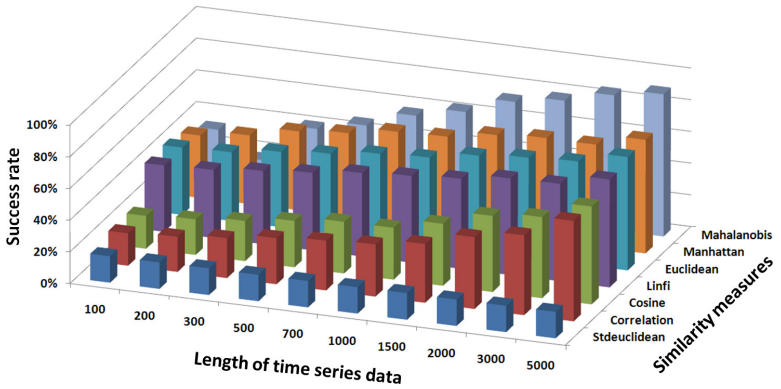


Figure 6.15: Average success rate of pattern recognition using DWT-based feature extraction in second scenario

rates of pattern recognition for each damage pattern with different lengths of time series. The similarity measure used in the tests is the Mahalanobis distance. For most damage

patterns, the success rate increases as the length of time series increases. Figure 6.18 and Figure 6.19 show the distribution of the feature vectors in two scenarios. The length of time series is 5000 in both plots. Comparing the success rate plot with feature distribution in both test scenarios, the impact of the separation of feature vectors on the success rate can clearly be seen again. The success rates of pattern 2 and 3 in the first scenario are much higher comparing with other patterns, and the success rates of pattern 6 and 14 in the second scenario are much higher comparing with other patterns.

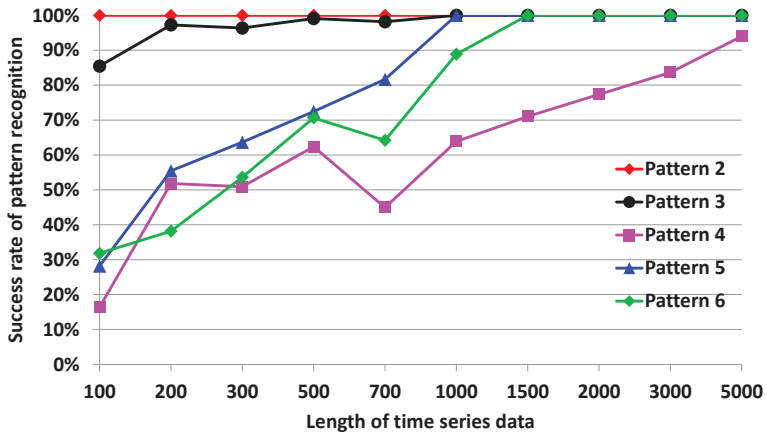


Figure 6.16: Success rate of pattern recognition using DWT-based feature extraction for the patterns in the first scenario (sensor 232).

To investigate the impact of the time series length on computing time, simulation tests were performed using different feature extraction methods and with various time series lengths. The evaluation tests were conducted using a Dell computer with Intel Core2 Quad 2.4GHz CPU and 4 GB of RAM. Figure 6.20 shows the computing time for three feature extraction methods when the length of the time series changes. For the DFT- and DWT-based feature extraction methods, the length of the time series does not have significant impact on the

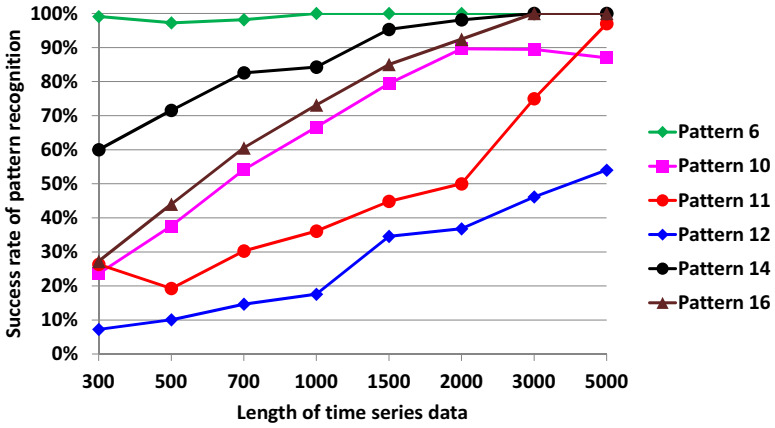


Figure 6.17: Success rate of pattern recognition using DWT-based feature extraction for the patterns in the second scenario (sensor 232).

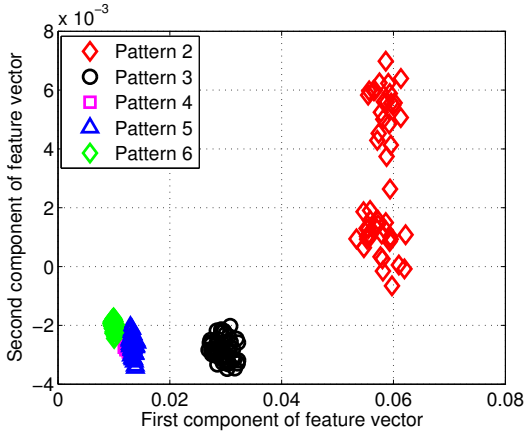


Figure 6.18: The feature vectors of data patterns in first scenario from sensor node 232

computing time. For the AR-based feature extraction method, the computing time increases when the length of the time series increases.

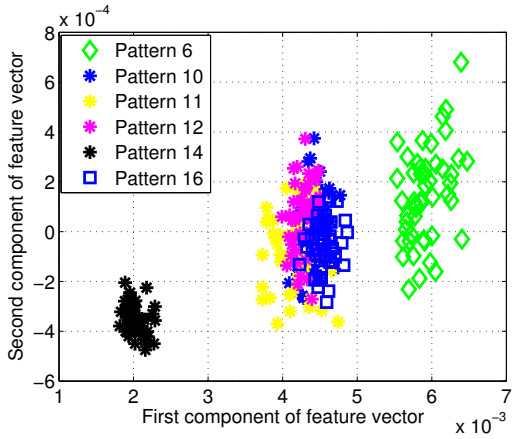


Figure 6.19: The feature vectors of data patterns in second scenario from sensor node 232

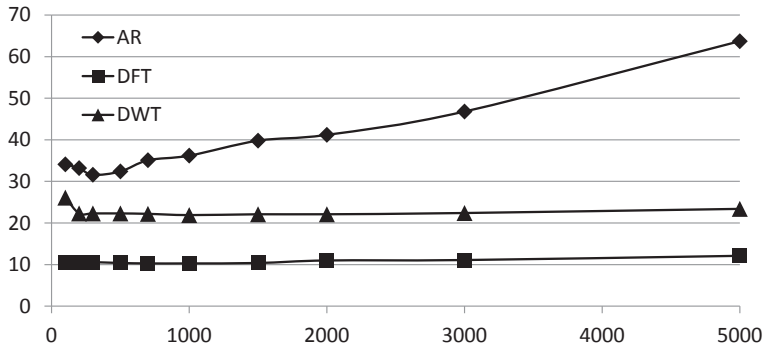


Figure 6.20: The impact of the time series length on the computing time for different feature extraction methods

6.3 Structural Damage Localization Using Pattern Recognition Approach

Damage localization is important when a damage is detected. To investigate the applicability of pattern recognition approach for structural damage localization, the numerical analysis study has been conducted to examine the shift of the representative

feature vectors of the damage patterns from a normal pattern in the feature space using damage pattern 6 data of Z24 Bridge. To find out the potential relationship between damage location and the feature vectors of sensor data, the distances between normal pattern feature vectors and damage pattern 6 feature vectors are calculated. The similarity measure used in the calculation is Mahalanobis distance as shown below:

$$FeatureShiftDistance = Mahalanobis(damagefeaturevectors, normalfeaturevectors) \tag{6.8}$$

Figure 6.21 shows the feature vector shift of damage pattern 6 from the normal pattern on the sensor nodes 120-335. The distribution of these sensor nodes on the bridge is indicated by numbers corresponding to their IDs as shown in Figure 6.21. There are three rows of sensor nodes. The sensor nodes 120-135 form the first row and are located in the front edge of the bridge; the sensor nodes 220-235 form the second row and are located in the middle of the bridge; the sensor nodes 320-335 form the third row of sensor arrays. The sensor data used for the numerical analysis are chosen from the forced vibration tests with vertical directionality. The length of the sensor data time series is 5000. The shifted distances are measured by the centroids of the normal feature vectors and the damage pattern 6 feature vectors. From Figure 6.21, we can see that the closer the sensor nodes to the damage location (pier 3), the larger the shifted distance from pattern 6 feature vectors to the normal feature vectors. This result shows the potential of using pattern recognition approach for damage localization analysis.

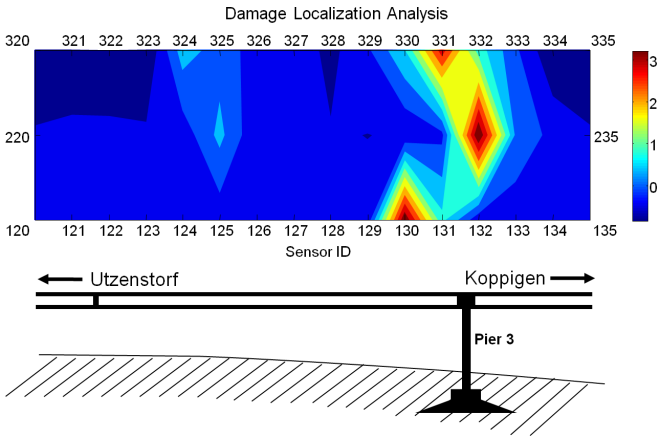


Figure 6.21: The shifted distances of damage pattern 6 feature vectors from normal pattern feature vectors

6.4 Effects of Environmental Factors on the Performance of Pattern Recognition

To study the impact of the temperature on the normal feature vectors and the updating of the normal pattern model, the environmental monitoring data collected from Z24 Bridge as part of the SIMCES (System Identification to Monitor Civil Engineering Structures) project [107] are used. The AIPR-based damage pattern recognition is achieved by establishing representative feature vectors for each damage pattern in a training process. The training process of the AIPR algorithm is based on the clonal selection principle of the natural immune system. The initial representative feature vectors are generated by the random selection of the feature vectors from the training data. The rest of training feature vectors are used to stimulate the evolution of representative feature vectors using clonal selection principle.

To find normal pattern model, discrete Fourier transform method is used to extract features from environmental data. Each feature vector contains 16 feature members which are the first 8 model frequencies and the corresponding amplitudes. To study the impact of the temperature on the normal pattern model, the representative feature vectors of the normal pattern at different temperatures are generated using the AIPR algorithm. Figure 6.22 shows the representative feature vectors of the normal pattern at $8°C, 9°C, 27°C, 29°C$, and the representative feature vectors of the damage pattern 3 at $29°C$. These representative feature vectors are generated using the acceleration data collected by the sensor node 7 in the Z24 bridge environmental monitoring data set. Each acceleration data file in the EMS test contains 65536 number of data points. To avoid using the unstable data, the first 5000 number of data points are dropped. Starting from the 5001th data point, each 10000 of data points are used to extract one feature vector using discrete Fourier transform method. The second feature vector is calculated by advancing 1500 data points, in other words, starting from 6501th data point. The temperature sensor used in the calculation is TDT3 (temperature in deck top) due to its relatively stable performance discussed in the [2].

Figure 6.22 shows that the normal pattern model moves in the feature space when the temperature changes. To improve the performance of the damage pattern recognition algorithm, the updating of the normal pattern model to the changes of the temperature is necessary. Figure 6.23 compares the success rates of detecting damage 3 using two normal pattern models, one at $9°C$ and the other at $29°C$. The feature vectors of the damage pattern 3 are generated from the sensor data at $29°C$. In Figure 6.23, the k-nearest neighbor (kNN)

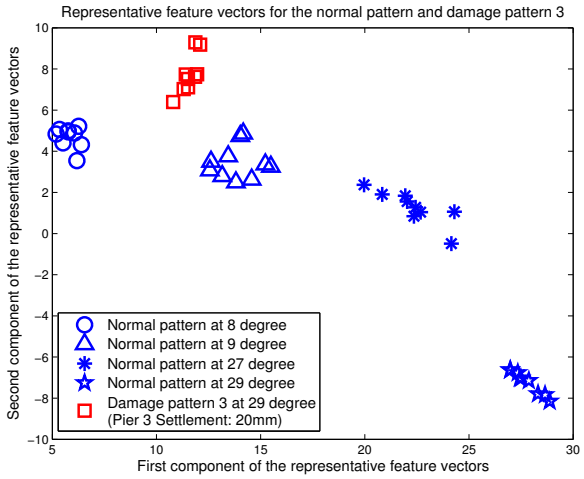


Figure 6.22: The impact of the temperature on the representative feature vectors of the normal pattern model

algorithm is used for the damage detection. The success rates are compared with several distance measures. Euclidean distance, Manhattan distance, L-infinity (Maximum) norm, Cosine distance, and Mahalanobis distance are employed for the similarity measure. The comparison result in Figure 6.23 shows that the success rates of detecting damage pattern 3 are much lower if the $9°C$ normal pattern model is used. This is due to the fact that the $9°C$ normal pattern model is very close to the damage pattern 3 at $29°C$ as shown in Figure 6.22.

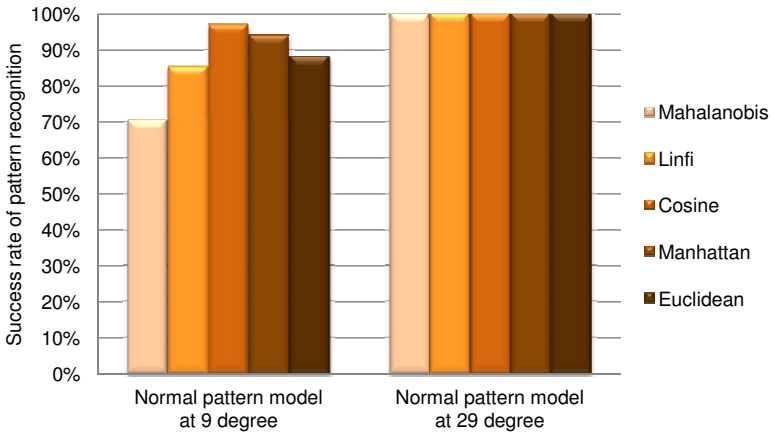


Figure 6.23: The success rates of the damage pattern recognition with or without the updating of the normal pattern model

Chapter 7

Applications of Mobile Agents, High Computational Power Monitoring Networks and Pattern Recognition [1] [2] [3] [4]

[1] ©Wiley. Portions reprinted with permission, from B. Chen and W. Liu, "Mobile Agent Computing Paradigm for Building a Flexible Structural Health Monitoring Sensor Network," Computer-Aided Civil and Infrastructure Engineering, vol. 25, no. 7, pp. 504-516, 2010.

[2] ©Inderscience. Portions reprinted with permission, from B. Chen and W. Liu, "A web-based structural health monitoring sensor network," International Journal of Computer Applications in Technology, Vol. 44, 2012.

[3] ©Inderscience. Portions reprinted with permission, from B. Chen and W. Liu, "A High Computational Power Wireless Sensor Network for Distributed Structural Health Monitoring," International Journal of Sensor Networks, vol. 11, 2012.

[4] ©SAE. Portions reprinted with permission,SAE Paper No. 2012-01-0742 ©2012 SAE International.

7.1 Mobile-Agent-based Structural Health Monitoring

Structural health monitoring (SHM) is an emerging technology in civil, mechanical, and aerospace engineering to detect damage in structures [108, 109, 110, 111, 112]. The SHM process is the combination of signal collection from the structure via sensors, feature extraction from the measurements and pattern recognition of the features to check the status of the monitored structure [113].

Generally, the sensor deployed close to the damage location is expected to collect data which are more affected than the data collected from sensors deployed far from the damage location [114]. Thus a sensor nextwork which is deployed on a complicated structure is able to supply the damage degree and location information if each sensor node has powerful computation and communication capabilities.

Although sensor network approach is suitable for SHM, the design of wireless sensor networks presents a number of challenges. (1) Adaptability: Sensor networks suffer substantial network dynamics due to node failure, added new nodes, environmental obstructions, and user demand changes. A sensor network should be able to make appropriate adjustments to operate robustly when the environment and network itself change [115]. (2) Distributed data processing and damage diagnosis: Due to the high sampling frequency, a SHM sensor network generates a huge amount of measurement data

during the monitoring process. If all the sensor data are centrally processed, these data need to be sent to a central station. Transmitting this large amount of data over a wireless sensor network is challenging because of the significant limitation of communication bandwidth. To reduce the raw data transmission and the response time, a number of researchers have proposed distributed data processing in SHM sensor networks [116]. (3) Scalability: Scalability is the ability of a sensor network to allow the growth of the number of sensor nodes without affecting the performance of the network [81]. Scalability is a desirable property of a sensor network since the size of the required network is usually unknown at the design stage. The sensor network should maintain at an acceptable performance level as the network grows for a larger sensing area or higher resolution. (4) Self-organization: For large structures, sensor networks usually consist of thousands of nodes and may be deployed in unreachable environments (embedded in physical structure). Having such a deployment size and environment, it is impossible to pay special attention to any individual node. Self-organization is a key issue in the design of sensor networks [117]. (5) Multi-tasking: Most existing sensor networks were designed to be application specific. However, it is widely accepted that sensor networks will have a long-deployment cycles serving multiple transient users with dynamic needs [118]. In addition, multiple applications (tasks) may be performed concurrently over a single sensor network.

To address aforementioned challenges, a framework of mobile agent-based sensor network that pursues desirable characteristics, such as adaptability, distributed damage diagnosis, and sensor node collaboration is developed. The major design considerations of the

presented sensor network framework are as follows. (1) Sensor node design: possess high computational power; equip with multi-modality sensors; open source Linux operating system (OS); and open source software implementation; (2) Network middleware design: reduce network traffic by moving computational algorithms instead of sensor data; support generation and migration of mobile monitoring agents; allow collaboration in local sensor communities and collaborative distributed data processing; self-organize through mutual interaction among agents to agents and agents to the environment. The hardware and software design of the sensor node has been introduced in Chapter 4.

7.1.1 Experimental Setup

A scaled steel bridge shown in Figure 7.1 was used for the mobile agent validation test. The bridge has two side beams and eight cross members. Each side beam is composed of six beam sections. Cross members are distributed near the connections of side beams with two members are crossed at the center of the bridge. Accelerometers were mounted on the top of side beams as shown in Figure 7.2. The outputs of accelerometers were connected to analog/digital (A/D) converters on the sensor board nearby.

During test, the bridge was excited by a shaker at the center of the bridge as shown in Figure 7.1. Figure 7.3 shows the excitation and force sensing loop. The excitation signals of the shaker were generated by Siglab and virtual instruments. The siglab can read in and

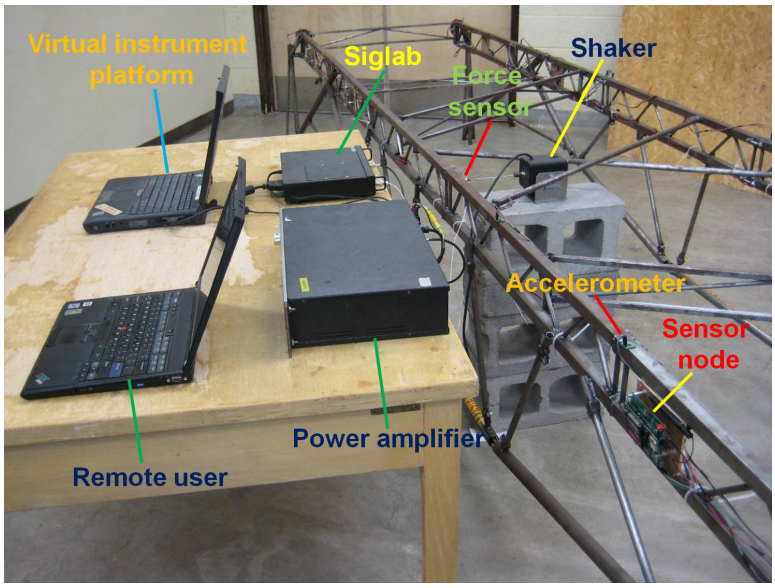


Figure 7.1: Test bridge structure

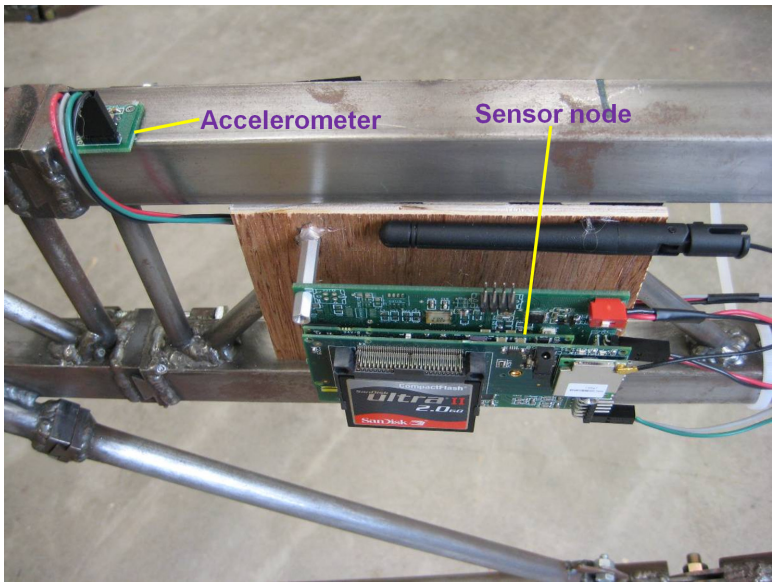


Figure 7.2: Sensor node and accelerometer

send out signals via the virtual instruments which is based on the MATLAB platform. The virtural instruments include Function Generator, Oscilloscope, Spectrum Analyzer and Network Analyzer. In the bridge test, the virtual Function Generator is adopted to

generate shaker excitation signals and the virtual Network Analyzer is used to check the signals collected by a force sensor which was attached to the end of the shaker. The shaker excitation signals generated by the Function Generator were amplified by a power amplifier. Both the shaker and power amplifier are made by the labworks company. The output of the force sensor was fed back to the Siglab and displayed in the Graphical User Interface (GUI) of the virtual instruments on the Laptop.

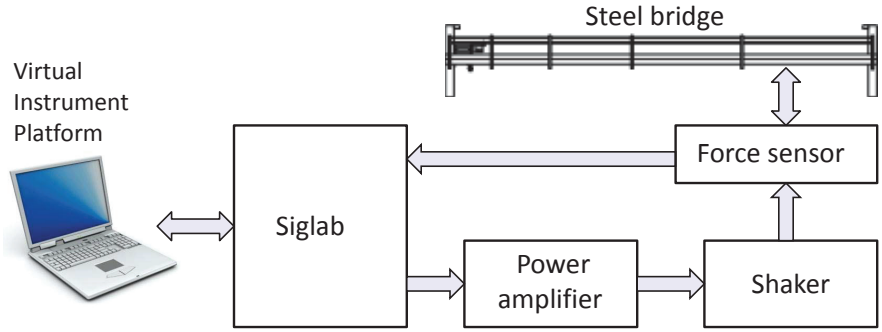


Figure 7.3: Shaker and excitation signal generation

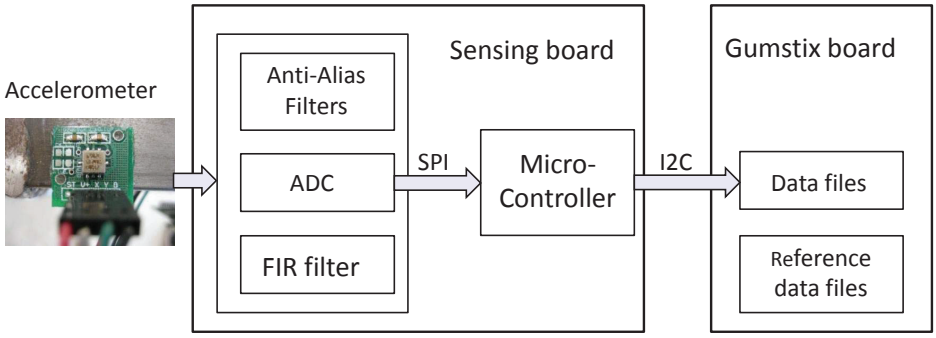


Figure 7.4: Acceleration signal conditioning and data transmission between the sensing board and the Gumstix board

Figure 7.4 shows the acceleration data collection, signal conditioning, and data transmission between the sensing board and the Gumstix board. Acceleration data were

sampled at a rate of 125sps. Microcontrollers on the sensing boards read acceleration data from A/D converters through an SPI interface. The collected acceleration data were transmitted to the Gumstix board and saved into data files on the Gumstix board. The inter-board communication between the Gumstix board and the sensing board is achieved by I^2C serial communication. To validate the dynamic deployment of mobile agents for damage diagnosis, sensor nodes 1 and 2 on the scaled steel bridge were chosen for the validation test. Figure 7.5 shows the size of the steel bridge and locations of sensor nodes.

Structural damage was simulated by removing two cross members at the center of the bridge. For both of normal and damage patterns, acceleration data collected by sensor nodes 1 and 2 were recorded. The excitation signals for the shaker were sinusoid waves with peak-to-peak voltage of 0.255 volts, 0.275 volts, and 0.295 volts, respectively.

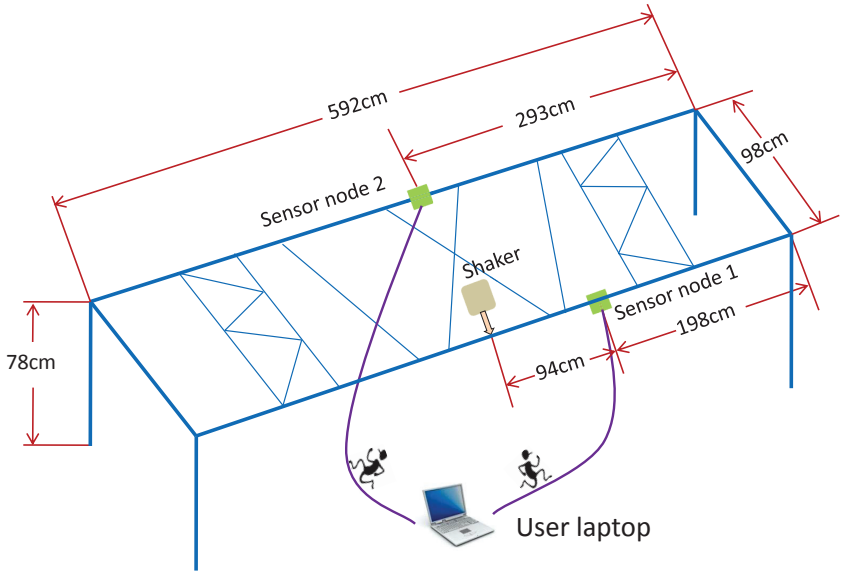


Figure 7.5: Sensor node locations (not to scale)

7.1.2 Auto-Regressive and AR with Exogenous Input (ARX) Structural Damage Diagnosis Algorithm

The damage diagnostic algorithm selected is Auto-Regressive and AR with Exogenous input (ARX) models proposed by Sohn and Farrar [4]. This two-stage prediction method firstly uses an AR model as shown in Equation 7.1 to fit a discrete time series of acceleration data $x(k)$. The structural response data at time $t = k\Delta t$, $x(k)$, is a function of p previous response data plus the error term $e_x(k)$. Weights on previous response data are AR coefficients. Since the error $e_x(k)$ in Equation 7.1 is also affected by unknown external inputs, an ARX model is used in the second stage to establish the relationship between time signal $x(k)$ and AR model error $e_x(k)$, as shown in Equation 7.2. The term $\varepsilon_x(k)$ is the residual error of the ARX model.

$$x(k) = \sum_{i=1}^{p} c_{xi} x(k-i) + e_x(k) \tag{7.1}$$

$$x(k) = \sum_{i=1}^{a} \alpha_i x(k-i) + \sum_{j=0}^{b} \beta_j e_x(k-j) + \varepsilon_x(k) \tag{7.2}$$

To use AR-ARX method for damage diagnosis, a reference file that contains AR and ARX prediction model pairs is required. These AR and ARX predication models are constructed

based on discrete time data sets representing the undamaged structure. During damage diagnosis, AR coefficients are computed with Equation 7.3 using measured discrete time acceleration data $y(k)$ from the monitoring structure. Next, the identification of an ARX model in the reference file is conducted by matching the measured AR model with an AR model in the reference file based on the minimum distance measure shown in Equation 7.4. The counterpart (ARX model) of the matched AR model in the reference file is used to calculate the residual errors of measured data set using Equation 7.5. The ratio $\frac{\sigma(\varepsilon_y)}{\sigma(\varepsilon_x)}$ is defined as a damage sensitive feature. An appropriate threshold of this ratio is chosen to minimize false-positive and false-negative damage identification.

$$y(k) = \sum_{i=1}^{p} c_{yi} y(k-i) + e_y(k) \tag{7.3}$$

$$Distance = \sum_{i=1}^{p} (c_{xi} - c_{yi})^2 \tag{7.4}$$

$$\varepsilon_y(k) = y(k) - \sum_{i=1}^{a} \alpha_i y(k-i) - \sum_{j=0}^{b} \beta_j e_y(k-j) \tag{7.5}$$

At each excitation voltage level mentioned in the test setup, a total of 26 acceleration time series on each sensor node were recorded for the normal pattern, while 19 time series

Table 7.1
Off-line calculation of the ratio of standard deviation of the residual time series

Sensor	Pattern	Ratio $\frac{\sigma(\varepsilon_y)}{\sigma(\varepsilon_x)}$
1	Normal	1.5240 2.0204 1.6099 1.6241 2.4093 1.8547 1.5547 1.8743 2.0055 1.2815 1.5093 1.7076 1.7283 1.1498 1.8325 1.9087
	Damage	3.8485 3.8225 4.2859 4.0668 3.8307 3.8489 4.1123 3.8565 3.8528 3.8592 4.1418 4.1415 3.8429 4.1251 4.1180 4.1094 4.1293 4.1176 3.8475
2	Normal	1.5054 1.0246 1.5293 1.4173 1.7241 1.6026 1.9286 1.6165 1.5583 1.1919 1.3007 1.4971 1.1587 1.2641 1.4014 1.3650
	Damage	7.5001 14.4038 7.6681 14.3588 7.7951 7.5751 7.2733 6.9559 12.7211 7.0272 5.9883 5.9110 7.2269 12.5588 6.3172 10.9039 6.2257 6.1708 6.0201

were recorded for the damage pattern. The AR-ARX reference file was constructed by using 10 normal acceleration time series. The standard deviation of the residual time series of rest 16 time series for the normal pattern acceleration data and 19 time serials for the damage pattern were calculated off-line for the comparison of mobile agent results in the next section. The values of the ratio, $r = \frac{\sigma(\varepsilon_y)}{\sigma(\varepsilon_x)}$, are listed in Table 7.1 when the excitation signal voltage is at 0.275 volts.

7.1.3 Mobile Agents for Distributed Structural Damage Diagnosis

For the mobile agent test, mobile agent 1 and 2 were sent to sensor node 1 and 2 respectively. The mobile agents are created to diagnose if there were damage happened by performing AR-ARX method on the collected acceleration data. In this example, the flowchart of the mobile agent code is shown in Figure 7.6. A mobile agent fits the acceleration data to the AR model on the resided sensor node. Based on the calculated AR coefficients, the mobile agent searches an AR model in the AR-ARX reference file on the sensor node, which has the smallest Euclidean distance with the calculated AR model. Once the matched AR model is found, the coefficients of the ARX model paired with the matched AR model $[\alpha_i(i = 1,2,3,4,5), \beta_j(j = 1,2,3,4,5)]$ will be used to calculate the standard deviation of the residual errors of the measurement data $\sigma(\varepsilon_y)$ and determines if damage presents based on the ratio of $\frac{\sigma(\varepsilon_y)}{\sigma(\varepsilon_x)}$. The value of $\sigma(\varepsilon_x)$ is the standard deviation of the residual errors of the matched ARX model in the reference file. If the value of $r = \frac{\sigma(\varepsilon_y)}{\sigma(\varepsilon_x)}$ is greater than a predefined threshold, a structural damage presents. Otherwise, no damage presents.

During simulated damage test, each sensor node ran mobile agent server program waiting for mobile agents. When a mobile agent arrived at a sensor node, it performed structural damage diagnosis using equipped AR-ARX algorithm and the acceleration data collected by the local sensor node. The execution of the AR-ARX algorithm was supported by the

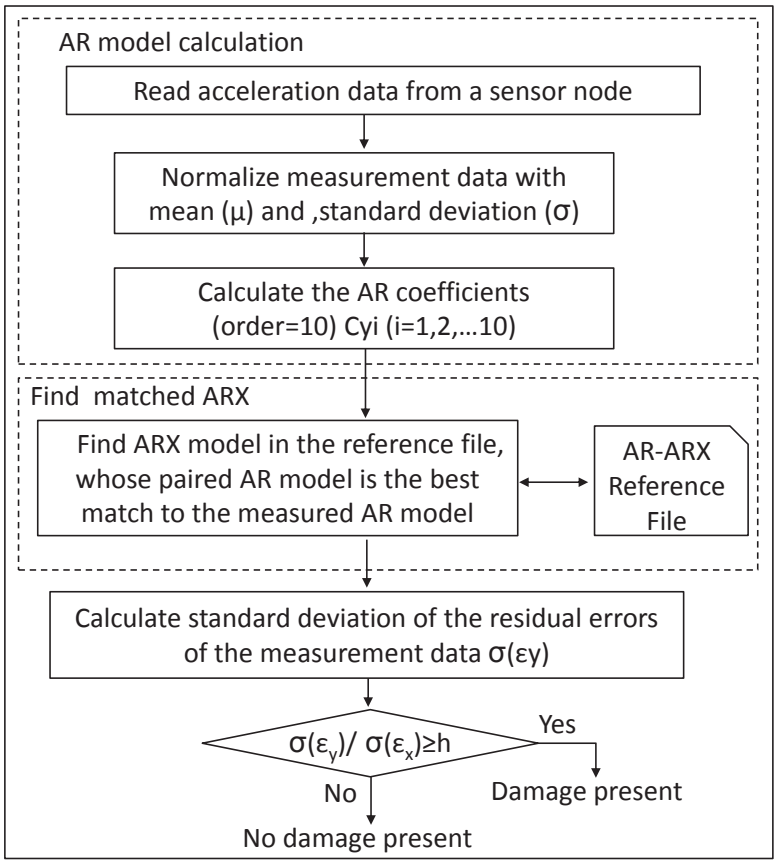


Figure 7.6: The flow chart of the mobile agent code

execution engine of the mobile agent system installed on the sensor node. Based on the off-line calculation results shown in Table 7.1, the threshold of the standard deviation ratio was selected to be 3.5 for the sensor node 1 and 5.0 for the sensor node 2, respectively. The calculated ratios of standard deviation were 3.931253 at the sensor node 1 and 6.284010 at the sensor node 2. Both ratios were greater than the threshold selected for the sensor nodes 1 and 2. As a result, structural damage was detected at both sensor nodes 1 and 2. The damage diagnosis results of both mobile agents were sent back to the mobile agent dispatcher and displayed on its terminal.

7.1.4 Discussions

The strength of the mobile agent approach in enhancing flexibility and reducing data transmission has been demonstrated in the previous sections. Damage diagnosis algorithms originally not designed in the sensor nodes, such as AR-ARX in the given example, can be dynamically added to the sensor nodes during the monitoring process without the need to interrupt normal operation. This feature provides great flexibility for an SHM sensor network to adopt newly developed diagnosis algorithms and change monitoring tasks. In addition, the local damage diagnosis does not require transmitting sensor data to a central data station. The reduction of data transmission is significant in SHM sensor networks comparing to environmental monitoring networks since SHM monitoring networks usually have a high sampling frequency. The typical accelerometer sampling rate for SHM is between 100sps to 150sps. We chose 125sps in the given example. Each acceleration reading includes two bytes of data. For the acceleration measurement, the sensor node collects data from a 3-axis accelerometer. The sensor node generates 3*125*2=750 bytes per second (45000 byte/min or 2.7 MB/hour). The size of the agent message, on the other hand, is 8722 bytes. Once a mobile agent is dispatched, it can work independently at the remote sensor nodes to perform assigned tasks.

Mobile agents, in theory, can deploy any algorithm programs on remote sensors. There are several practical issues, however, we would like to discuss in this section. Structural

damage diagnosis needs two major components: structural dynamic response data and diagnosis algorithms. The SHM algorithms typically involve intensive numerical computation for data analysis and decision making. The size of mobile agent messages and the diagnosis speed depend on the availability of numerical functions at sensor nodes. If all the necessary numerical functions are available at sensor nodes, the size of a mobile agent message will be reduced and the speed of the damage diagnosis will be increased. The reduction of the mobile agent size is obvious because the mobile agent does not need to carry on required numerical function code. For the diagnosis speed, the mobile agent code is typically executed interpretively, which is slower comparing to executing binary code. If numerical functions are integrated into binary libraries at sensor nodes, the small mobile agent code and high diagnosis speed can be achieved.

The example given is a completely distributed algorithm that only needs acceleration data from the local sensor node. For algorithms that extract damage information from multiple sensors, the time synchronized data are needed. In this case, a feasible solution is to have a mobile diagnosis agent migrate to the cluster head of a sub-network that covers the region in interest. The mobile diagnosis agent collaborates with data acquisition agents (typically stationary agents) residing in sensor nodes in the region to perform a synchronized data acquisition. Time synchronized data acquisition for wireless sensor networks has been investigated extensively. A number of synchronization schemes [119, 120, 121, 122] are proposed to achieve network-wide synchronization.

7.2 Driving Cycle Pattern Recognition

A driving pattern is typically defined as the driving cycle of a vehicle in a particular environment [123]. In this research, supervised pattern recognition approach is studied for the classification of a real-world driving cycle to a similar driving cycle in the representative driving cycle group. Four federal driving cycles, Urban Dynamometer Driving Schedule (UDDS), Highway Fuel Economy Driving Schedule (HWFET), a high acceleration aggressive driving schedule (US06), and an air conditioning driving schedule (SC03), are selected as representative driving cycles. These driving cycles represent different street types, driver behavior, and weather condition. With pattern recognition method, driving cycles and environmental information for various driving patterns are represented by corresponding features vectors. The classification is based on the distance of a test feature vector (test driving cycle) to the representative feature vectors (representative driving patterns). The test driving pattern is classified to one of representative driving pattern with which the test driving pattern has the smallest distance.

7.2.1 Feature Selection for Driving Cycle Pattern Recognition

To classify driving cycles, thirty-nine characteristic parameters were initially chosen based on Jeon and Ericsson's work [124, 125]. The high dimension of feature vectors,

however, impedes practical application of driving cycle pattern recognition in real-time. To reduce the dimension of feature vectors, numerous simulation tests were performed to find a reduced set of feature members and the weighting factor for each feature member. The simulation work finally identified fifteen feature members listed in Table 7.2 and corresponding weighting factors to form feature vectors for driving cycles as shown:

$$f = (k_1 \times a_1, k_2 \times a_2, \cdots, k_i \times a_i, \cdots, k_{15} \times a_{15}),\ i \in [1, 15] \tag{7.6}$$

where $a_i$is a feature member and $k_i$is a weighting factor.

7.2.2 Representative Feature Vectors for Selected Driving Cycles

Four federal driving cycles, UDDS, HWFET, US06, SC03, are selected as representative driving cycles. The profiles of these four driving cycles are shown in Figure 7.7. A feature vector of a driving cycle is calculated based on partial data points of a driving cycle for the quick recognition of driving cycle patterns, which is especially valuable in real-time applications. During real-time driving cycle pattern recognition, vehicle controllers collect a number of data points defined by the size of the sample window and calculate a feature vector for the current sample window using parameters defined in Table 7.2. The real driving cycle is then classified to one of four representative driving cycles

Table 7.2
Driving cycle feature parameters and corresponding weighting factors

	Feature Parameters	Weighting Factor
1	Average Cycle Speed (m/s)	10
2	Positive Average Acceleration (a>0.1m/s^2)	1
3	Low Speed Time (15-30 Km/h)/Total Time (%)	10
4	Mid High Speed Time (70-90 Km/h)/Total Time (%)	100
5	High Speed Time (>90 Km/h)/Total Time (%)	10
6	Extreme Deceleration Time (a<-2.5 m/s^2)/Total Time (%)	1000
7	High Deceleration Time (a<-2 & a>-2.5 m/s^2)/Total Time (%)	1
8	Maximum Cycle Acceleration (m/s^2)	100
9	Maximum Cycle Speed (Km/h)	6
10	Standard Deviation of Cycle Speed (Km/h)	1
11	High Deceleration Time (a<-2 & a>-2.5 m/s^2)/Total Time (%)	1000
12	Mid High Deceleration Time (a>-2 & a<-1.5 m/s^2)/Total Time (%)	1000
13	Mid Acceleration Time (a>1.5 & a<2 m/s^2)/Total Time (%)	1
14	High Acceleration Time (a>2 & a<2.5 m/s^2)/Total Time (%)	1000
15	Extreme Acceleration Time (a>2.5 m/s^2)/Total Time (%)	1000

using classification algorithms and calculated feature vector.

To generate the representative feature vectors for the UDDS, HWFET, US06, and SC03, the velocity data for these driving cycles were downloaded from the U.S. Environmental Protection Agency website [126]. From each driving cycle, the most representative data segment was used to generate feature vectors for the corresponding driving cycle. For the UDDS, velocity data starting from 347 to the end of the driving cycle was used. Since these representative segments have different length, a common data length of 2000 data

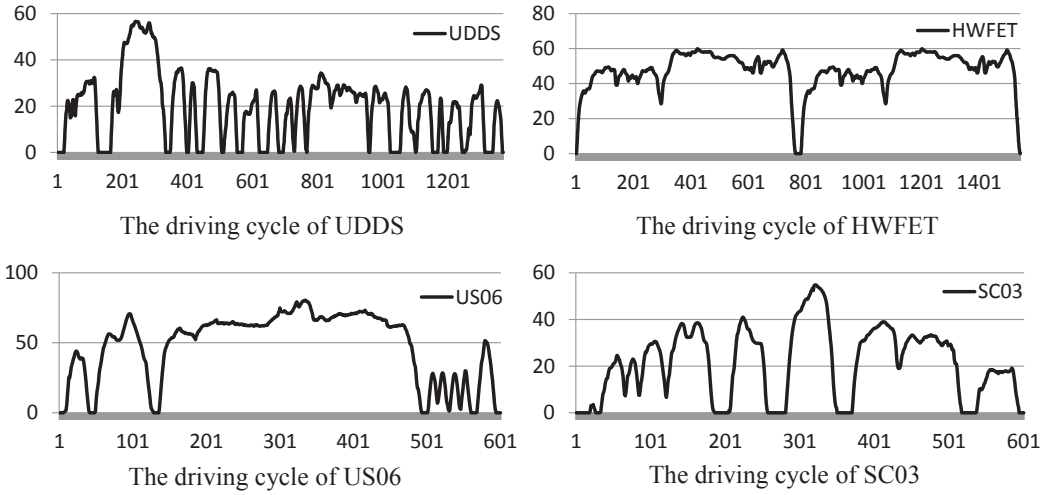


Figure 7.7: Four federal driving cycles

points was specified to ensure the same number of representative feature vectors for each driving cycle. The representative segment for each driving cycle was repeated to form a data set with a length of 2000. In each data set, the first 450 data points were used to form the first sample window and find the first feature vector using equation (2). The second feature vector was calculated by advancing the sample window by 50 data points. From equation (2) we can see that the dimension of feature vectors is 15. To display high dimensional driving cycle feature vectors, the Principal Component Analysis (PCA) algorithm was applied to the generated representative feature vectors. The first and second principal components were then used to plot representative feature vectors in 2-dimensional space. Figure 7.8 shows the distribution of representative feature vectors for the selected four driving cycles.

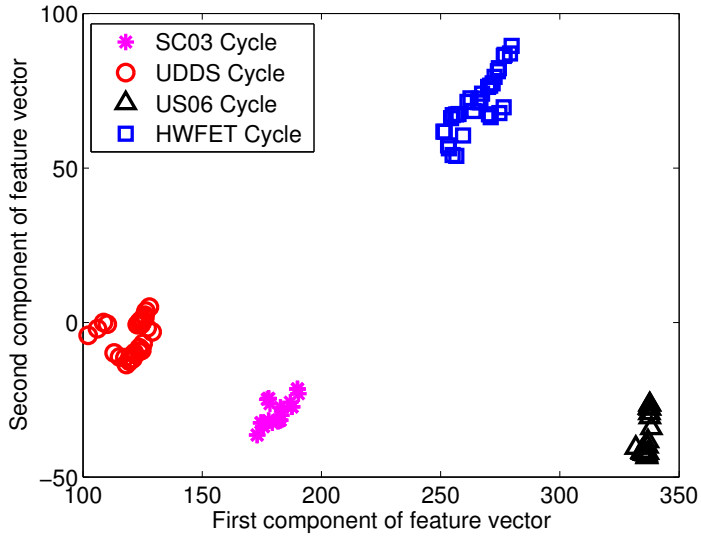


Figure 7.8: The distribution of the feature vectors of selected four driving cycles

7.2.3 Classify Real World Driving Cycles Using Representative Feature Vectors

To validate the effectiveness of representative feature vectors for real-world driving cycle pattern recognition, two real-world driving cycles were adopted for the performance test. The two real-world driving cycles were collected by a mild-hybrid Chevy Malibu driven in the urban and suburban area near the downtown Hancock. The speed profiles for these two driving cycles are shown in Figure 7.9 and Figure 7.10. To classify real-world driving cycles to one of the selected four driving cycles, the feature vectors for the RW-CC and RW-HC were generated using the feature extraction method described in previous Section. The classification is to identify to which pattern the test pattern belongs. The k-Nearest

Neighbor (kNN) algorithm was employed for the driving cycle classification. For a test feature vector , the nearest neighbor rule is summarized as follows. (1) Calculate the distances of the test feature vector x to each of representative feature vectors shown in Figure 7.8. (2) Identify the k nearest neighbors of representative feature vectors to the vector x. The number of k is general not to be a multiple of the number of classes M. (3) Out of these $k$samples, find out the number of feature vectors, k_i, which belong to class ω_i, $i = 1,2,\cdots,M, \sum_i k_i = k$. (4) Label x to the class ω_i which has the maximum number k_i of the samples. In the real-world driving cycle test, the value of M is 4 and the number of k is chosen to be 13. The distances between the test feature vector and representative feature vectors was calculated by the Euclidean distance.

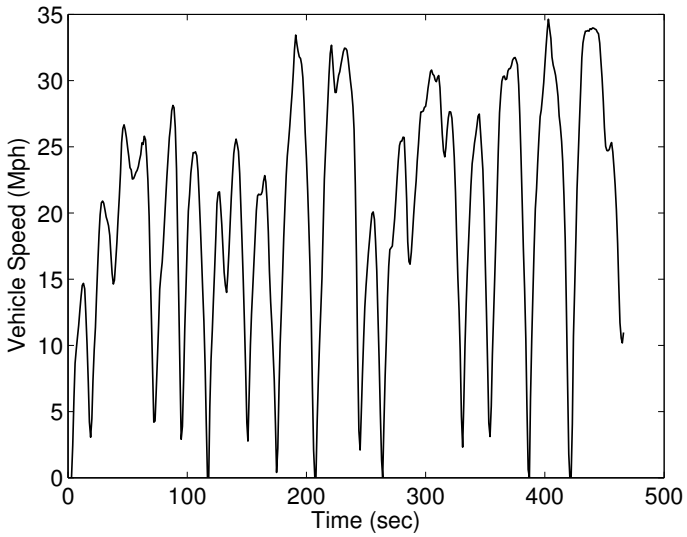


Figure 7.9: The speed profile of city cycle

The success rates of classifying the RW-CC to the UDDS category and the RW-HC to the

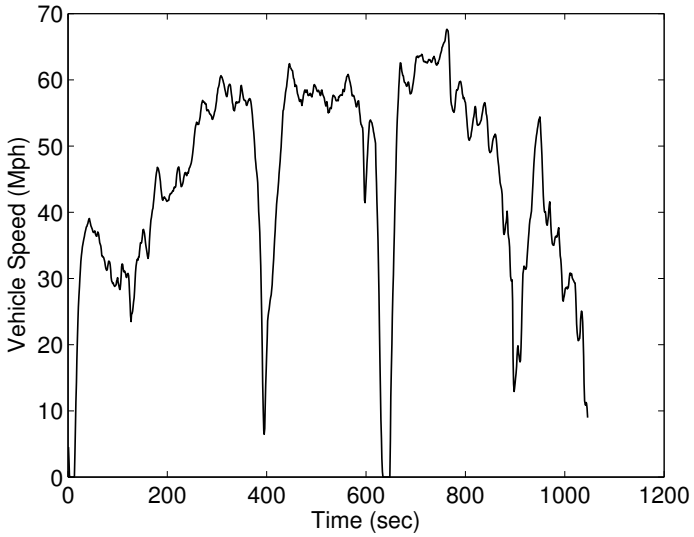


Figure 7.10: The speed profile of highway cycle

HWFET are shown in Figure 7.11. In the classification test, the success rate of driving cycle pattern recognition was calculated with different size of the sample window. The tested sample window sizes include 50, 100, 150, 200, 250, 300, 350, 400, and 450. Figure 7.11 shows that the larger size of the sample window has a higher pattern recognition success rate for both RW-CC and RW-HC. In addition, the success rates of the RW-CC are higher than RW-HC. This is due to the fact that the similarity of the RW-CC with the UDDS is higher than the similarity of the RW-HC with the HWFET.

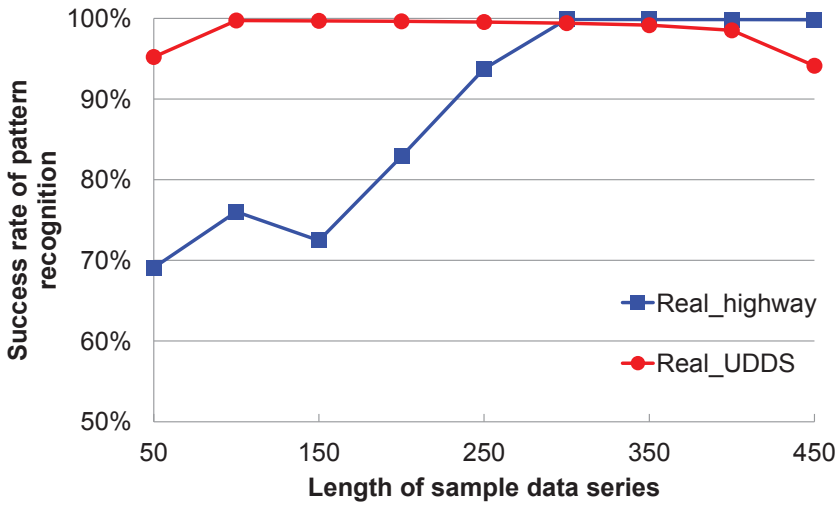


Figure 7.11: The success rate of real world driving cycle pattern recognition

7.2.4 The Impact of Dissimilarity Measures on Driving Cycle Pattern Recognition

To test the impact of the dissimilarity measure (distance between feature vectors) on the performance of the pattern recognition, a number of dissimilarity measures were tested for the driving cycle pattern recognition. The tested dissimilarity measures include Euclidean distance, Chebyshev distance, Cosine distance, Correlation distance, and Mahalanobis distance. Figure 7.12 shows the success rate of RW-CC pattern recognition with various dissimilarity measures using kNN 13 classification method. From Figure 7.12 we can see that the Euclidean distance is the only dissimilarity measure that has good pattern recognition performance for the RW-CC pattern recognition. Other dissimilarity measures, including Cosine distance, Correlation distance, and Mahalanobis distance have very

bad performance for the RW-CC pattern recognition. The average success rate of the Chebyshev distance is only about 60% and it fluctuates significantly. Figure 7.13 shows the success rate of RW-HC pattern recognition with aforementioned dissimilarity measures. The Mahalanobis distance shows the best performance of the RW-HC pattern recognition. The Euclidean distance and the Chebyshev distance also show the good performance when the sample window size is larger than 300 data points.

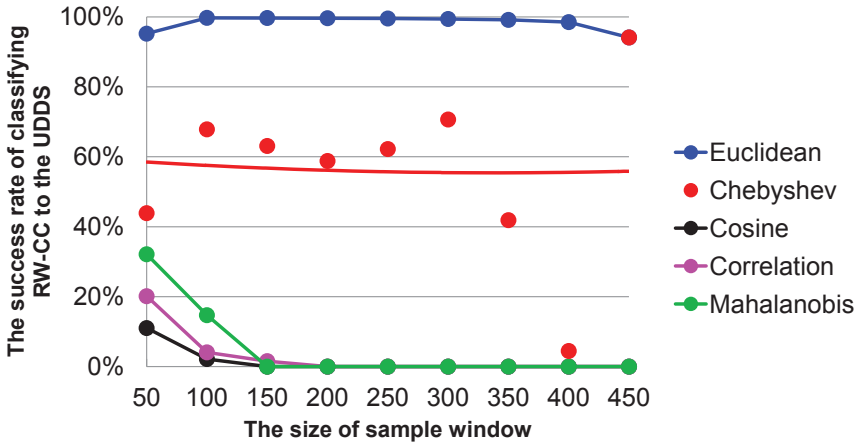


Figure 7.12: The success rate of RW-CC pattern recognition

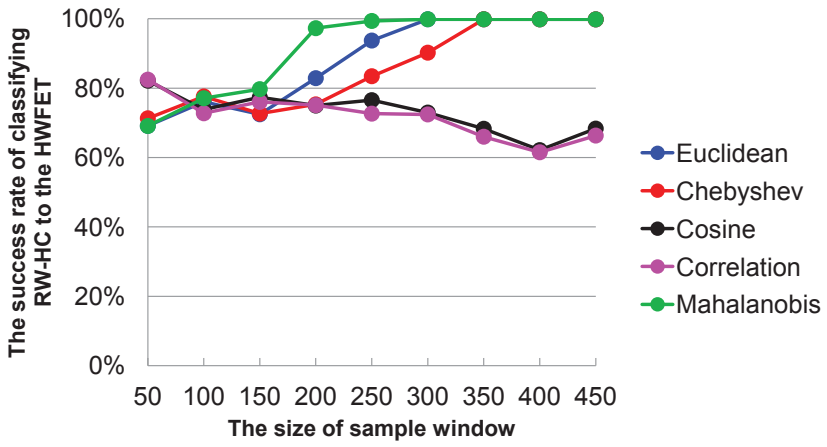
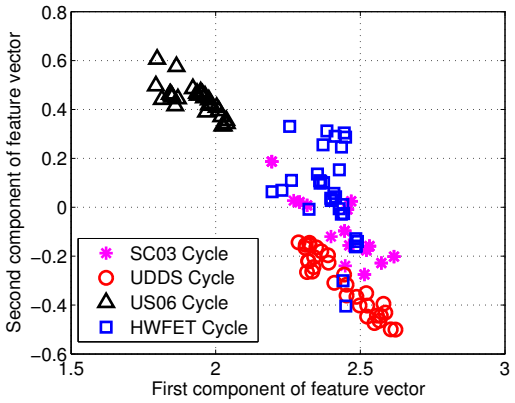


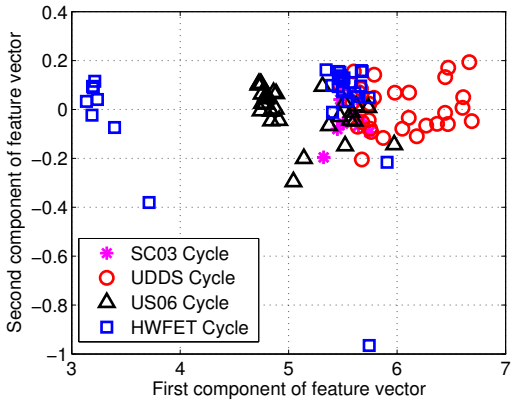
Figure 7.13: The success rate of RW-HC pattern recognition

7.2.5 The Impact of Feature Extraction Methods on the Performance of Driving Cycle Pattern Recognition and the Quality of Representative Feature Vectors

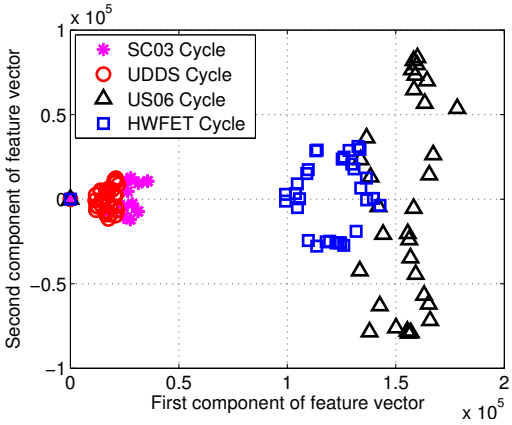
In this section, the performance of the autoregressive, DFT, and DWT feature extraction methods for the driving cycle pattern recognition is studied. Autoregressive, discrete Fourier transformation, and discrete wavelet transformation feature extraction methods were applied to the UDDS, HWFET, US06, and SC03 driving cycle data. The generated representative feature vectors for these four driving cycles are shown in Figure 7.14(a) to Figure 7.14(c).



(a) The distribution of AR-based feature vectors

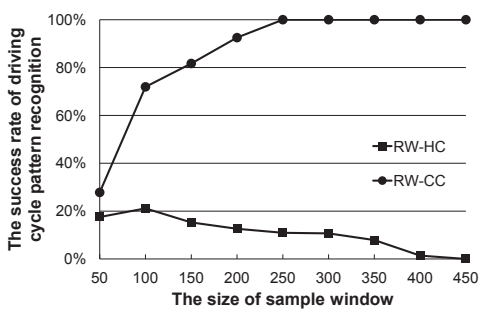


(b) The distribution of DFT-based feature vectors

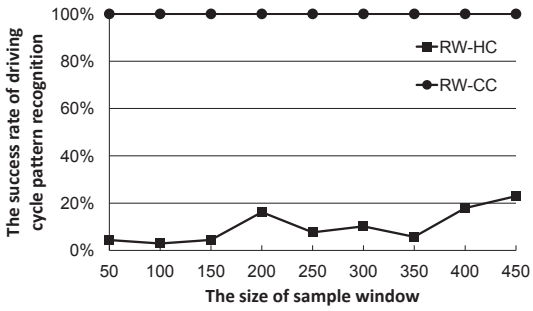


(c) The distribution of DWT-based feature vectors

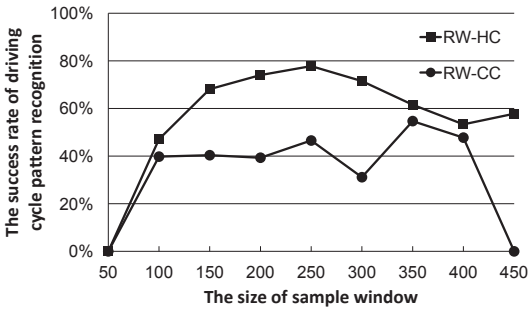
Figure 7.14: The distribution of feature vectors based on different feature extraction methods



(a) Success rate of pattern recognition using AR based feature extraction method



(b) Success rate of pattern recognition using DFT based feature extraction method



(c) Success rate of pattern recognition using DWT based feature extraction method

Figure 7.15: Success rate of pattern recognition using different feature extraction methods

The features vectors generated using AR, DFT, and DWT methods are not well separated. With these representative feature vectors, the success rates of driving cycle pattern recognition were tested using kNN-9 classification method and Euclidean distance. The test results are shown in Figure 7.15(a) to Figure 7.15(c). As we can see from these figures, the success rates, in generally, are lower than the pattern recognition success rate using representative feature vectors shown in Figure 7.8. Although the DFT feature extraction method has high success rates for the RW-CC pattern recognition, the success rates for the RW-HC pattern recognition is extremely low.

7.3 Web-based Structural Health Monitoring Networks

Wireless monitoring network approach is emerging as a feasible solution for structural health monitoring (SHM). While sensor network approach presents a number of advantages, the structural damage detection over a sensor network faces many challenges due to the severe limitations of communication bandwidth and sensor node computational capabilities. Different from sensor networks for wildlife habitat monitoring, SHM sensor networks generate a huge amount of measurement data. To monitor the performance of a structure, these data are typically transferred to a central data station for processing and analysis. The transmission of a large amount of data over wireless sensor networks is challenging. In addition, data transmission will cause a response delay of a system to extreme events.

Web-based approach has gained wide applications in industry and education, such as web-based enterprise resource planning data mining for decision making [127], quality management in computer assembly industry [128], construction information management [129], and control system design and analysis [130]. Conventional web applications implement web servers on resource rich computers for intensive numerical computation and information management. This centralized web server approach is also adopted in the most existing web-based sensor networks, which requires transferring sensor data to the central web server. The severe limitation of the communication bandwidth in

wireless sensor networks impedes the practical application of the centralized web server approach. Motivated by the advances of the embedded systems, a number of researchers have investigated to embed web services in embedded computers, controllers, and devices [115, 131, 132].

7.3.1 The Architecture of the Web-based Structural Health Monitoring Sensor Networks

The architecture of the presented web-based SHM sensor network is shown in Figure 7.16. The integration of a wireless sensor network with the World Wide Web (WWW) technology enables end users to remotely perform structural health monitoring. The architecture of the system adopts a typical client-server architecture [133]. An end user, at the client side, accesses remote sensor data and requests a certain type of data processing or structural damage diagnosis. The distributed web servers in the wireless sensor nodes or the data repository machine process the user's requests and send the requested information back to the client's web browser. The distributed sensor nodes collect the dynamic response data of a structure and selectively transfer raw data or processed data to a data repository machine. The system not only allows end users to view sensor data, but also perform sensor data analysis and damage diagnosis. Sensor data can be analyzed either online on the distributed sensor nodes and the data repository machine or offline by downloading the sensor data to client machines. The Gumstix provides web services through a Boa

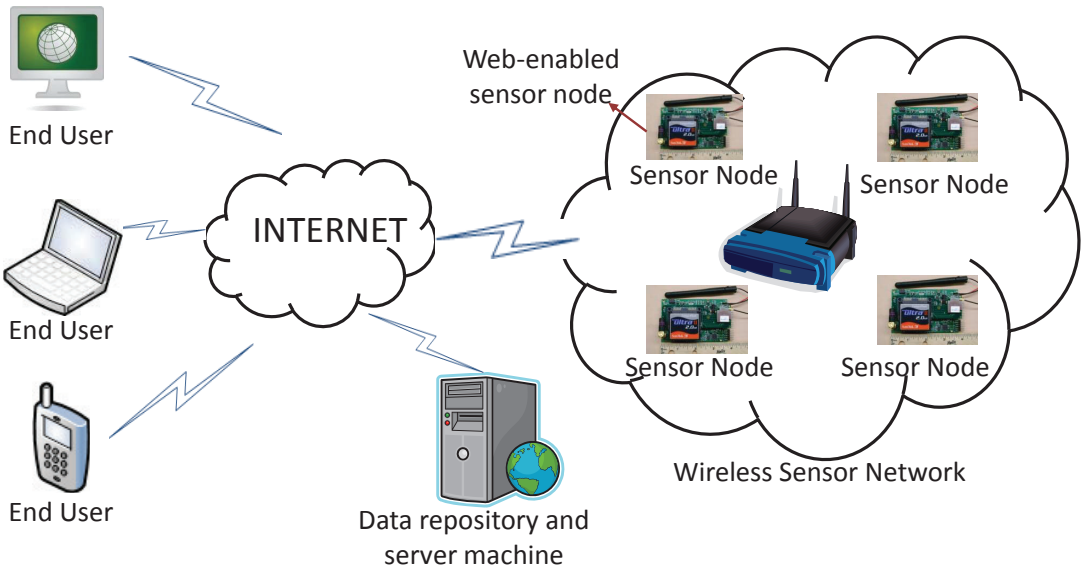


Figure 7.16: The architecture of the web-based structural health monitoring sensor network

HTTP server [132]. Boa is one of the lightweight server software and has minimal hardware resource requirement, which makes it suitable for embedded applications. The Boa supports the Common Gateway Interface (CGI) [134]. The CGI is a standard that defines how information is exchanged between a web server and a CGI program running on the server. A web server receives an input from a client, and passes the input to a CGI program. Then, the CGI program is executed in the web server to retrieve requested information and send it back to the client [135]. The web service on a wireless sensor node is shown in Figure 7.17. When a Boa web server receives a request for sensor data visualization from an end user, the web server passes this request to a CGI program with data parameters specified by the end user through the CGI. The CGI program is executed to read the sensor data from corresponding sensor data files and generate a data plot that is sent back to the client computer and displayed in a web browser. The end user can also

request remote sensor node to process real-time sensor data or perform damage diagnosis based on the sensor data available in the sensor node.

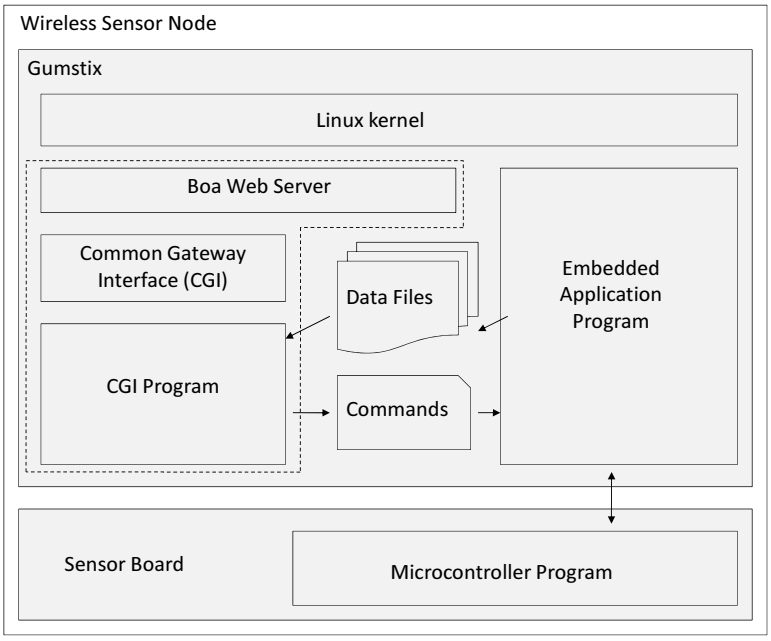


Figure 7.17: Web service in a wireless SHM sensor node

7.3.2 Data Management in a Web-enabled Sensor Node

The sensor board connects to several types of sensors, including accelerometer, strain gauge, piezoelectric transducer, humidity, and temperature. The accelerometers and strain gauges are usually used for passive sensing in structural health monitoring while piezoelectric sensors are used for active sensing. The sensor data flow in wireless sensor nodes is shown in Figure 7.18. The microcontroller program that runs on the sensor board includes a communication module and a data acquisition module. The data acquisition

module is responsible for collecting data from sensors and storing the data to an external SRAM memory buffer. To manage the buffer effectively and avoid the data collision, the buffer is divided into four portions. The size of each portion is 60KB. The upper three 60KB memory blocks are allocated for storing acceleration data; each block for one axis. The strain data and the impedance data share the lowest 60KB memory block (The strain sensor and the impedance sensor work at different times). Each 60KB memory portion is further divided into two sub-buffers. The microcontroller continuously saves sensor data to one of these two sub-buffers. When one sub-buffer is full, the sensor data is saved to another sub-buffer, and the data in the full sub-buffer are transmitted to the Gumstix board.

The communication between the Gumstix and the sensor board employs the I^2C communication protocol. The Gumstix acts as the master device and initiates the communication. The sensor board works as the slave device and responds to the commands from Gumstix. Since the Gumstix doesn't know when the sensor data are ready for transmission, an interrupt signal is generated when one of the sub-buffer in the sensor board is full to notify the Gumstix. The embedded application program running on the Gumstix, then, initiates data transmission process to read sensor data from the sensor board and save the data to data files. A 2 GB compact flash memory mounted on the WiFi board is used to store these data files. When a sensor node operates in a continuous passive sensing mode with a sampling rate of 125sps, one data file is generated for each minute. For ease of searching data, the data files are named in a form of "year-month-day-hour-minute".

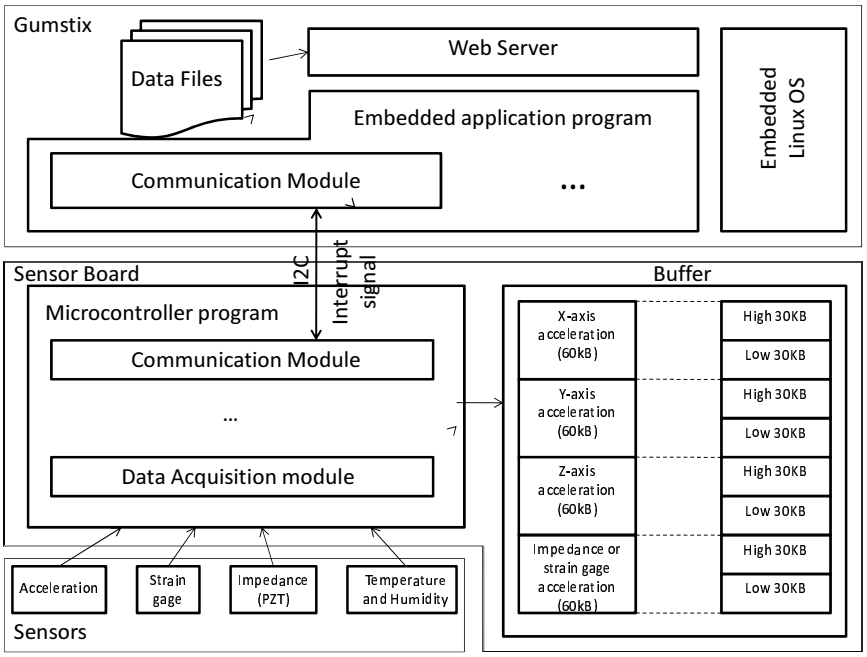


Figure 7.18: The data flow in SHM sensor nodes

7.3.3 Design of Web Functions

The main goals of the web services are to remotely observe real-time sensor data, perform data analysis, and diagnose structural damage. Users can request these services through a web interface in client machines by entering information, such as sensor ID, sensor data type, and the time period of interest. Because these data processing and damage diagnosis tasks can be performed at the sensor node level, the system exhibits good scalability. The designed web functions include four categories as shown in Figure 7.19. A unique feature of the presented web-based SHM system is its ability to visualize distributed sensor data and perform remote damage diagnosis over the internet. The following section illustrates the web interface for the sensor data visualization.

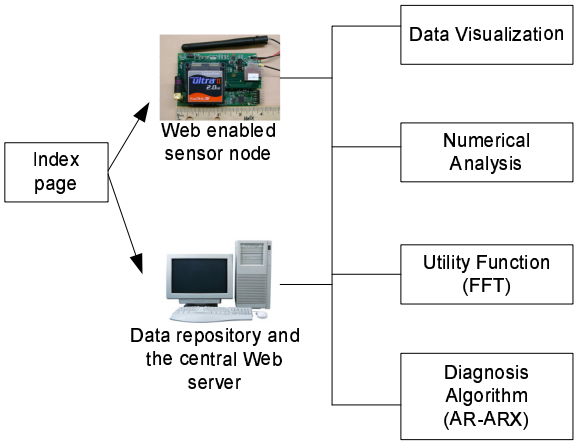


Figure 7.19: Overview of web function categories

7.3.3.1 Web Interface for Sensor Data Visualization

Users can access either recent sensor data on remote sensor nodes or historical data in the data repository machine through a web interface. Based on users' inputs (sensor ID, sensor data type, and the time period), the web system will generate a data plot for the requested data set and send the plot back to the client machine to be displayed in the user's web browser. On the index page, users are requested to input sensor ID to specify which sensor the users are interested in. The ID for the data repository machine is defined as 0. Each sensor in the sensor network has a unique ID. By selecting different sensor nodes, users can monitor data collected from different locations. By clicking the "Data Visualization" hyperlink on the function selection web page, a new web page is brought up for the users to choose the sensor data type and the time period of interest. There are several sensor data types in each sensor node, such as acceleration, impedance, and temperature. Users can choose to plot different type of sensor data using the radio buttons on the web page. In

addition, users need to specify the start and end time of the sensor data. After entering the required information and clicking the "Submit" button, the requested sensor data plot will be presented to the users on the following page.

7.3.3.2 Software Design for the Web Functions

The web-based sensor information inquiry is achieved through the interactions of clients and web servers as shown in Figure 7.20. When a remote user request a web service from a client web browser, the web server will generate a web form that allows the remote user to input required information for performing requested web service. For example, when a user requests for sensor data visualization, the user needs to specify sensor data type and time period of interest in a web form. The data plot parameters submitted in a web browser are encoded by the browser. These parameters are decoded by a member function CRequest::getFormNameValue() in the data visualization CGI program as shown in Program 1. For the web plotting, a plotting CGI program is needed, which reads the sensor data from sensor data files and generates a plot as requested. To pass user inputs to the plotting CGI program, code segment shown in Program 2 is used to encode name-value pairs to query strings in the data visualization CGI program. These parameters are passed to the plotting CGI program and decoded again using the member function CRequest::getFormNameValue(). The time period input in our system is used to locate the sensor data files. To find the sensor data files easily, the sensor data files are named as

Table 7.3
Program 1. Code segment for decoding user input

```
chstrarray name, value;
class CResponse Response;
class CRequest Request;
Request.getFormNameValue(name, value);
```

year-month-date-hour-minute. With the user's time period input, the plotting CGI program opens the corresponding sensor data files and generates a plot using Ch plotting class. The generated plot in a PNG file format is displayed in the client's web browser by a new HTML file generated by the data visualization CGI program.

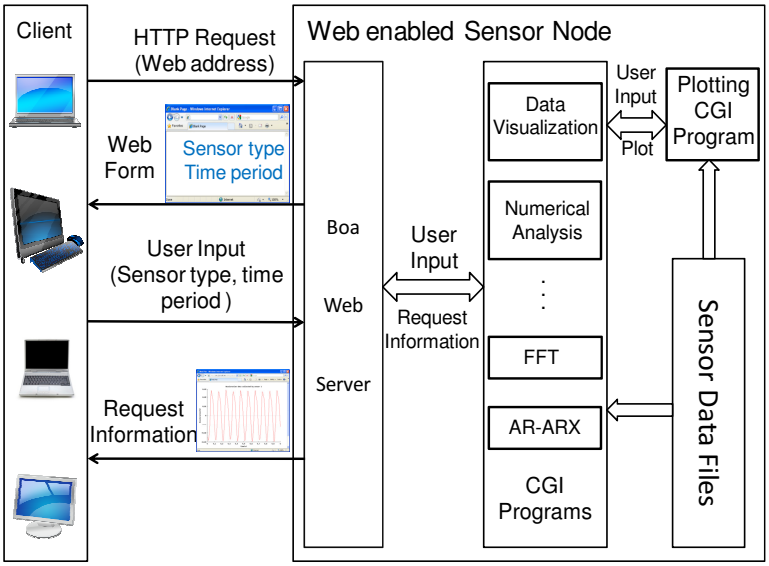


Figure 7.20: Web-based client-server interaction

Table 7.4
Program 2. Code segment for encoding data

```
for (i=0; i<num; i++){
// transferring data between CGI programs
putc(i==0 ? '?' : '&', stdout);
fputs(Server.URLEncode(name[i]), stdout);
putc('=', stdout);
fputs(Server.URLEncode(value[i]), stdout);
}
```

The file system of the web-based SHM sensor network is shown in Figure 7.21. The file system contains two types of files: HTML files and CGI programs. HTML files can be displayed in clients' web browsers and used as an interface of a web system with human operators. CGI programs, on the other side, accept users' input from HTML forms through CGI, perform defined tasks, and generate a new HTML file to report request information to the remote users. In our system, the index page (HTML file) of the system is installed in the central web server (although each sensor node has its own index page for direct sensor node web access). The action of this HTML file will generate a new HTML file which allows users to select different types of web functions, such as data visualization, FFT, and AR-ARX. Each function is implemented by both HTML files and CGI programs. The HTML files and CGI programs for all the functions are available in each sensor node and the central web server. For web services in sensor nodes, the HTML files and the corresponding CGI programs are installed in local sensor nodes. For instance, the FFT hyperlink on function selection page is linked to an HTML file fft.html located in the sensor node with an ID specified by the user on the index page of the sensor ID entry. When a user clicks on the FFT hyperlink, the fft.html in the specified sensor node will be sent to

the client's web browser to ask for user inputs. When the user fills out the form fft.html and click the "Submit" button, the corresponding CGI program, fft.ch, will be called. Since the output of the FFT function is a plot, the fft.ch will invoke a plotting CGI program to generate a FFT plot. With the FFT plot generated by the plotting CGI program, the fft.ch generates a new HTML file to display FFT plot in the user's web browser.

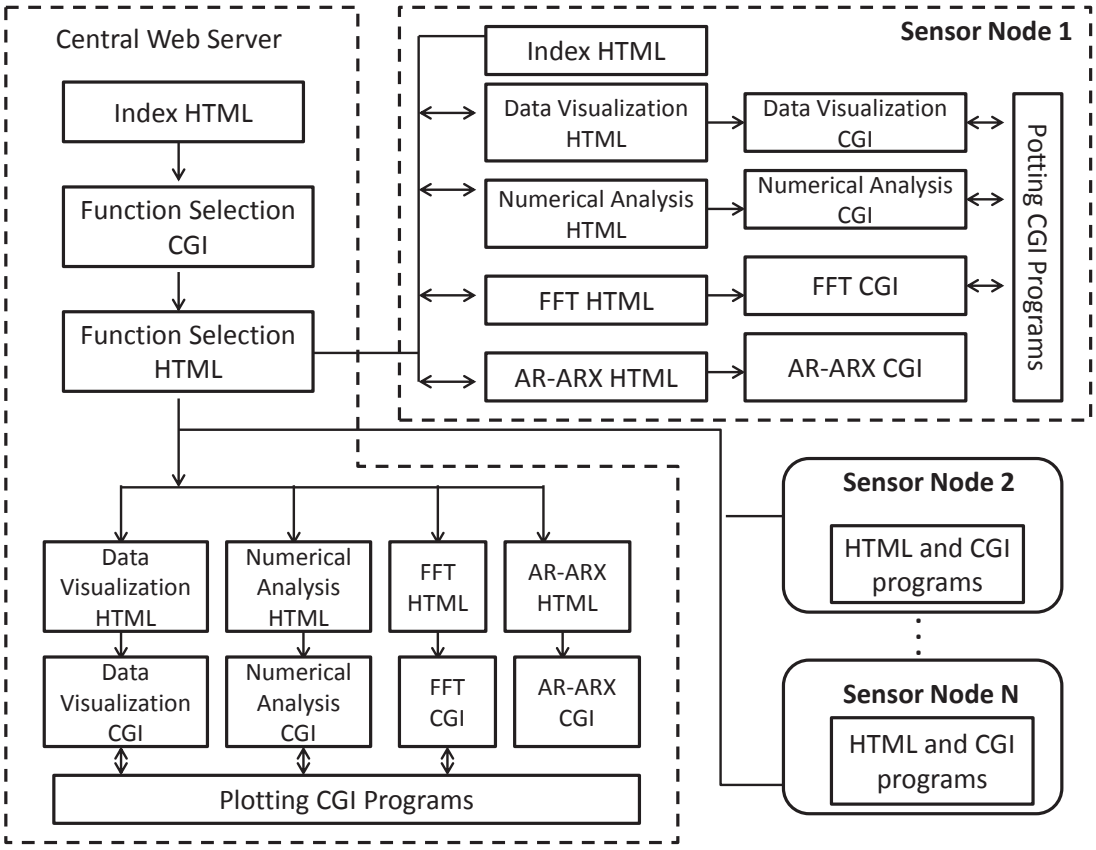


Figure 7.21: The file system of the web-based SHM sensor network

7.3.3.3 Experimental Validation of the Web-based SHM Sensor Networks

The presented web-based system was tested on a scaled steel bridge and the accelerometers were deployed along the test bridge. During test, the bridge was excited by a shake in vertical direction. The excitation signals of the shaker were generated by Siglab and virtual instruments.The excitation signals were 10 Hz sinusoid waves and peak-to-peak voltage of 0.255 volts, 0.275 volts, and 0.295 volts, respectively. One cross beam in the center of the bridge was removed to simulate the damage to the bridge as shown in Figure 7.5. Two sensor nodes, sensor nodes 1 and 2, were chosen for the system validation test. For both of normal and damage patterns, acceleration data collected by these two sensors were recorded for offline calculation.

To view real-time acceleration data through the presented web interface, users could click the "Data Visualization" hyperlink on the function selection page as shown in Figure 7.3.3.3. On the data visualization page shown in Figure 7.23, the sensor data type, acceleration in this case, and the time period of interest are specified. By clicking the "Submit" button, the acceleration signal of the sensor node 1 (number 1 is entered in the sensor ID entry on the index page) is displayed on the next web page. The enlarged signal shows the frequency of the acceleration is 10 HZ, the same as the frequency of the excitation signal of the shaker. The web page shown in Figure 7.23 can only specify one data set. For the visualization of two data sets on the same web page, users can click the

"click here" link at the bottom of this web page. A new web page shown in Figure 7.24 is brought up for users to specify two data sets for plotting. Upon entering the parameters of two data sets and clicking the "Submit" button, the acceleration signals at sensor node 1 for both normal pattern and simulated damage pattern are displayed on the next web page as shown in Figure 7.25. The amplitude of the acceleration signal in simulated damage pattern is larger comparing to the normal acceleration signal.

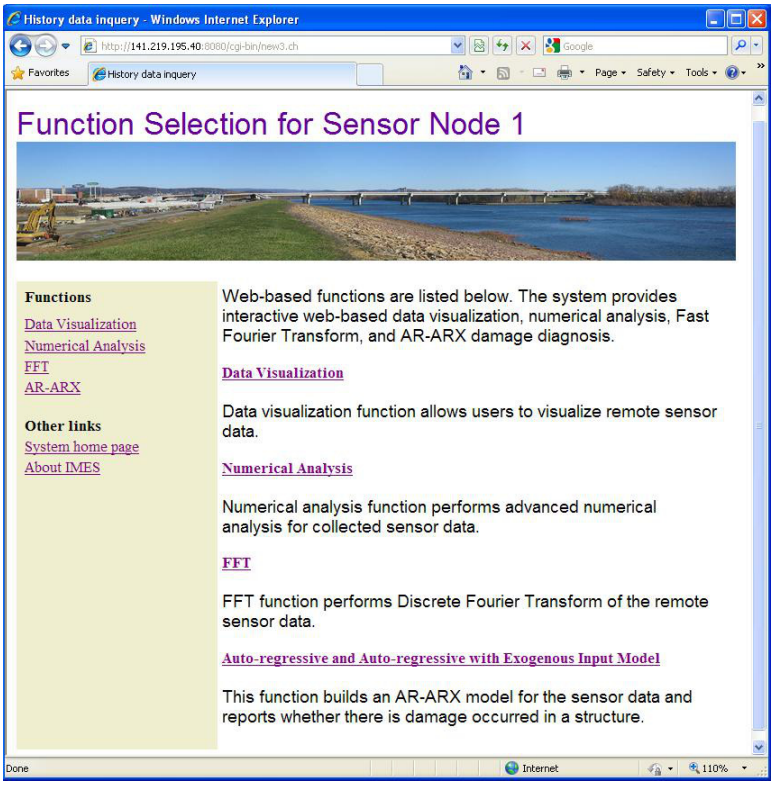


Figure 7.22: Web function selection page

To find the modal frequencies of the acceleration, the function of the Fast Fourier Transform is used to compute the discrete Fourier transform of the acceleration signal. By selecting the "FFT" function on the function selection page, a new web page, which is similar to the web

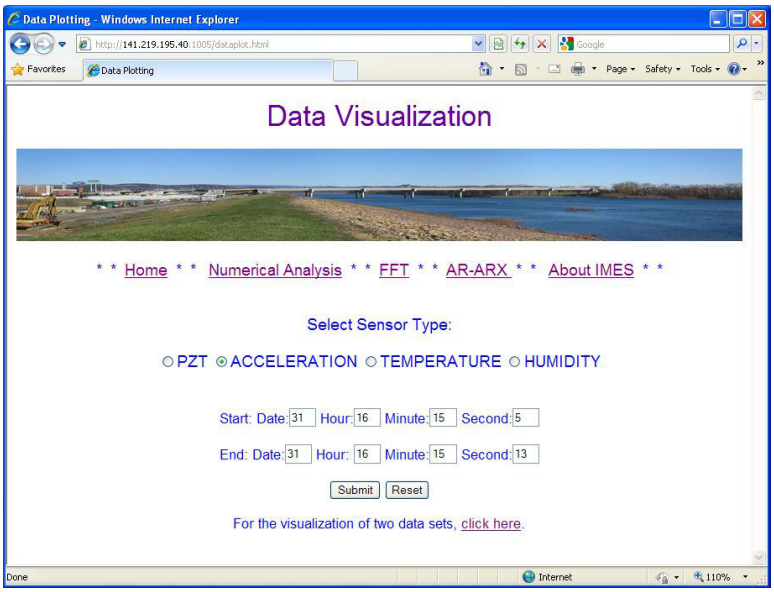


Figure 7.23: Sensor data type and time period

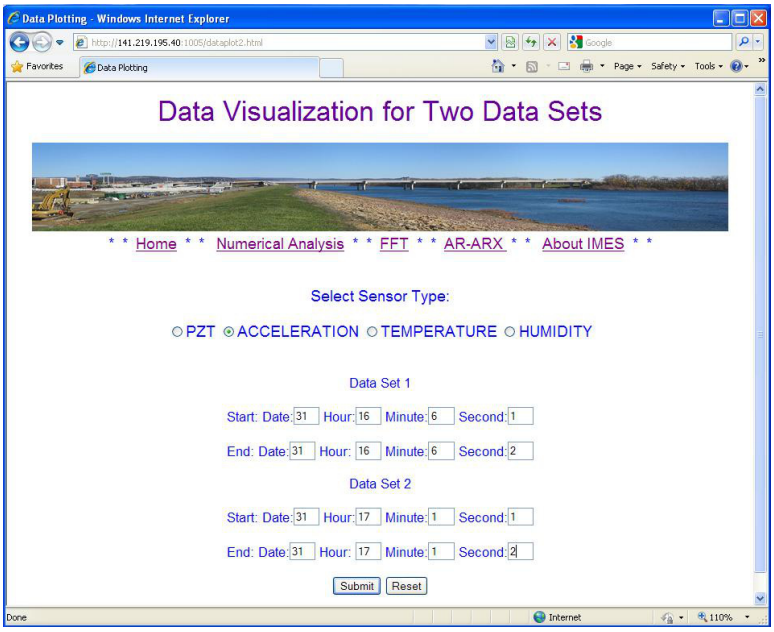


Figure 7.24: Web page for specifying two data sets

page shown in Figure 7.23, will be brought up to allow users to specify acceleration data by defining sensor data type and the time period of interest. The selected acceleration data are computed by the FFT algorithm and the resulting FFT plot is reported to users as shown

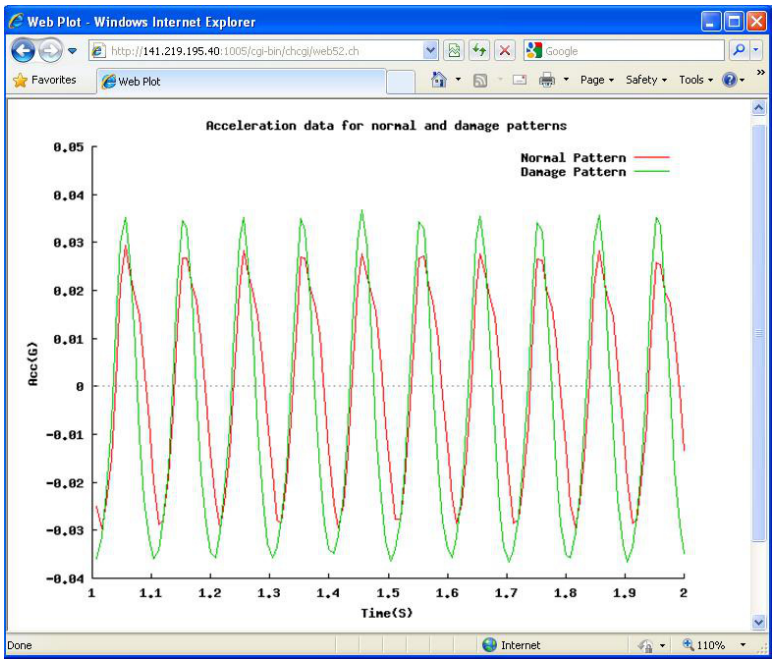


Figure 7.25: Acceleration signals collected by the sensor node 1 for both normal and simulated damage patterns

in Figure 7.26. The FFT plot shown in Figure 7.26 indicates that the major frequency of the acceleration signal is 10 Hz, which is the same as the frequency of the excitation signal of the shaker. To generate an AR-ARX reference file for damage diagnosis using AR-ARX method, acceleration data of the steel bridge structure were collected with various voltage levels of the shaker excitation signals. At each excitation voltage level, a total of 30 acceleration time series on each sensor node were recorded for the normal pattern, while 15 time series were recorded for the damage pattern. The AR-ARX reference file was constructed by using 10 normal acceleration time series and 5 damaged acceleration time series. The standard deviation of the residual time series of rest 20 time series for the normal pattern acceleration data and 10 time serials for the damage pattern were calculated off-line for the comparison of the web results. The calculated values of the ratio, $r = \frac{\sigma(\varepsilon_y)}{\sigma(\varepsilon_x)}$,

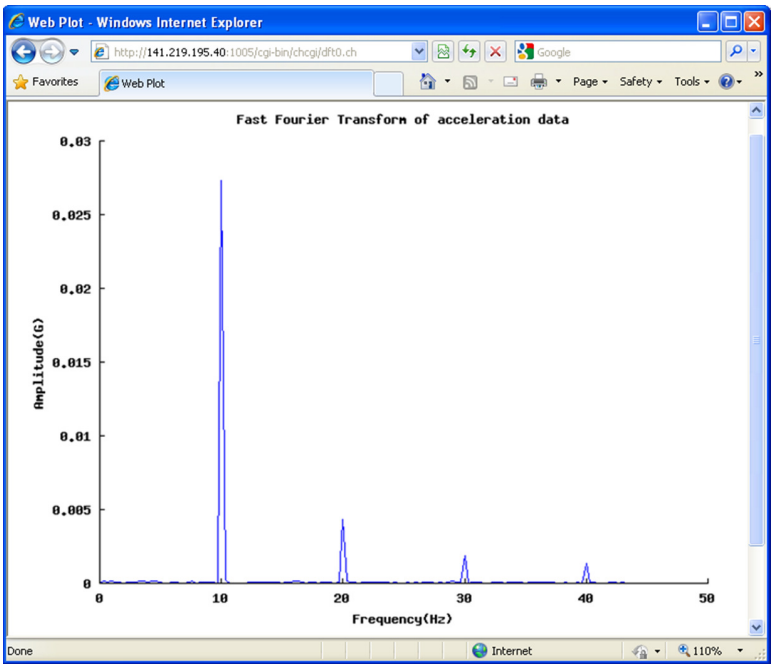


Figure 7.26: Output of FFT analysis

are listed in Table 7.5 when the excitation signal voltage is at 0.255 volts. Table 7.5 shows that the value of r is about 1.0 for the normal pattern and 15.0 for the damage pattern at sensor node 1. At sensor node 2, the value of r is less than 1 for the normal pattern and greater than 2 for the damage pattern. The web AR-ARX result shown in Figure 7.27 for the sensor node 1 in the normal pattern is within the normal range comparing to the offline calculation results shown in Table 7.5.

With this web-based system, users can view real-time sensor data and perform damage diagnosis remotely through a web interface. Different from most existing web-based sensor networks, which transmit sensor data to a central web server, the presented web-based SHM

Table 7.5
Off-line calculation of the ratio of standard deviation of the residual time series

Sensor	Pattern	Ratio $\frac{\sigma(\varepsilon_y)}{\sigma(\varepsilon_x)}$
1	Normal	0.9724 0.9661 0.9629 0.9766 0.9658 0.9564 0.9538 0.9384 0.7987 0.7661 0.7719 0.7673 0.7442 0.7399 0.7395 0.7418 0.7718 1.1137 1.0726 1.1630
	Damage	15.8678 15.8597 15.8695 15.8646 15.8124 15.8120 15.8425 15.7976 15.8049 15.6335
2	Normal	0.9978 1.0234 0.9675 0.9988 0.9885 0.9900 0.9560 0.9638 0.9547 0.9488 0.9334 0.9295 0.9247 0.9373 0.9114 0.9223 0.9430 0.9509 0.9389 0.9603
	Damage	3.5629 4.6960 4.7539 6.6498 6.7670 3.7302 2.7484 2.5665 3.8426 2.3477

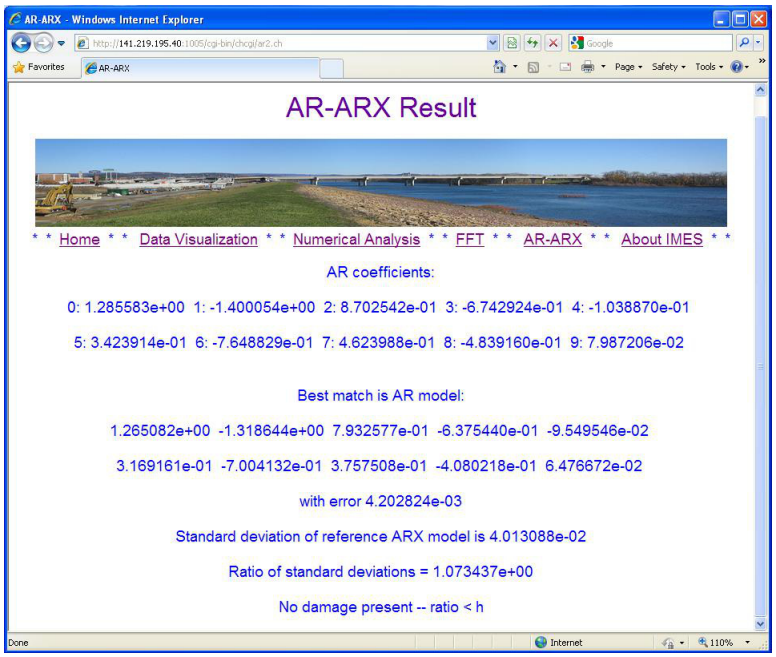


Figure 7.27: AR-ARX result for sensor node 1 in the normal pattern

sensor network provides sensor-level web services. Users can directly access remote sensor data through a web browser without the need of transferring sensor data to a central data

station. The distributed web service removes the data transmission burden and enhances the practical applicability of the system. The remote sensor data are visualized in the users' web browser using web plots. The distributed structural damage diagnosis is achieved through embedded data processing and damage diagnosis algorithms at the sensor nodes. The presented web-based system is validated using a scaled steel bridge. The acceleration data collected by SHM sensor nodes mounted on the steel bridge can be viewed though a web browser in the user's laptop. The web interface can also display the FFT plot and the AR-ARX results of the acceleration data for damage diagnosis.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

This research is motivated by the emerging need of adaptive monitoring and control in distributed systems. The major contributions of this dissertation include following aspects.

Firstly, a number of feature extraction methods: autoregressive model, discrete Fourier transform, and discrete wavelet transform, for structural damage pattern recognition are investigated. The impact of the dissimilarity measures on the performance of feature extraction and pattern recognition is also studied. The test data for evaluating the performance of feature extraction methods and the dissimilarity measures are chosen from the Z24 bridge test. The Z24 bridge test data include the progressive damage data of

the same type but varying levels as well as radically distinct damage modalities. These features of the test data allow us to evaluate the performance of feature extraction methods and dissimilarity measures for different damage modalities and different levels of damage severity. In addition, the feasibility of using pattern recognition approach for damage localization analysis is also explored. The simulation result shows that the closer the sensor nodes to the damage location, the larger the distances of damage feature vectors shift from the normal pattern feature vectors.

Secondly, an agent-based prototype sensor network is developed to support distributed damage diagnosis for structural health monitoring. To achieve required computational capability, a tiny embedded computer, Gumstix, is selected as the computational engine of the sensor nodes. A Wifistix board is used to provide wireless communication capability. The measurements of accelerometers, strain gauges, relative humidity, and temperature sensors are connected to a customized sensor board for real-time data collection. These sensor data are then transmitted to the Gumstix board for damage pattern recognition. The tight integration of the Gumstix with sensor board offers significant reduction of communication cost and the enhancement of data transmission efficiency. To support agent application, an embeddable mobile agent system is integrated with this sensor unit. The validation of this agent-based sensor network shows its effectiveness for the intelligent and adaptive structural health monitoring.

Thirdly, multi-objective optimization algorithms for the control of mobile monitoring

agents in artificial-immune-system-based monitoring networks are studied. The developed algorithms optimize agent generation and distribution to increase damage detection probability, reduce response time, and extend network lifespan. The amount and type of generated monitoring agents are tuned to the detected damage. The selected sensor node for a mobile monitoring agent to visit is based on the affinity between the sensor data feature vector and the memory cells of the monitoring agent. In addition, the remaining battery capacity of a sensor node will impact the selection decision. A sensor node with high remaining battery capacity has high probability to receive a mobile monitoring agent. This selection strategy not only ensures the detection effectiveness, but also balances the remaining battery capacities of sensor nodes across the monitoring network.

Finally, case study is performed to prove the validity of the proposed data driven agent network. The first case illustrates that a mobile agent paradigm applied to structural health monitoring reduces data transmission and enhances the flexibility of the monitoring network. The mobile agent approach distributes damage diagnosis algorithms, such as AR-ARX algorithm, to the sensor nodes instead of transmitting sensor data to a central data station for the damage diagnosis. The data processing and damage diagnosis are performed at sensor nodes. This case shows that the mobile agent approach can successfully deploy damage diagnosis algorithms on distributed sensor nodes via mobile agents. The second case reveals the potential relationship between damage location and the feature vectors of sensor data. The shifted distance between normal pattern features and damaged pattern features varies according to the location of sensors. The result in this case shows the

potential of using pattern recognition techniques for damage localization. In the third case, driving cycle recognition using pattern recognition method is investigated. Four federal driving cycles are used as reference driving cycles. The feature members to represent driving cycles are identified. The simulation results show that these feature members are able to distinguish different real-world driving cycles. The final case studies a web-based structural health monitoring network. In this network, each sensor node can serve as a web server and be accessed by remote users.

8.2 Future work

This dissertation research focused on the feature extraction and pattern recognition for temporal sensor data collected from a single senor unit, the optimal control of agent generation and distribution in a network, and building an agent-based prototype network for the evaluation of designed adaptive monitoring and control strategies. To advance the development of intelligent monitoring and control for the large scale complex systems, following future work is proposed.

(1) Explore adaptive anomaly detection with dynamic model updating

The anomaly detection in existing methods is based on the detection of real-time sensor data deviating from a normal behavior model. The normal behavior model contains a set

of feature vectors extracted from normal data during the training process. The generated normal behavior model is typically used for anomaly detection throughout the entire lifetime of the system. Our initial study shows that the normal behavior model is impacted by the changes of environmental and operational conditions. To enhance the adaptability of pattern recognition algorithms, the dynamic updating of normal behavior model will be necessary in real applications.

(2) Investigate the application of adaptive immune response in distributed monitoring networks

Adaptive immune response provides a good model for the real-time monitoring networks to deal with emerging threat. To mimic adaptive immune response in monitoring networks, mathematical models need to be investigated to study the interactions of immune cells and provide the guideline for the design of agent functions and interaction protocols. These mathematical models should include the major participants involved in the adaptive immune response, such as B cells, T cells, antigens, antibody memory cells, and antigen presenting cells.

(3) Incorporate spatial information for the prediction of system states

The algorithms developed in this work only consider the temporal information from a single location. To analyze and predict the dynamics of the system globally, the incorporation of spatial information is necessary. This change will introduce a number of challenges,

including the increase of transmission load, the delay of system response, and the need of distributing computation in an area.

(4) Perform rigorous validation tests for the developed algorithms and the agent-based network platform

The validation tests in this project were conducted in the simulation and laboratory environments. To verify the effectives of the developed algorithms and the agent-based network platform, a comprehensive validation test in the real-world scenario is needed. Additional applications may be pursued to evaluate the applicability of this adaptive monitoring and control strategy in different areas.

References

[1] H. Verbraken, "Z24 bridge." Website, 1999. `http://www.kuleuven.be/bwm/Z24/index.html`.

[2] B. Peeters and G. D. Roeck, "One-year monitoring of the Z24-bridge: environmental effects versus damage events," *Earthquake Engineering and Structural Dynamics*, vol. 30, pp. 149–171, 2001.

[3] B. Chen, H. H. Cheng, and J. Palen, "Mobile-C: a mobile agent platform for mobile C/C++ agents," *Software: Practice and Experience*, vol. 36, no. 15, pp. 1711–1733, 2006.

[4] H. Sohn and C. R. Farrar, "Damage diagnosis using time series analysis of vibration signals," *Smart Materials and Structures*, vol. 10, pp. 446–451, 2001.

[5] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Academic Press, 2008.

[6] V. Shapiro, G. Gluhchev, and D. Dimov, "Towards a multinational car license plate recognition system," *Machine Vision and Applications*, vol. 17, pp. 173–183, 2006.

[7] A. Amin, "Recognition of hand-printed characters based on structural description and inductive logic programming," in *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pp. 333–337, 2001.

[8] S. Basu, N. Das, R. Sarkar, M. Kundu, M. Nasipuri, and D. K. Basu, "A hierarchical approach to recognition of handwritten bangla characters," *Pattern Recogn.*, vol. 42, pp. 1467–1484, 2009.

[9] W. MacLean and J. Tsotsos, "Fast pattern recognition using normalized grey-scale correlation in a pyramid image representation," *Machine Vision and Applications*, vol. 19, pp. 163–179, 2008.

[10] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Journal of Knowledge and Information Systems*, vol. 3, pp. 263–286, 2000.

[11] M. Siadaty and W. Knaus, "Locating previously unknown patterns in data-mining results: a dual data- and knowledge-mining method," *BMC Medical Informatics and Decision Making*, vol. 6, pp. 1–13, 2006.

[12] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data Min. Knowl. Discov.*, vol. 8, pp. 53–87, 2004.

[13] N. Nair and T. Sreenivas, "Viterbi algorithm for multi-pattern joint decoding," in *TENCON 2009 - 2009 IEEE Region 10 Conference*, pp. 1–5, 2009.

[14] A. Alapetite, "Impact of noise and other factors on speech recognition in anaesthesia," *International Journal of Medical Informatics*, vol. 77, no. 1, pp. 68–77, 2008.

[15] M. Kallergi, "Computer-aided diagnosis of mammographic microcalcification clusters," *Medical Physics*, vol. 31, pp. 314–326, 2004.

[16] O. D. Trier, A. K. Jain, and T. Taxt, "Feature extraction methods for character recognition - a survey," *Pattern Recognition*, vol. 29, no. 4, pp. 641–662, 1996.

[17] R. Nevatia and K. Babu, "Linear feature extraction and description," in *Proceedings of the 6th international joint conference on Artificial intelligence - Volume 2*, pp. 639–641, 1979.

[18] T. R. Reed and J. M. H. du Buf, "A review of recent texture segmentation and feature extraction techniques," *CVGIP: Image Underst.*, vol. 57, pp. 359–372, 1993.

[19] J. Yang, A. F. Frangi, J. Y. Yang, D. Zhang, and Z. Jin, "KPCA plus LDA: a complete kernel fisher discriminant framework for feature extraction and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 2, pp. 230–244, 2005.

[20] B. Manjunath and W. Ma, "Texture features for browsing and retrieval of image data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 837–842, 1996.

[21] T. Gevers and A. Smeulders, "Pictoseek: combining color and shape invariant features for image retrieval," *IEEE Transactions on Image Processing*, vol. 9, no. 1, pp. 102–119, 2000.

[22] C. Tang and H. Hang, "A feature-based robust digital image watermarking scheme," *IEEE Transactions on Signal Processing*, vol. 51, no. 4, pp. 950–959, 2003.

[23] F. Korn, H. V. Jagadish, and C. Faloutsos, "Efficiently supporting ad hoc queries in large datasets of time sequences," *Sigmod Record*, vol. 26, pp. 289–300, 1997.

[24] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient similarity search in sequence databases," in *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, FODO '93, pp. 69–84, 1993.

[25] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," *Sigmod Record*, vol. 23, pp. 419–429, 1994.

[26] K. P. Chan and A. W. Fu, "Efficient time series matching by wavelets," in *Data Engineering, 1999. Proceedings., 15th International Conference on*, pp. 126 –133, 1999.

[27] E. Keogh, K. Chakrabarti, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *Sigmod Record*, vol. 30, pp. 151–162, 2001.

[28] Y. Cai and R. Ng, "Indexing spatio-temporal trajectories with chebyshev

polynomials," in *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pp. 599–610, ACM, 2004.

[29] A. Bagnall, C. A. Ratanamahatana, E. Keogh, S. Lonardi, and G. Janacek, "A bit level representation for time series data mining with shape based similarity," *Data Mining and Knowledge Discovery*, vol. 13, pp. 11–40, 2006.

[30] Z. Wang, S. Chen, J. Liu, and D. Zhang, "Pattern representation in feature extraction and classifier design: Matrix versus vector," *IEEE Transactions on Neural Networks*, vol. 19, no. 5, pp. 758–769, 2008.

[31] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine Learning*, vol. 29, pp. 131–163, 1997.

[32] Q. Wang, G. M. Garrity, J. M. Tiedje, and J. R. Cole, "Naive bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy," *Applied and Environmental Microbiology*, vol. 73, no. 16, pp. 5261–5267, 2007.

[33] J. Quinlan, "Induction of decision trees," *Machine Learning*, pp. 81–106, 1986.

[34] T. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine Learning*, vol. 40, pp. 139–157, 2000.

[35] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[36] C. Hsu and C. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[37] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy K-nearest neighbor algorithm," *IEEE Transactions on Systems Man and Cybernetics*, vol. 15, no. 4, pp. 580–585, 1985.

[38] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 881–892, 2002.

[39] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognition*, vol. 36, pp. 451–461, 2003.

[40] A. A. Alaiya, B. Franzén, A. Hagman, B. Dysvik, U. J. Roblick, S. Becker, B. Moberger, G. Auer, and S. Linder, "Molecular classification of borderline ovarian tumors using hierarchical cluster analysis of protein expression profiles," *International Journal of Cancer*, vol. 98, no. 6, pp. 895–899, 2002.

[41] B. Chen and C. Zang, "Artificial immune pattern recognition for structure damage classification," *Computers and Structures*, vol. 87, no. 21-22, pp. 1394 – 1407, 2009.

[42] N. Jennings and S. Bussmann, "Agent-based control systems: Why are they suited

to engineering complex systems?," *IEEE Control Systems*, vol. 23, no. 3, pp. 61–73, 2003.

[43] H. Wada and S. Okada, "An autonomous agent approach for manufacturing execution control systems," *Integrated Computer-Aided Engineering*, vol. 9, pp. 251–262, 2002.

[44] L. D. Chou, K. C. Shen, K. C. Tang, and C. C. Kao, "Implementation of mobile-agent-based network management systems for national broadband experimental networks in Taiwan," in *Holonic and Multi-Agent Systems for Manufacturing*, vol. 2744 of *Lecture Notes in Computer Science*, p. 1090, Springer Berlin / Heidelberg, 2003.

[45] H. Tu and J. Hsiang, "An architecture and category knowledge for intelligent information retrieval agents," *Decision Support Systems*, vol. 28, no. 3, pp. 255–268, 2000.

[46] T. Sandholm, "eMediator: A next generation electronic commerce server," in *Computational Intelligence*, pp. 73–96, 2002.

[47] S. P. M. Choi, J. Liu, and S. P. Chan, "A genetic agent-based negotiation system," *Computer Networks*, vol. 37, no. 2, pp. 195–204, 2001.

[48] D. B. Lange and O. Mitsuru, *Programming and Deploying Java Mobile Agents Aglets*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1st ed., 1998.

[49] K. Ono and H. Tai, "A security scheme for aglets," *Software Practice and Experience*, vol. 32, pp. 497–514, 2002.

[50] D. Wong, N. Paciorek, T. Walsh, J. DiCelie, M. Young, and B. Peet, "Concordia: An infrastructure for collaborating mobile agents," in *Proceedings of the First International Workshop on Mobile Agents*, pp. 86–97, 1997.

[51] H. Peine, "Application and programming experience with the Ara mobile agent system," *Software Practice and Experience*, vol. 32, pp. 515–541, 2002.

[52] D. Johansen, K. J. Lauvset, R. V. Renesse, F. B. Schneider, N. P. Sudmann, and K. Jacobsen, "A TACOMA retrospective," *Software Practice and Experience*, vol. 32, pp. 605–619, 2002.

[53] H. H. Cheng, "Ch: A C/C++ interpreter for script computing," *C/C++ Users Journal*, vol. 24, pp. 6–12, 2006.

[54] B. Chen, H. Cheng, and J. Palen, "Integrating mobile agent technology with multi-agent systems for distributed traffic detection and management systems," *Transportation Research Part C: Emerging Technologies*, vol. 17, pp. 1–10, 2009.

[55] S. Nestinger and H. Cheng, "Flexibile vision: Mobile agent approach to distributed vision sensor fusion," *IEEE Robotics and Automation Magazine*, vol. 17, no. 3, pp. 66–77, 2010.

[56] S. S. Nestinger, B. Chen, and H. H. Cheng, "A mobile agent-based framework for flexible automation systems," vol. 15, pp. 942–951, 2010.

[57] L. Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, 1 ed., 2002.

[58] S. Hofmeyr and S. Forrest, "Architecture for an artificial immune system," *Evol. Comput.*, vol. 8, pp. 443–473, 2000.

[59] J. Timmis, M. Neal, and J. Hunt, "An artificial immune system for data analysis," *Biosystems*, vol. 55, no. 1-3, pp. 143–150, 2000.

[60] P. K. Harmer, P. D. Williams, G. H. Gunsch, and G. B. Lamont, "An artificial immune system architecture for computer security applications," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 252–280, 2002.

[61] S. Sarafijanovic and J. L. Boudec, "An artificial immune system approach with secondary response for misbehavior detection in mobile ad hoc networks," *IEEE Transactions on Neural Networks*, vol. 16, no. 5, pp. 1076–1087, 2005.

[62] 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," *Proc. VLDB Endow.*, vol. 1, pp. 1542–1552, August 2008.

[63] J. Lin, M. Vlachos, E. Keogh, and D. Gunopulos, "Iterative incremental clustering of time series," in *EDBT'04*, pp. 106–122, 2004.

[64] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proceedings of KDD-94: AAAI Workshop on KnowledgeDiscovery in Databases*, pp. 359–370, 1994.

[65] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems*, vol. 7, pp. 358–386, 2005.

[66] H. Sohn, "Effects of environmental and operational variability on structural health monitoring," *Philosophical Transactions of the Royal Society a-Mathematical Physical and Engineering Sciences*, vol. 365, pp. 539–560, 2007.

[67] N. Saravanan and K. I. Ramachandran, "Fault diagnosis of spur bevel gear box using discrete wavelet features and decision tree classification," *Expert Systems with Applications*, vol. 36, pp. 9564–9573, 2009.

[68] S. Huang and T. Wu, "Integrating ga-based time-scale feature extractions with svms for stock index forecasting," *Expert Systems with Applications*, vol. 35, pp. 2080–2088, 2008.

[69] S. Pittner and S. V. Kamarthi, "Feature extraction from wavelet coefficients for pattern recognition tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 83–88, 1999.

[70] P. K. Sen and J. M. Singer, *Large Sample Methods in Statistics: An Introduction with Applications*. Chapman and Hall, 1994.

[71] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: a tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[72] B. Chen and C. Zang, "A hybrid immune model for unsupervised structural damage pattern recognition," *Expert Systems with Applications*, vol. 38, pp. 1650–1658, 2011.

[73] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Prentice Hall, 1988.

[74] D. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[75] M. Sato-Ilic and L. Jain, "Introduction to fuzzy clustering," in *Innovations in Fuzzy Clustering*, vol. 205 of *Studies in Fuzziness and Soft Computing*, pp. 1–8, 2006.

[76] B. Chen, "XML-based agent communication, migration and computation in mobile agent systems," *Journal of Systems and Software*, vol. 81, pp. 1364–1376, 2008.

[77] D. Kirschner, *The multi-scale immune response to pathogens: M. tuberculosis as an example*. Springer US, 2007.

[78] M. Neal and B. Trapnell, *Go Dutch: Exploit Interactions and Environments with Artificial Immune Systems*. Springer, 2007.

[79] P. Delves, S. Martin, D. Burton, and I. Roitt, *Roitt's Essential Immunology*. Wiley-Blackwell, 2006.

[80] B. Chen and W. Liu, "Mobile agent computing paradigm for building a flexible structural health monitoring sensor network," *Computer-Aided Civil and Infrastructure Engineering*, vol. 25, pp. 504–516, 2010.

[81] S. Hadim and N. Mohamed, "Middleware for wireless sensor networks: a survey," in *First International Conference on Communication System Software and Middleware*, 2006.

[82] Gumstix, "Gumstix," 2009. `http://www.gumstix.com`.

[83] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, 1999.

[84] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes in C : the art of scientific computing*. Cambridge University Press, 1992.

[85] Gumstix, "The foundation for intelligent physical agents," 2009. `http://fipa.org/`.

[86] K. P. Ferentinos and T. A. Tsiligiridis, "Adaptive design optimization of wireless sensor networks using genetic algorithms," *Computer Networks*, vol. 51, pp. 1031–1051, 2007.

[87] R. Khanna, H. Liu, and H. H. Chen, "Self-organization of sensor networks using

genetic algorithms," in *IEEE International Conference on Communications*, vol. 8, pp. 3377 –3382, 2006.

[88] S. Hussain, "Genetic algorithm for hierarchical wireless sensor networks," *Journal of Networks*, vol. 2, pp. 87–97, 2007.

[89] I. V. Maslov and I. Gertner, "Multi-sensor fusion: an evolutionary algorithm approach," *Information Fusion*, vol. 7, pp. 304–330, 2006.

[90] A. P. Bhondekar, R. Vig, M. L. Singla, C. Ghanshyam, and P. Kapur, "Genetic algorithm based node placement methodology for wireless sensor networks," in *International Multi Conference of Engineers and Computer Scientists*, vol. 1, 2009.

[91] Q. Qiu, Q. Wu, D. Burns, and D. Holzhauer, "Lifetime aware resource management for sensor network using distributed genetic algorithm," in *Proceedings of the 2006 international symposium on Low power electronics and design*, ISLPED '06, pp. 191–196, ACM, 2006.

[92] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, 2001.

[93] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[94] R. Peng and M. Pedram, "An analytical model for predicting the remaining

battery capacity of lithium-ion batteries," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, pp. 441–451, 2006.

[95] A. J. Chipperfield and P. J. Fleming, "The matlab genetic algorithm toolbox," in *IEEE Colloquium on Applied Control Techniques Using MATLAB*, pp. 10/1 –10/4, 1995.

[96] L. Garibaldi, S. Marchesiello, and E. Bonisoli, "Identification and up-dating over the Z24 benchmark," *Mechanical Systems and Signal Processing*, vol. 17, pp. 153–161, 2003.

[97] A. Fasana, L. Garibaldi, E. Giorcelli, and D. Sabia, *Z24 bridge dynamic data analysis by time domain method*, vol. 1, pp. 852–856. Society of Experimental Mechanics (SEM), 2001.

[98] D. J. Luscher, J. M. W. Brownjohn, H. Sohn, and C. R. Farrar, *Modal parameter extraction of Z24 bridge data*, vol. 1, pp. 836–841. 2001.

[99] S. Marchesiello, B. A. D. Piomboand, and S. Sorrentino, *Application of the CVA-BR method to the Z24 bridge vibration data*, vol. 1, pp. 864–869. Society of Experimental Mechanics (SEM), 2001.

[100] K. Womack and J. Hodson, *System identification of the Z24 Swiss bridge*, vol. 1, pp. 842–845. Society of Experimental Mechanics (SEM), 2001.

[101] L. Mevel, M. Goursat, and M. Basseville, "Stochastic subspace-based structural

identification and damage detection and localisation-application to the Z24 bridge benchmark," *Mechanical Systems and Signal Processing*, vol. 17, pp. 143–151, 2003.

[102] R. A. Swartz and J. P. Lynch, "Damage characterization of the Z24 bridge by transfer function pole migration," in *Proceedings of the 26th International Modal Analysis Conference (IMAC) XXVI*, 2008.

[103] C. Kramer and W. Zhang, "Z24 bridge damage detection tests," in *Proceedings of the International Modal Analysis Conference IMAC*, vol. 2, pp. 1023–1029, 1999.

[104] J. Kullaa, "Stochastic subspace-based structural identification and damage detection and localisation-application to the Z24 bridge benchmark," *Mechanical Systems and Signal Processing*, vol. 17, pp. 163–170, 2003.

[105] J. Maeck and G. D. Roeck, "Damage assessment using vibration analysis on the Z24-bridge," *Mechanical Systems and Signal Processing*, vol. 17, pp. 133–142, 2003.

[106] A. Teughels and G. D. Roeck, "Structural damage identification of the highway bridge Z24 by FE model updating," *Journal of Sound and Vibration*, vol. 278, pp. 589–610, 2004.

[107] G. D. Roeck, B. Peeters, and J. Maeck, "Dynamic monitoring of civil engineering structures," in *Fourth International Colloquium on Computation of Shell and Spatial Structures*, pp. 1–24, 2000.

[108] X. He, B. Moaveni, J. Conte, A. Elgamal, and S. Masri, "Modal identification study of vincent thomas bridge using simulated wind-induced ambient vibration data," *Computer-Aided Civil and Infrastructure Engineering*, vol. 23, no. 5, pp. 373–388, 2008.

[109] S. Li and Z. Wu, "A model-free method for damage locating and quantifying in a beam-like structure based on dynamic distributed strain measurements," *Computer-Aided Civil and Infrastructure Engineering*, vol. 23, pp. 404–413, 2008.

[110] B. Moaveni, X. He, J. P. Conte, and R. A. D. Callafon, "Damage identification of a composite beam using finite element model updating," *Computer-Aided Civil and Infrastructure Engineering*, vol. 23, no. 5, pp. 339–359, 2008.

[111] P. A. Psimoulis and S. C. Stiros, "Experimental assessment of the accuracy of gps and rts for the determination of the parameters of oscillation of major structures," *Computer-Aided Civil and Infrastructure Engineering*, vol. 23, pp. 389–403, 2008.

[112] H. Sohn, S. D. Kim, and K. Harries, "Reference-free damage classification based on cluster analysis," *Computer-Aided Civil and Infrastructure Engineering*, vol. 23, no. 5, pp. 324–338, 2008.

[113] P. Kolakowski, "Structural health monitoring - a review with the emphasis on low-frequency methods," *Engineering Transactions*, vol. 55, pp. 239–275, 2007.

[114] T. Nagayama, B. F. Spencer, and J. A. Rice, "Autonomous decentralized structural

health monitoring using smart sensors," *Structural Control and Health Monitoring*, vol. 16, pp. 842–859, 2009.

[115] K. Römer, "Programming paradigms and middleware for sensor networks," in *GI/ITG Workshop on Sensor Networks*, pp. 49–54, 2004.

[116] Y. Gao, B. F. Spencer, and M. Ruiz-Sandoval, "Distributed computing strategy for structural health monitoring," *Structural Control and Health Monitoring*, vol. 13, pp. 488–507, 2006.

[117] J. Blumenthal, M. Handy, F. Golatowski, M. Haase, and D. Timmermann, "Wireless sensor networks - new challenges in software engineering," in *IEEE Conference on Emerging Technologies and Factory Automation*, vol. 1, pp. 551–556, 2003.

[118] A. Boulis, C. Han, and M. B. Srivastava, "Design and implementation of a framework for efficient and programmable sensor networks," in *Proceedings of the 1st international conference on Mobile systems*, pp. 187–200, ACM Press, 2003.

[119] J. Elson, L. Girods, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 147–163, 2002.

[120] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys '03, pp. 138–149, 2003.

[121] J. van Greunen and J. Rabaey, "Lightweight time synchronization for sensor

networks," in *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, pp. 11–19, 2003.

[122] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, (New York, NY, USA), pp. 13–24, ACM, 2004.

[123] R. Beta, Y. Yacoub, W. Wang, D. Lyons, M. Gambino, and G. Rideout, "Heavy duty testing cycles: Survey and comparison," 1994.

[124] E. Ericsson, "Independent driving pattern factors and their influence on fuel-use and exhaust emission factors," *Transportation Research Part D: Transport and Environment*, vol. 6, no. 5, pp. 325–345, 2001.

[125] S. I. Jeon, S. T. Jo, Y. I. Park, and J. M. Lee, "Multi-mode driving control of a parallel hybrid electric vehicle using driving pattern recognition," *Journal of dynamic systems, measurement, and control*, vol. 124(1), pp. 141–149, 2002.

[126] U. S. E. P. Agency, "Testing and measuring emissions," 2010. `http://www.epa.gov/nvfel/testing/dynamometer.htm`.

[127] R. S. Chen, C. C. Chen, and C. C. Chang, "A web-based erp data mining system for decision making," *International Journal of Computer Applications in Technology*, vol. 17, no. 3, pp. 156–169, 2003.

[128] M. A. A. Hasin, P. Natavudh, and M. A. Sharif, "A web-based quality management system and its implementation in a computer assembly industry," *International Journal of Computer Applications in Technology*, vol. 17, no. 4, pp. 202–212, 2003.

[129] A. P. Chassiakos and S. P. Sakellaropoulos, "A web-based system for managing construction information," *Adv. Eng. Softw.*, vol. 39, pp. 865–876, 2008.

[130] Q. Yu, B. Chen, and H. H. Cheng, "Web based control system design and analysis," *IEEE Control Systems*, vol. 24, no. 3, pp. 45–57, 2004.

[131] U. Topp, P. Müller, J. Konnertz, and A. Pick, "Web based service for embedded devices," in *Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Services, and Database Systems*, pp. 141–153, 2003.

[132] BOA, "Introduction to boa," 2011. `http://www.boa.org/documentation/boa-1.html.2011`.

[133] N. Kakanakov and G. Spasov, "Adaptation of web service architecture in distributed embedded systems," in *Proceedings of the International Conference on Computer Systems and Technologies*, 2005.

[134] CGI, "CGI: Common gateway interface," 2011. `http://www.w3.org/CGI/.2011`.

[135] Softintegration, "The Ch language environment CGI toolkit," 2011.