THESIS ~~[illegible]~~
3
2010

This is to certify that the
dissertation entitled

AUTOMATIC DELAMINATION DETECTION OF CONCRETE
BRIDGE DECKS USING ACOUSTIC SIGNATURES

presented by

Gang Zhang

has been accepted towards fulfillment
of the requirements for the

Doctoral	degree in	Civil Engineering

[illegible]
Major Professor's Signature

6/22/2010
Date

AUTOMATIC DELAMINATION DETECTION OF CONCRETE BRIDGE DECKS USING ACOUSTIC SIGNATURES

By

Gang Zhang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Civil Engineering

2010

ABSTRACT

AUTOMATIC DELAMINATION DETECTION OF CONCRETE BRIDGE DECKS USING ACOUSTIC SIGNATURES

By

Gang Zhang

Delamination of the concrete cover above the upper reinforcing bars is a common problem in concrete bridge decks. The delamination is typically initiated by corrosion of the upper reinforcing bars and promoted by freeze-thaw cycling and traffic loading. The detection of delamination is important for bridge maintenance and acoustic non-destructive evaluation (NDE) is widely used due to its low cost, speed, and easy implementation. In traditional acoustic approaches, the inspector sounds the surface of the deck by impacting it with a hammer or bar, or by dragging a chain, and assesses delamination by the "hollowness" of the sound. The acoustic signals are often contaminated by traffic and ambient noise at the site and the detection is highly subjective.

The performance of acoustic NDE methods can be improved by employing a suitable noise-cancelling algorithm and a reliable detection algorithm that eliminates subjectivity. Since the noise is non-stationary and unpredictable, the algorithms should be adaptive. After evaluating different noise cancelling algorithms based on a numerical performance criterion and through visual inspection, a noise cancelling algorithm using a modified independent component analysis (ICA) was used to separate the sounding signals from recordings in a noisy environment. After the noise signals and the impact signals were successfully separated, the features of filtered signal were extracted. Different feature extraction algorithms were used to extract features of the filtered signals. The performance of different feature extraction algorithms were evaluated against repeatability, separability and mutual information which measures the information about the condition of the concrete bridge deck. Mel-frequency cepstral coefficients (MFCC) were used as features for detection. The extracted features were further selected based on the value of the mutual information to reduce the negative effect of features with poor separability. The features selected were used to train the classifiers and the trained classifiers were used to classify new signals. The error rate was used to evaluate the performance of different classifiers. The radial basis function neural network had the lowest error rate and was selected as the classifier for field application.

The proposed noise-cancelling and delamination detection algorithms were then implemented using mixed-language programming in MATLAB, LabVIEW and C/C++. The performance of the system was verified using both experimental and field data. The proposed system showed good noise robustness. The performance of the system was satisfactory when there was sufficient available data for training and the selection of the training data was representative.

ACKNOWLEDGMENTS

First and foremost, I wish to thank my advisor, Dr. Ronald S. Harichandran, for the advice, support and guidance during the course of this research. His assistance and trust were fundamental to the achievement of the project. His understanding attitude, encouragement and the time spent in amiable conversations made my Ph.D study very enjoyable. He is not only the academic advisor for my Ph.D study, but also a mentor and a good friend for the rest of my life.

I wish to extend my gratitude to my current committee members – Dr. Rigoberto Burgueño, Dr. Jung-Wuk Hong and Dr. Lalita Udpa, and former committee members – Dr. Pradeep Ramuhalli and Dr. Niell G. Elvin for their invaluable suggestions, comments, and feedback. Dr. Ramuhalli's guidance was especially valuable during the development of the algorithms. I would like to thank them for the time and care they have taken to guide this work and the encouragement they gave me during this critical period of my life.

I gratefully acknowledge the financial aid from Michigan Department of Transportation and the suggestions and the help they provided. Special thanks are due to Sia Ravanbakhsh and James C. Brenton for their help in the experiments.

My parents have always supported me in many ways and I wish to thank them infinitely, for everything. Last, but not least, I thank my wife, Huijun Dong, who has supported my decision to fulfill my academic goals and has shared with me the same understanding of leading a prosperous life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

With the development of civilization and modernization, millions of bridges have been built to accommodate growing needs for safe and efficient transportation. Bridges account for a substantial component of the highway infrastructure. Due to the advantages of being durable, versatile and economic, concrete bridges are of great interest to transportation agencies. According to the National Concrete Bridge Council (NCBC), among the 475,000 bridges in the US, concrete bridges consistently outperform bridges made from other materials by a wide margin and more than 70 percent of the bridges built today are made of concrete [1].

Like the human body, structures have their own life cycles. After years of usage, the aging of structures is inevitable and has now become one of the most severe problems facing the infrastructure in the United States. According to the American Society of Civil Engineers (ASCE), more than 26% of the America's bridges were either structurally deficient or functionally obsolete and an annually investment of $17 billion was needed to improve the current bridge conditions [2]. The priority has shifted from building new structures to inspection, assessment and maintenance of existing structures [3]. As the designers of these structures, civil engineers are required not only know how to design structures with sufficient strength at least cost, but also to understand that maintenance and rehabilitation of the structures is as important as, if not more important than, the design of the structure because failure of bridges while in service can result in expensive replacement costs and user delays. Therefore, it is of vital importance to detect damages and defects and have them repaired before they progress and lead to structural or functional failures.

Reinforced concrete bridge decks are continuously degraded due to normal traffic and environmental exposure. This degradation is exacerbated in climatic regions where de-icing chlorides are used and in coastal regions where bridges are exposed to high salt air concentrations. Delamination is the major form of deck distress. This type of damage usually initiates underneath the surface due to corrosion of the steel reinforcement and freeze-thaw of the water in the cracks, and cannot be easily detected by visual inspection in its early stage. With time, the delamination propagates and leads to spalling of the bridge deck. Small delaminated areas can be repaired by patching the affected area. A very large area of delamination will usually result in the replacement of the entire deck, which is expensive and causes significant user delay. It is therefore necessary to detect delamination at an early stage to reduce the cost of repair.

In order to detect delamination and evaluate the condition of a bridge deck such that appropriate repair or rehabilitation measures can be taken, an effective non-destructive evaluation technique that can provide information about the damage location and damage type is needed.

1.2 Problem Statement

Many methods have been considered for the inspection of bridge deck systems including impact echo, ultrasonic pulse velocity, ground penetrating radar, infrared thermography, X-ray imaging and sounding methods. Although many of these methods have been successfully used in detecting delamination and other defects in bridge decks, the different methods have their own advantages and limitations. A detailed comparison between these techniques is described in Chapter 2.

Sounding methods have the advantages of being fast, simple and inexpensive when compared with other more sophisticated techniques. However traditional sounding methods have several problems. First, the detection is subjective and operator dependent. Second, the effectiveness of the method is affected by the level of ambient noise. Although several attempts have been made to improve the performance of traditional sounding methods, only modest improvements have been made and the results are still not satisfactory. It is therefore desirable to develop an improved acoustic method that can be used directly by bridge inspectors for bridge deck inspection. An effective method should be able to eliminate the ambient noise, not require the subjective judgment of a well-trained operator. The method should also be fast, automatic and robust.

1.3 Research Objectives

The major task of this research is to develop an automated inspection system to accurately detect the delamination in concrete bridge deck. This can be achieved by accomplishing the following objectives:

1. Develop a noise cancelling algorithm that can cancel or separate the ambient noise from field measurements. There are different noise cancelling algorithms and each algorithm has its own range of application. A criterion is needed to evaluate the performance of different algorithms so that the optimal algorithm can be selected.

2. Develop algorithms that can differentiate between soundings on the solid concrete and those on a delaminated concrete. These algorithms must be robust and fast.

3. Develop an automatic delamination detection system so that the operator can perform the detection on-site. The system must be easy to use, not require extensive set-up, and be fast so that lane closure durations are minimized

1.4 Organization of the Dissertation

This dissertation is organized into seven chapters.

In Chapter 1, an introduction of the motivation of the research on delamination detection, the problems to be solved and research objectives were described.

In Chapter 2, a literatures review on the type of damage in concrete bridge decks and on the techniques available to detect these defects and damages was provided. A comparison of different non-destructive evaluation (NDE) based on previous research was also included.

In Chapter 3, the performance of different noise cancelling algorithms were compared and evaluated. Numerical criteria were first developed for the comparisons. Detailed theoretical background and derivation about each algorithm were included and their performance was evaluated. The most efficient noise cancelling algorithm based on a modified independent component analysis (ICA) was selected to separate the ambient noise from the recordings.

In Chapter 4, the problem of dimension reduction to facilitate delamination detection was investigated. Different models were used to extract features of the signals. Mel-frequency cepstral coefficients (MFCCs) were selected as the best features for delamination detection.

In Chapter 5, an algorithm was developed for delamination detection. This task was formulated as a classification problem. First, the theoretical background of different classifiers was described. The performance of different classifiers was compared and evaluated and the best classifier was selected based on weighted rank and the error rate of test samples.

In Chapter 6, the development and verification of the hardware and software components of the automatic delamination detection system was described. Various components in the software for collecting and processing the data and detecting delamination were described. The inspection and training process used was also described. Data from two full-scale concrete slab conctructed in the laboratory were used to test the performance of system under different noise levels and with the existence of different repair overlays. Field data from two bridges were used to verify the performance of the system in field conditions. A brief discussion on how to select the training set to improve the performance was also included.

In Chapter 7, a summary of the findings of this research and recommendation for further research were provided.

CHAPTER 2

LITERATURE REVIEW

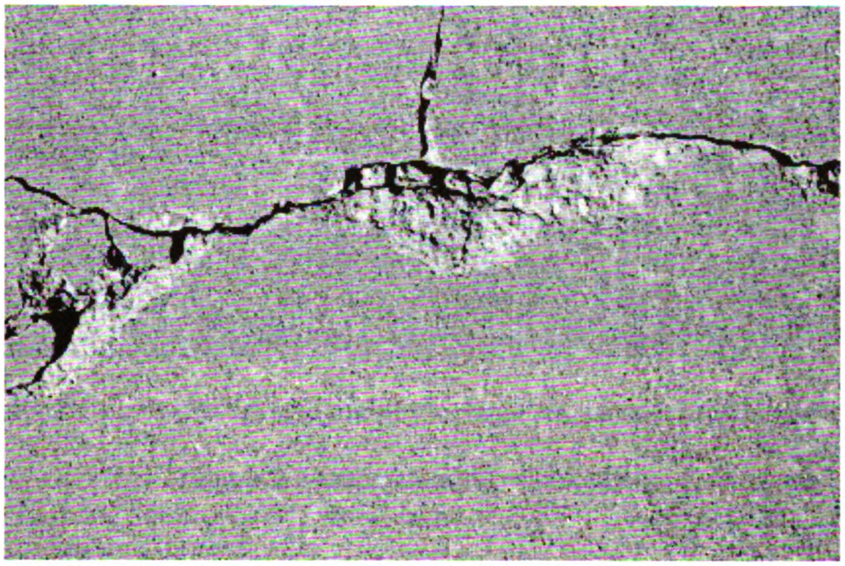
Various types of defects and damage may be caused in concrete structures due to constructional or environmental factors. The presence of damage in concrete may significantly reduce the strength, service life and the integrity of structures. Detecting concrete damage at an early stage can reduce maintenance costs and prevent potentially prevent structural collapse. In order to have a better understanding of the types of damage in concrete and the methods to detect them, this chapter briefly describes the common defects in concrete first and then provides a review of existing non-destructive evaluation methods.

2.1 Damage in Concrete

Cracks, voids and delaminations are considered the most commonly observed damage in concrete, especially in concrete bridges [4]. This section mainly describes the cause of damage as well as the effect of the damage on structures.

2.1.1 Crack

Cracks are defined as the break of concrete causing a discontinuity but not complete separation of the structure [5]. Cracks are the most commonly observed type of damage in concrete because of the low tensile strength of concrete. Figure 2.1 shows a typical crack in concrete. Cracks can be caused by the concrete shrinkage, freeze-thaw cycle, chemical reactions inside the concrete (such as alkali-silica reaction (ASR)) as well as loading. The presence of cracks may affect the performance of concrete structures. Cracks open a path for water to penetrate and accelerate damage due to freeze-thaw cycling in cold regions. When de-icing salts are used or in marine environments, cracks enable the rapid ingression of chloride that accelerates corrosion of the steel reinforcement which leads to expansion and further opening of cracks. Continuously developing cracks may even affect the integrity of the entire structural system. Visual inspection can easily detect large and visible cracks. Other sophisticated methods such as impact echo [6] and ultrasonic pulse velocity [7] may be used to detect finer cracks.

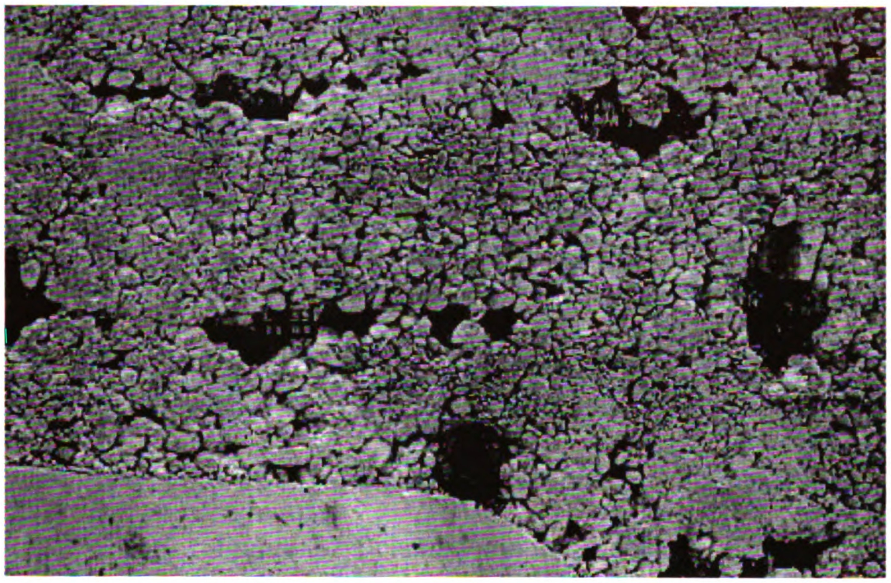


(http://www.silverspringsconcrete.com/concrete-question/)

Figure 2.1 Cracks in Concrete

2.1.2 Honeycombing

Honeycombing refers to the small holes inside concrete caused by poorly graded concrete mixes or by insufficient consolidation during construction [5]. Concrete with honeycombing will often not have enough strength. The presence of the honeycombing increases the permeability of concrete and makes it susceptible to freeze-thaw damage and other environmental attacks. The reinforcement in concrete with honeycombing are also more exposed to corrosive agents from the outside thereby leading to greater corrosion. All these effects will greatly reduce the durability of concrete structures with honeycombing. Commonly used NDE techniques for detecting honeycombing includes the impact echo [8] and ground penetrating radar (GPR) [9].

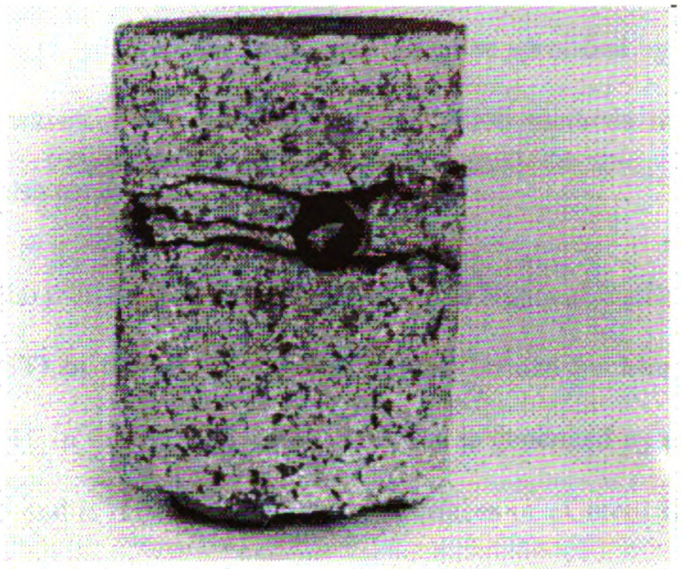


(http://www.concrete.org/Troubleshooting/afmviewfaq.asp?faqid=63)

Figure 2.2 Honeycombing in Concrete

2.1.3 Delamination

Delamination is a layered separation of concrete from the main concrete body. The separation usually occurs at or just above the level of reinforcement as shown in Figure 2.3. This type of damage is usually caused by corrosion of the steel reinforcement [10], a high amount of moisture, and the presence of cracks in the concrete. The progress of the delamination leads to open spalling of the concrete and eventually affects the functional performance of the structure. Concrete delamination impairs not only the appearance of the structure but also its serviceability, and incurs costly repairs if it is not detected in the early stages. Techniques used for detecting delamination include sounding method [11], impact echo [12], and GPR [13].



(http://www.fhwa.dot.gov/pavement/pccp/pubs/04150/chapt3.cfm)

Figure 2.3 Delamination in Concrete

2.2 Non-Destructive Evaluation Methods for Concrete

The last section described three major types of damage in concrete and their effects on the safety and serviceability of structures. The damage needs to be detected early to prevent it from propagating and causing greater loss. Researchers have developed many different non-destructive methods to detecting concrete damage. This section summarizes the more commonly used non-destructive evaluation (NDE) methods for concrete damage detection.

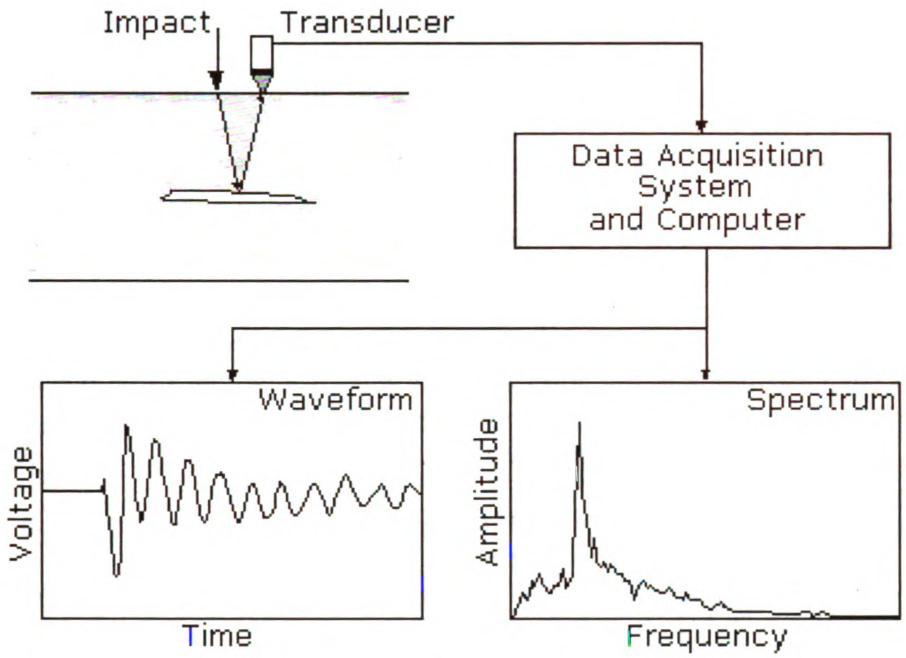
2.2.1 Impact Echo and Impulse Response

When the concrete is impacted, a stress wave is generated and propagates inside the body of the concrete. The presence of damages or defects changes the propagation path of the stress wave reflections. The damage can then be identified by measuring and analyzing the stress waves. There are two dominant NDE methods in this category: impact echo and impulse response.

The impact echo method was first developed at the National Institute of Standards and Technology (NIST) in the 1980s and then further refined by Mary Sansalone at Cornell University [14]. In this method, the stress wave is generated by a short duration impact on the surface and is reflected by internal interfaces or external boundaries. The surface displacement is measured by a transducer and analyzed in the frequency domain. The principle of the method is shown in Figure 2.4. The distance between the receiving surface and the reflecting surface can be calculated as:

$$D = \beta \frac{C_p}{2f} \tag{2.1}$$

where β is a factor related to the shape of the cross-section [14], C_p is the velocity of the P wave in the concrete and f is the peak frequency obtained through frequency domain analysis (for example, fast Fourier transform (FFT)) of the signal.



(www.impact-echo.com/Impact-Echo/impact.htm)

Figure 2.4 Principle of the Impact Echo Method

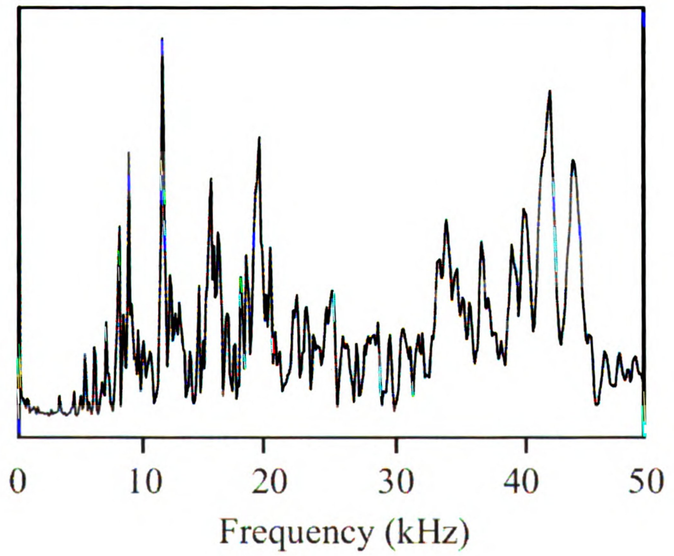


Figure 2.5 Example Spectrum from Impact Echo Tests [15]

The impact echo method can not only detect the presence of the defects, but it also can find the location (depth) of the defects. This method can determine the thickness of the slab and the depth of the defect and needs only one transducer to carry out the test.

However, the signal obtained from the impact-echo test in real situations can be difficult to interpret. When the surface of the defect is irregular, the reflection and scattering of the stress wave become very complex. Multiple peaks appear in the frequency domain and it is difficult to identify the peak associated with the defect as shown in Figure 2.5 [15]. Even though, some algorithms such as SIBIE [15] have been proposed to identify the peak corresponding to the defects, they requires prior information about the properties and the size of the testing samples which is usually not available for field tests. Also, the method is not sensitive to very shallow defects [5]. There are two reasons for this. First, the frequency of the peak for shallow defects can be very high (the frequency for a 1 inch delamination will be as high as 80 kHz according to Equation (2.1) and the peak can be difficult to detect. Second, the frequency corresponding to the bending mode of the shallow defects will produce false peaks in the frequency domain which may lead to faulty results. The impact echo is not sensitive to those defects that are parallel to the direction of stress propagation [16]. Finally, the sensors have to be coupled with the concrete surface to obtain good measurement and the coupling process is time and labor consuming when inspecting a large area such as a bridge deck.

To increase the efficiency of the traditional impact echo method, Zhu [17] proposed a non-contact impact echo method using air-coupled sensors. Instead of using contact sensors such as accelerometers, this method uses air-coupled sensor microphones to measure the response. This method was reported to be successful in detecting the presence and locations of delamination and voids in concrete structures. However, the method requires the microphone to be highly directional to record sound in a very limited range. Also the microphone has to be very close to the surface to be able to pickup the surface response. The analysis of the signal also can be difficult due to the air-coupling effects.

The impulse response (IR) technique [18] is another NDE method that uses the stress wave generated by an impact on the surface of the concrete. In IR, an impact hammer is used to generate the stress wave with the impacting force measured by a built-in load cell. The response, usually the velocity, of the concrete to the impact is also measured. The transfer function between the impact force and the response can then be computed, from which certain parameters such as dynamic stiffness, mobility and damping can be measured. The integrity of the concrete can then be estimated from the calculated parameters. This method has the same disadvantages as the impact echo as full coupling of the sensors to the ground is needed.

2.2.2 Ultrasonic Methods

Ultrasonic methods also use the speed of waves inside concrete. The difference between the impact methods and ultrasonic methods is that the latter uses high frequency (usually greater than 20 kHz) sonic wave as the excitation method while the impact employs a stress wave resulting from mechanical impacts. One of the commonly used ultrasonic methods is the ultrasonic pulse velocity (UPV). In this method, two transducers are needed: one is used to send and one to receive the ultrasonic wave. By measuring the arrival time of the signals, the propagation speed of the ultrasonic wave in concrete can be calculated. The test equipment used is shown in Figure 2.6 [19].

The speed of the P-wave in a solid is:

$$C_p = \sqrt{\frac{\lambda + 2\mu}{\rho}} = \sqrt{\frac{E(1-\nu)}{\rho(1-2\nu)(1+\nu)}} \tag{2.2}$$

where λ and μ are Lame's constants, E and ρ are Young's Modulus and density of the solid, and ν is Poisson's ratio.

As the equation shows, the speed is determined by the density and Young's Modulus of the concrete. The defects in a concrete such as crack or delamination are usually of different densities from that of concrete and will lead to a change in the measured pulse velocity. For example, the diffraction of a wave pulse around an air void will cause an increase in the time of propagation and the measured velocity will decrease. By determining the P-wave speed, the uniformity of concrete can be determined. If multiple sensors are used, the 3D image of the internal defect may be obtained through tomography and the synthetic aperture focusing technique (SAFT) as shown in Figure 2.7 [20].

However, there are several problems associated with this method. First, the transducers have to be coupled to the concrete surface usually by a couplant to ensure that there is no air gap between the surface and the transducer. This will greatly be time consuming if the inspection a large area such as a bridge deck is needed. Second, the accuracy of the method can be affected by other factors such as the temperature and moisture content of the concrete. Third, it might be difficult to use this method on asphalt coated concrete surfaces due to the difference in mechanical properties and the rough texture of asphalt layers [21].

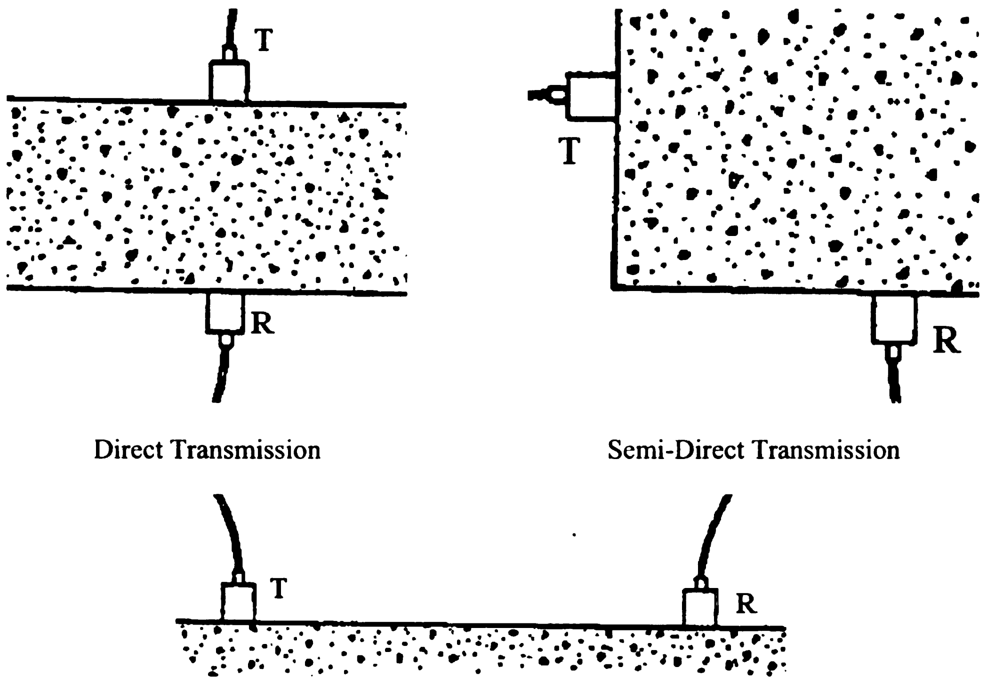


Figure 2.6 Ultrasonic Pulse Velocity Method [19]

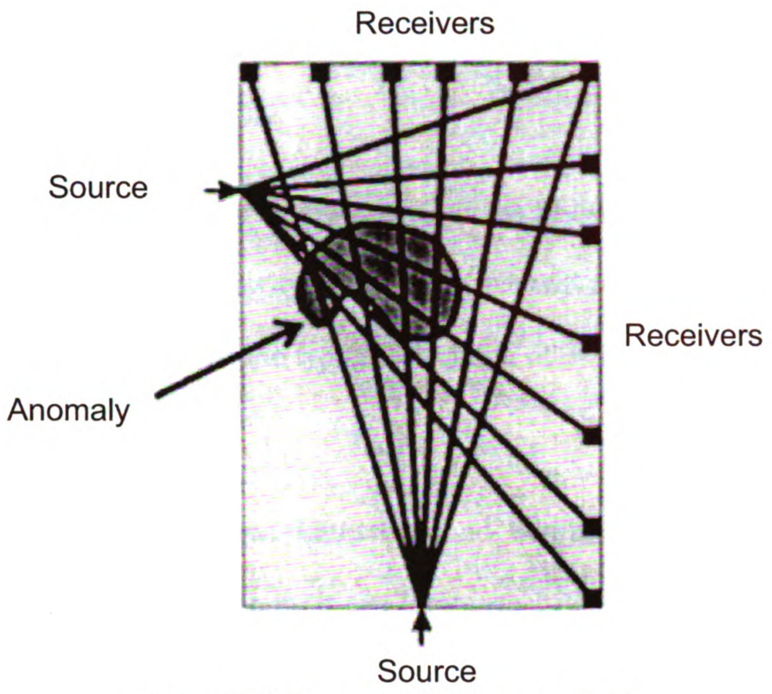


Figure 2.7 Ultrasonic Tomography [20]

2.2.3 Ground Penetrating Radar

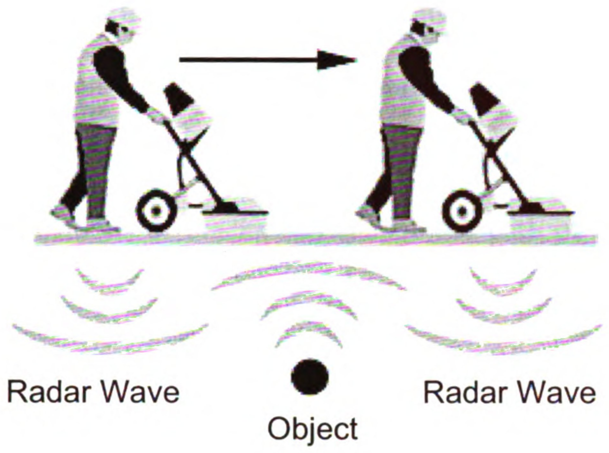
The two methods described above are both contact methods requiring that the sensors be fully coupled with the surface to obtained good measurement which is time consuming if a large area needs to be inspected. Ground penetrating radar (GPR) is a non-contact method. It uses the interaction between the electro-magnetic (EM) wave and boundaries of materials with different electronic properties. The EM wave will be reflected and backscattered if there's a boundary. The reflected wave is captured by the antenna, as shown in Figure 2.8. The amplitude of the reflected wave is dependent on the relative dielectric constant between the two materials and can be calculated as [22]:

$$\rho = \frac{\sqrt{\varepsilon_{r1}} - \sqrt{\varepsilon_{r2}}}{\sqrt{\varepsilon_{r1}} + \sqrt{\varepsilon_{r2}}} \tag{2.3}$$

where ρ is the reflection coefficient, and ε_{r1} and ε_{r2} are the dielectric constants of the materials at the interface.

If the difference across the interface is large, the EM wave will be reflected back; if the difference is small, the majority of the EM wave will pass through. By measuring the energy of the reflected wave, the type and location of defects inside the concrete are detected.

The GPR approach is non-destructive and non-invasive and results can be displayed in real time as a radiogram. It can locate steel reinforcement and damaged or deteriorated areas inside the concrete. Also, the equipment can be carried on a car/truck and can rapidly scan large areas. However, the method has its own disadvantages. The results are presented in the form of a B-scan or C-scan (shown in Figure 2.9 [9]) and require professional knowledge for interpretation. The performance may be affected by many variables including material type, moistures etc. Also, there is a trade-off between penetration and resolution due to limits on the antenna selection. The radar cannot detect objects smaller than the wavelength. In order to increase the resolution, the frequency of the radar wave needs to be high. However, the increase in frequency leads to a reduction in the penetration capacity. Another problem associated with GPR is the inability to detect voids and cracks filled with air because the contrast between the dielectric constants of air and concrete is small.



(http://www.worksmartinc.net)

Figure 2.8Ground Penetrating Radar

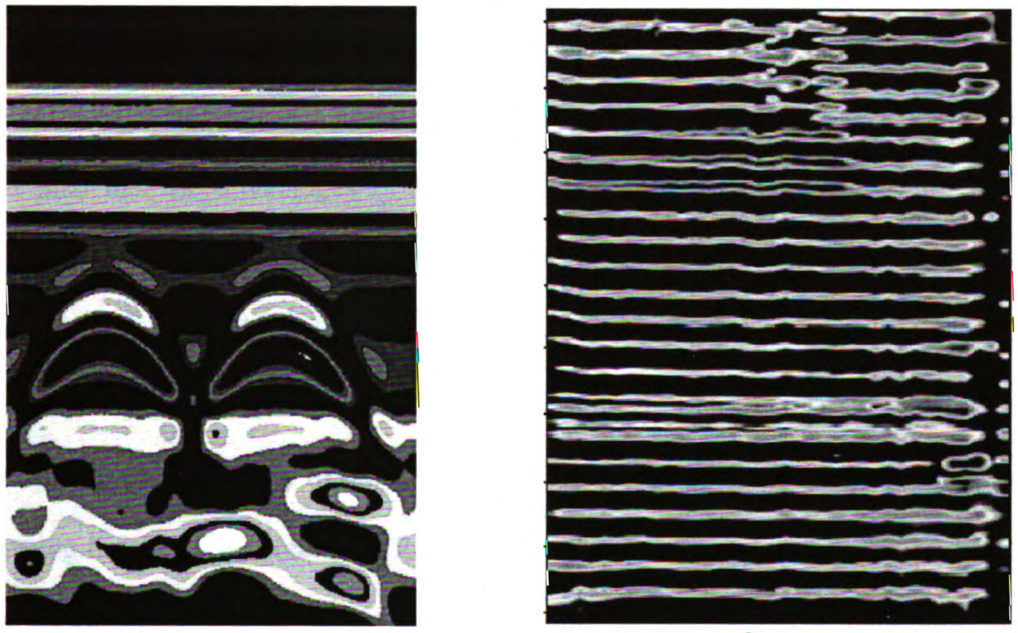


Figure 2.9 Results from Ground Penetrating Radar [9]

2.2.4 Infrared Thermography

Infrared thermography detects sub-surface defects from the distribution of the temperature field. The heat transfer process is dependent on material properties like thermal conductivity, heat capacity and density. The heat transfer is even only when the material is homogeneous. If there are anomalies inside the materials, the heat flow and the temperature distribution will change in these areas. Defects near the surface can be detected by observing the variations of surface temperature. The presence of air-filled defects such as delaminations or cracks can change the path of the heat flow and can therefore be detected using this method [23]. The surface temperature distribution is obtained indirectly by measuring the infrared radiation with an infrared camera. The relationship between the infrared radiation R and the surface temperature T is:

$$R = e\sigma T^4 \tag{2.4}$$

where e is the emissivity of the surface and σ is the Boltzmann constant.

Infrared technology is a non-contact and non-invasive way to detect defects. This method also has the advantages of being portable and fast. The results are usually displayed in the form of a thermograph (shown in Figure 2.10), which can be readily understood. This method has been used to detect defects in civil structures [24]. Since this method uses infrared radiation to measure the temperature, the results are affected by the variance in the emissivity of the surface, for example, surface moisture, patched areas and varying finishes [25]. Heat flow is needed for this technique to work and an object in thermal equilibrium will not provide useful information. Therefore, heat sources are needed. Depending on the type of heat sources, this method can be divided into two categories: passive investigation and active investigation. In passive investigation, natural heat sources such as the sun are used. In active investigation, active heat sources such as infrared radiators are used. For large structures, it takes a very long time and much energy to create sufficient heat flow in the structure. In practice, most of the field tests are conducted using passive heat sources. The tests are performed in the morning when the sun starts to heat the structure or after sunset when the heat in the structure starts to radiate into the environment. This makes the performance of the method weather-dependent. Active investigation is usually applicable only to small specimens or for localized testing. Also, the infrared camera can only measure the temperature close to the surface; the deep defects will not be reflected in the surface temperature. Therefore this method is insensitive to deep damage. Lastly, the high cost of the infrared camera is another limiting factor for this technique.

Figure 2.10 Example Image from an Infrared Camera [25]

2.2.5 X-ray Imaging

As mentioned in Section 2.2.3, ground penetrating radar has the problem of penetration. X-rays however, can easily penetrate into concrete. When the X-ray beam passes through a material, the energy is absorbed or scattered. The amount of energy absorbed or scattered is a function of the mass density of the components, and materials with higher mass density absorb or scatter a greater amount of energy. A collector placed behind the specimen receives the scattered signals as shown in Figure 2.11 [21]. In the images obtained, the high density materials are represented by light areas and low density materials are shown as dark areas. Common types of concrete defects are air-filled voids or cracks that have a clear contrast in mass densities. Therefore, these defects can be easily detected through the X-ray imaging. X-ray imaging can provide clear pictures of the internal structure of the specimens and the presence and locations of the defects can be identified with high accuracy if the energy of the X-ray is properly adjusted. There are limitations associated with this method. First, the cost of the equipment is usually very high. Second, there are safety concerns on the use of radiation. Third, two sides of the specimen need to be accessible. Also, X-ray images are not sensitive to defects that are parallel to the radiation direction and can only detect damage perpendicular to the radiation beams.

Traditional X-ray images can only provide an average density contrast in 2-dimensions and information about defects in the third dimension is not available. There are several reports [26-28] on using X-ray computerized tomography (X-ray CT) to obtain images in 3 dimensions (3D) by taking pictures of slices of the specimen in 2D and reconstructing the internal structures into 3D. This process is computational expensive and requires long processing time. Due to these limitations, X-ray imaging has not been reported to be used on real structures.

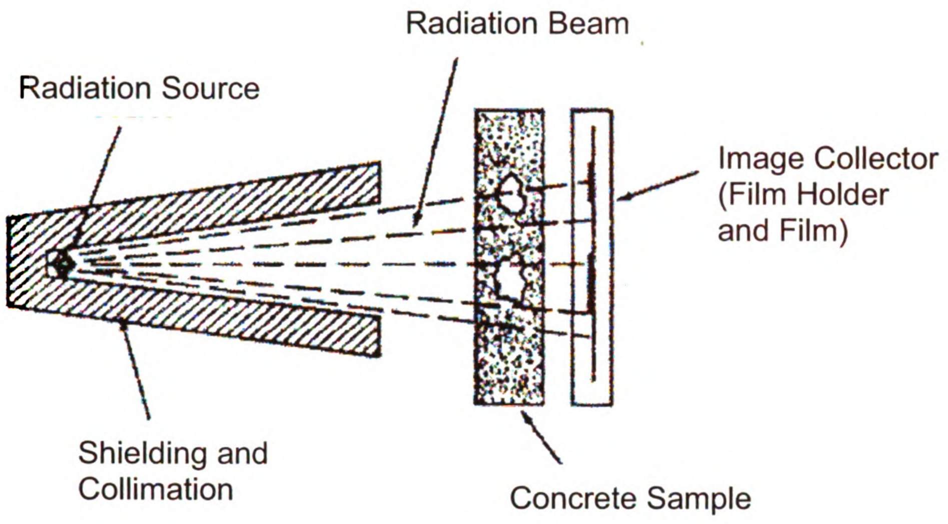
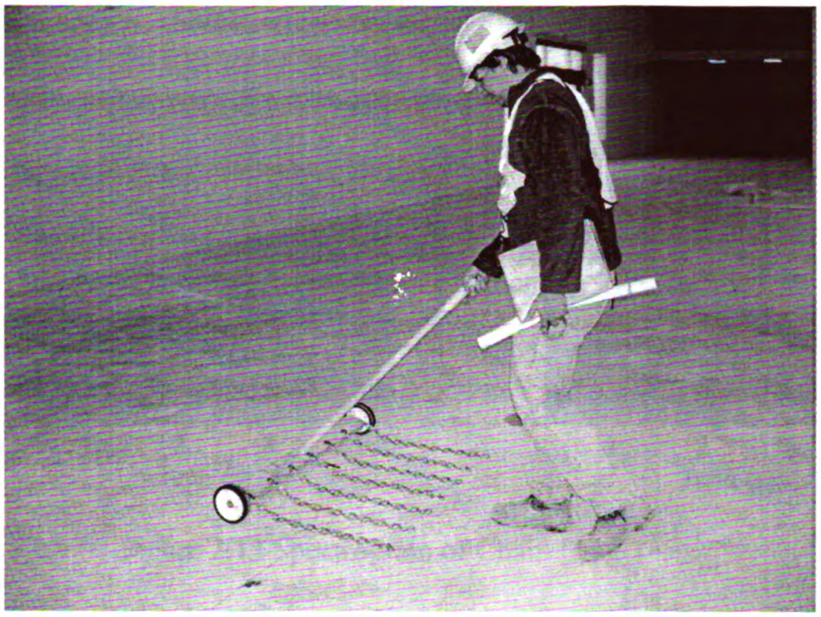

Radiation Beam
Radiation Source
Image Collector (Film Holder and Film)
Shielding and Collimation
Concrete Sample

Figure 2.11 X-ray Imaging of Concrete Samples [21]

2.2.6 Sounding Methods

Sounding techniques for non-destructive evaluation (NDE) of concrete decks have been widely used because they are fast, simple and inexpensive. Traditional sounding methods for delamination detection involve: (1) bar/hammer tapping of the deck or (2) dragging a chain over the deck as shown in Figure 2.12 and listening to the change in the sound. In both methods, good concrete with no delamination produces a clear ringing sound, while delaminated concrete is characterized by a dull, hollow sound. Standard test procedures are defined in ASTM C4580-2003 [11].



(http://www.rvtcorp.com/web pages/Services/Forensic Services.htm)

Figure 2.12 Chain Drag Test

Sounding methods have their own problems. The first problem arises due to traffic noise from adjacent lanes. Usually only one lane is closed for inspection and noise is generated by traffic in adjacent lanes as well as from wind and other sources. Figure 2.13 shows the spectrogram of the recorded signals under quiet and noisy environment. The complex environment makes the sound field difficult to analyze. Furthermore, the traffic noise is non-stationary and broadband. This makes the problem complicated and a simple band-pass filter cannot efficiently eliminate the noise. The second problem results from the fact that the detection is dependent on the subjective interpretation of the inspector, which makes it difficult to document the inspection results. Therefore, improvement of traditional sounding methods may enhance detection.

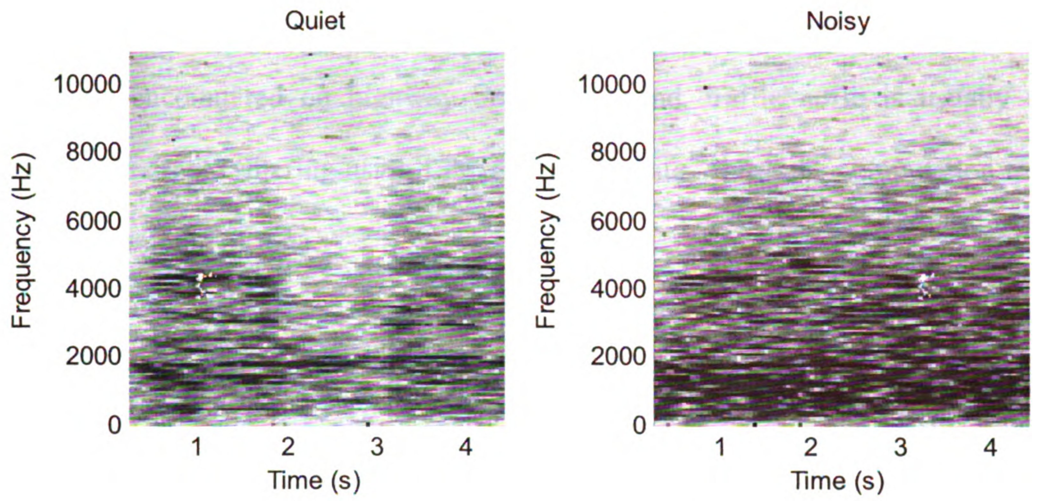


Figure 2.13 Spectrogram of Chain Drag Tests

Although, several attempts have been made to improve sounding methods, research on this topic is still quite limited. In 1977, researchers at the Michigan Department of Transportation (MDOT) designed a cart-like device for delamination detection [29]. The impulse was created by the chattering of two rigid wheels with the concrete and the vibration of the concrete was captured by a transducer coupled to the ground through soft tires and liquid in the wheels. The recorded signals were first truncated such that only the impact 5 ms after tapping was analyzed and then filtered by a fixed band pass filter with cut-off frequencies at 300 and 1200 Hz. The processed signals were recorded on charts. The audible signal was detected through earphones. This method was automatic, but the signal processing algorithm was primitive. Henderson et al. [30] used sound signals created by dragging a chain. The traffic noise was isolated by sound proofing around the chains and recording device. A computer algorithm using linear prediction coefficients (LPC) was used to analyze the recorded signals and perform the detection. Although this technique showed promise, the method had three major drawbacks. First, the traffic noise was reduced only by physical isolation and use of a directional microphone, which can be ineffective at high noise levels and for complex sound fields encountered on highway bridges. Second, traffic noise is usually non-stationary and simple filtering using LPC was inadequate. Third, the computation took a considerable amount of time.

Table 2.1Summary of Different Non-destructive Evaluation Methods

Methods	Applications and Advantages	Limitations
Impact Echo	♦ Can detect cracks, voids and delaminations ♦ The locations of the defects can also be determined	♦ Analyzing the results is difficult ♦ Shallow delaminations and delaminations parallel to stress propagation cannot be detected ♦ Sensors need to be coupled with surface
Ultrasonic Pulse Velocity	♦ Able to detect different types of detects ♦ Strength of concrete can also be determined ♦ Test procedure is easy	♦ Sensors have to be coupled with surface ♦ Accuracy can be affected by other factors ♦ It does not work well on asphalt overlays
Ground Penetrating Radar	♦ Has a wide range of applications ♦ Equipment is portable and mobile ♦ Inspection procedure is fast and result is provided in real time	♦ Interpreting the results requires professional knowledge ♦ Resolution and penetration needs to be balanced ♦ Air-filled defects cannot be detected
Infrared Thermography	♦ Non-contact method and fast to perform ♦ Equipment is mobile and provides results in real time ♦ Result is easy to understand	♦ External heat sources (active or passive) are needed ♦ Deeper defects cannot be detected ♦ Equipment is expensive
X-ray Imaging	♦ Has very good penetration capacity ♦ Contrast between concrete and air-filled defects is clear ♦ Presence and locations of the damage can be obtained	♦ Radiation is a safety concern ♦ Access to both sides of the specimens is needed ♦ Equipment is very expensive
Sounding	♦ The equipment is very cheap ♦ Inspection process is fast and easy ♦ Results are provided in real-time	♦ High traffic noise may affect the accuracy ♦ Detection process is subjective ♦ Extensive training of the operator is needed

2.3 Summary

This chapter described the commonly found defects such as craks air void, honeycombing and delamination in concrete and several non-destructive techniques for damage detection. Table 2.1 summaries of the advantages and limitations of different methods.

Each method has its own advantages and limitations. The selection of the NDE method should be based on the target defect. In the case of delamination detection for concrete bridge decks, the sounding method is a good choice because of its advantages of being inexpensive, simple and fast. However it has its own problems: noise contaminated signals and subjective interpretation of results. Research on the improvement of sounding methods has been limited. The research presented here will improve traditional sounding methods by focusing on the problem of noise c ancellation and automatic detection.

CHAPTER 3

NOISE CANCELLING ALGORITHMS

As described in Chapter 2, sounding tests are often conducted in a noisy environment. Traffic noise combined with other ambient noise such as wind often contaminates the sounding signals. This greatly reduces the accuracy of the delamination detection. Eliminating the unwanted noise can enhance the signal and improve the detection performance. Extensive research has been performed on this topic and various types of algorithm have been proposed and implemented. This chapter describes the technical details and performance of several commonly used algorithms and the selection of an effective algorithm for traffic noise cancellation.

3.1 Traffic Noise and Noise Cancelling Algorithms

Noise cancelling is a basic yet difficult problem. Figure 3.1 shows the power spectrum of a typical impact and the traffic noise. It can be seen that there is much overlap in the spectrum of the impact and noise. This means that the noise cannot be filtered using simple band-pass filters. Figure 3.2 shows that the properties of the traffic noise change due to the changing traffic and other factors. Due to the non-stationarity and the unpredictability of the traffic noise, the noise cancelling algorithm has to be adaptive and should require no prior information about the noise.

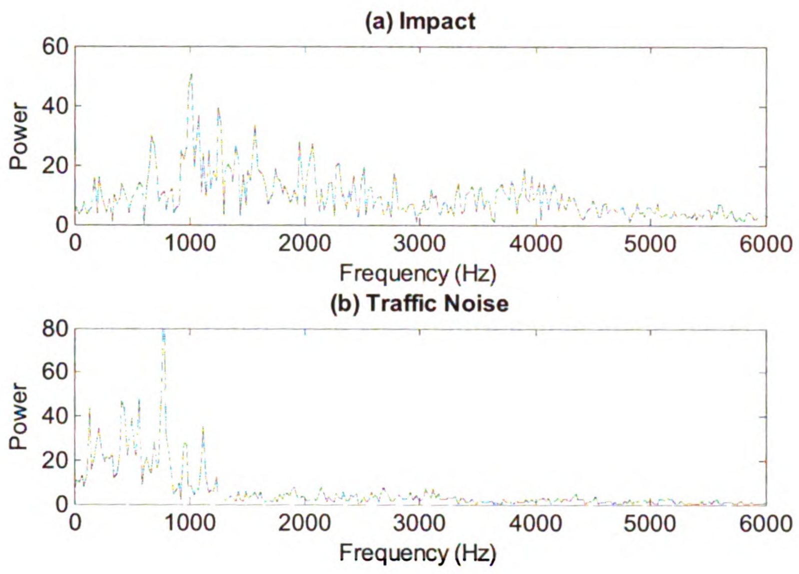


Figure 3.1 Spectrum of Impact and Traffic Noise

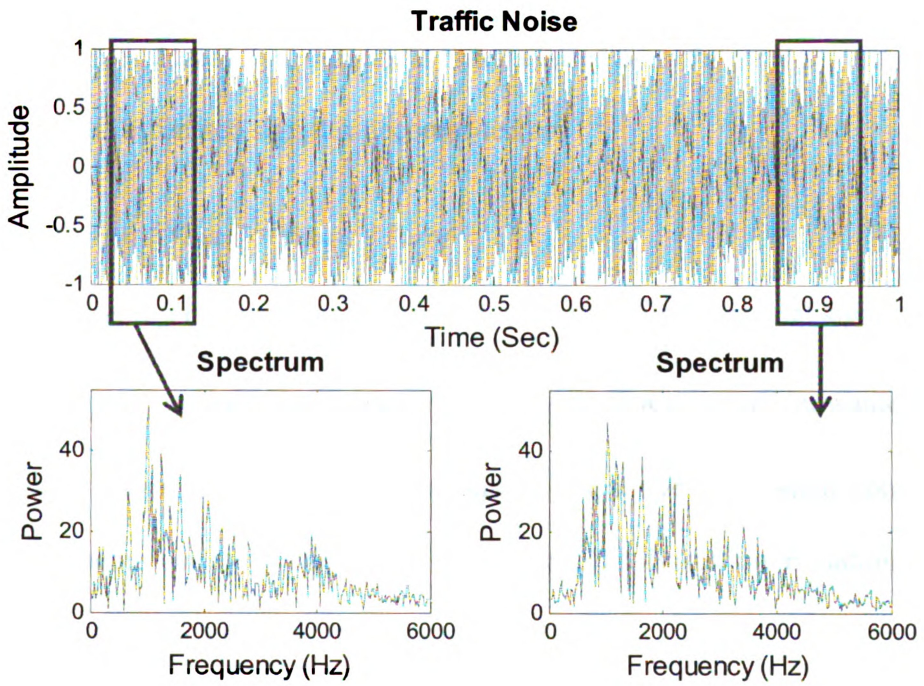


Figure 3.2 Non-Stationarity of Traffic Noise

3.2 Evaluation Criteria for Noise Cancelling Algorithms

There are various types of noise cancellng algorithms; each algorithm is designed for its own purposes. The performance of these algorithms needs to be evaluated in order to select a suitable algorithm for cancelling the traffic noise. The performance of algorithms can only be evaluated when some information about the system such as the original signal, the estimated signal, mixing type (instantaneous mixture or convolutive mixture) and filter length are available. For the work described, the impact signals were recorded in a quiet environment and the noise signal was obtained by recording the traffic noise on a highway bridge. The impact signal and traffic noise were mixed in different ways on the computer to simulate different mixing types. The performance of the algorithms was evaluated by a numerical criteria based on orthogonal projections [31]. In this method, the estimated signal is decomposed as:

$$\hat{s} = s_{target} + e_{interf} + e_{noise} + e_{artif} \tag{3.1}$$

where s_{target} represents the part of $\hat{s}$ from the wanted source (original signal in this case) and e_{interf} , e_{noise} and e_{artif} are the errors due to interference (unwanted sources), measurement noises and artifacts (other causes), respectively. Detailed computation of those components can be found in the reference [31]. Based on Equation (3.1), four performance criteria are defined as follows:

$$SDR = 10\log_{10}\frac{\left\|s_{target}\right\|^2}{\left\|e_{interf} + e_{noise} + e_{artif}\right\|^2} \tag{3.2}$$

$$SIR = 10\log_{10}\frac{\left\|s_{target}\right\|^2}{\left\|e_{interf}\right\|^2} \tag{3.3}$$

$$SNR = 10\log_{10}\frac{\left\|s_{target}+e_{interf}\right\|^2}{\left\|e_{noise}\right\|^2} \tag{3.4}$$

$$SAR = 10\log_{10}\frac{\left\|s_{target}+e_{interf}+e_{noise}\right\|^2}{\left\|e_{artif}\right\|^2} \tag{3.5}$$

where *SDR* = the Source to Distortion Ratio;

SIR = the Source to Interference Ratio;

SNR = the Source to Noise Ratio;

SAR = the Source to Artifacts Ratio.

Since the recordings here are simulated noise, there is no contribution from the unwanted sources and measurement noises and some artifacts may be introduced in the estimated signal due to the limit of the algorithm. Also, it can be shown from Equations (3.2) to (3.5) that SDR and SAR are equivalent in the absence of interference and measurement noise. In this work, only SDR was used as the performance criteria to evaluated different algorithms. The SDR were computed by a MATLAB [32] function coded by Vincent [31].

3.3 Spectral Subtraction

3.3.1 Theoretical Background

Spectral subtraction is a simple noise cancelling algorithm. A detailed description of this algorithm was provided in this section [33]. The algorithm assumes that the noisy recording is obtained by adding a windowed noise to a windowed signal, which can be expressed in the frequency domain as:

$$X\left(e^{jw}\right) = S\left(e^{jw}\right) + N\left(e^{jw}\right) \tag{3.6}$$

where $X\left(e^{jw}\right)$, $S\left(e^{jw}\right)$ and $N\left(e^{jw}\right)$represent the Fourier transform of the recording, the signal and noise, respectively.

For convenience, "recording" indicates the signal recorded by the microphone the includes unwanted noise; "signal" refers to the acoustic signal created by impacting the concrete using a hammer or other methods and "noise" refers to the ambient sound, such as traffic noise. These definitions are used for the remainder of the chapter.

Assuming that the noise is stationary over the duration of the recording, the spectrum of the noise can be estimated from the recording during the quiet period before or after the signal. The length of the noise recording can be increased by extracting and joining segments from adjacent windows. The spectrum of the signal can be estimated by:

$$\begin{aligned}\hat{S}\left(e^{jw}\right) &= \left[\left|S\left(e^{jw}\right)\right| - \left|\mu\left(e^{jw}\right)\right|\right] e^{j\theta_x} \\ &= H\left(e^{jw}\right) X\left(e^{jw}\right)\end{aligned} \tag{3.7}$$

where $\mu\left(e^{jw}\right)$ is the average value of the noise spectrum obtained from the recording segment with no signal and $H\left(e^{jw}\right)$ is calculated from Equation (3.8).

To reduce the variance in the spectrum of the noise, a longer window in the time domain is preferred. However, the actual noise is usually non-stationary indicating that the spectral properties of the noise change. The spectrum estimated from a long window will be an average over the entire window, which may not be a good estimate of noise spectrum during the period containing signal. This will introduce error in the final results. Therefore, a balance between these two factors should be considered in the selection of the window length.

$$H\left(e^{jw}\right)=1-\frac{\mu\left(e^{jw}\right)}{\left|X\left(e^{jw}\right)\right|} \tag{3.8}$$

In some instances $H\left(e^{jw}\right)$ may be negative, meaning that the sum of signal plus noise is less than the noise, which cannot be the case. Half-wave rectification is used to solve this problem, in which the negative value is replaced by zero. This process can be expressed by:

$$H_R\left(e^{jw}\right)=\frac{H\left(e^{jw}\right)+\left|H\left(e^{jw}\right)\right|}{2} \tag{3.9}$$

3.3.2 Performance Evaluation

The performance of the spectral subtraction algorithm was tested using simulated data. The traffic noise recorded from the highway was directly added to the impact signal and the noise in the adjacent window was used as the reference (signal recorded in quiet period) to estimate the signal in the previous window. The algorithm was implemented in the mathematical software MATLAB [32]. The results are shown in Figure 3.3. As can be seen from this figure, the original signal was not fully recovered. This may be due to two reasons. First, the traffic noise was not stationary and the properties of the noise in the window before the occurrence of the impact were different from the window in which the impact occurs. Second, there might be overlaps between the spectrum of the noise and that of the impact, and when the noise components were subtracted, some components of the impact signal also are likely to be cancelled. The SDR of this algorithm was computed to be -6.514 dB, indicating that the performance of the spectral subtraction algorithm was poor.

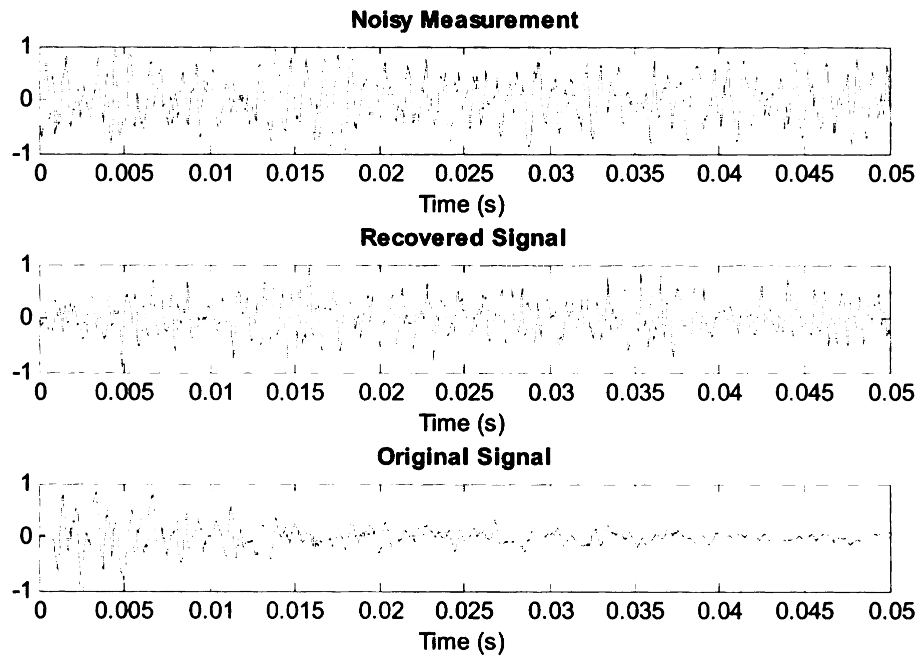


Figure 3.3 Performance of Spectral Subtraction

3.4 Adaptive Filters

3.4.1 Theoretical Background

As demonstrated in the previous section, the spectrum subtraction algorithm could not successfully reconstruct the impact signal from noisy recordings. The main reason for this is that the assumption that the noise signal is short-term stationary is not guaranteed for traffic noise at sites. Adaptive filter algorithms can be more effective in solving this problem. One of the commonly used adaptive filters is the least-mean-square (LMS) algorithm. Figure 3.4 [34] shows an adaptive noise cancelling system in which there are two microphones in the system. The primary microphone records the mixture of the source signal (s) and the noise (n_0) and the reference microphone is used to record a filtered version of the noise (n_l). The adaptive filter consists of a tapped delay line and the weights in the adaptive filter automatically seek an optimal impulse response by adjusting themselves such that the error between the outputs of the filter (y) is the best estimate of the noise in the primary microphone in the sense of least mean square error. The estimated source (z) can be obtained by subtracting the filter output from the primary signal.

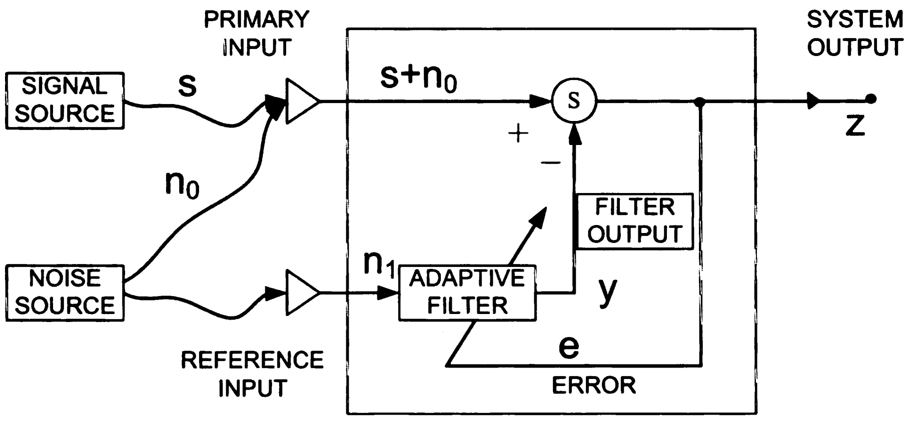


Figure 3.4 Noise Cancelling using Adaptive Filter

The mean square error (MSE) between the estimate source and the original source can be expressed as:

$$\begin{aligned} MSE &= E\left[(z-s)^2\right] \\ &= E\left[(s+n_0-y-s)^2\right] \\ &= E\left[(n_0-y)^2\right] \end{aligned} \tag{3.10}$$

where, $E[\cdot]$ represents the operation of expectation.

From Equation (3.10), it can be seen that minimizing the error between the estimated source and the original source is equivalent to minimizing the mean square error between the estimated noise and the noise in the primary microphone. Therefore, the LMS method can be used in the noise cancelling problem.

The output of the adaptive filter can be calculated from:

$$y = \sum_{l=1}^{L} w_l n_{1l} = W^T n_1 = n_1{}^T W \tag{3.11}$$

The MSE between the filter output and the noise signal can then be described as:

$$\begin{aligned} MSE &= E\left[(n_0-y)^2\right] \\ &= E\left[\left(n_0-n_1{}^T W\right)^2\right] \\ &= E\left[(n_0)^2\right] - 2E\left[n_0 n_1{}^T\right]\cdot W + W^T \cdot E\left[n_1 n_1{}^T\right]\cdot W \end{aligned} \tag{3.12}$$

The steepest descent algorithm updates the weight vector proportional to the gradient vector:

$$W_{j+1} = W_j - \mu \frac{\partial(MSE)}{\partial(W_j)} \tag{3.13}$$

where μ is a factor that controls the rate of adaptation.

From Equation (3.12), the gradient of the MSE can be computed as:

$$\frac{\partial(MSE)}{\partial(W)} = 2E\left[n_0 n_1{}^T\right] + 2W^T \cdot E\left[n_1 n_1{}^T\right] \tag{3.14}$$

Substituting Equation (3.14) into (3.13), the update law of the weight vector is:

$$\begin{aligned} W_{j+1} &= W_j - 2\mu n_{0j} n_{1j}{}^T + 2W^T \cdot n_{1j} n_{1j}{}^T \\ &= W_j - 2\mu\left(n_{0j} - W^T n_{1j}\right) n_{1j}{}^T \\ &= W_j - 2\mu\left(n_{0j} - y_j\right) n_{1j}{}^T \\ &= W_j - 2\mu e_j n_{1j}{}^T \end{aligned} \tag{3.15}$$

where, e_j is the error between the filter output and the desired signal.

For the adaptive noise canceling algorithm, the desired signal is the source ($s+n_0$). Therefore, the error in Equation (3.15) is in fact the estimated signal (z), as shown below:

$$e_j = (s + n_0) - y = z \tag{3.16}$$

Therefore, the update law for the weight vector becomes:

$$W_{j+1} = W_j - 2\mu z_j n_{1j}{}^T \tag{3.17}$$

In the LMS method, the cost function that the algorithm tries to minimize is calculated from the MSE in the current step. Therefore, the performance and convergence rate of the algorithm may be affected by the transient response. To improve the performance, a modified LMS algorithm called the recursive least square (RLS) algorithm is used in this section. Instead of minimizing the MSE from the current step, the RLS algorithm minimizes the total MSE over N steps, as shown below:

$$C = \sum_{i=1}^{N} \beta(n,i)|e(i)|^2 \tag{3.18}$$

where $\beta(n,i)$ is the weighting or "forgetting" factor.

The derivation of the update law is similar to that in the LMS method and will not be described in details here. The convergence of the RLS algorithm is much faster than that of the LMS method but the computation time is longer.

3.4.2 Performance Evaluation

To evaluate the performance of the adaptive filter, several difference cases were simulated. The noisy recordings of different conditions were obtained by mixing the scaled impact signal with the noise signal as shown below:

$$m(t) = \alpha s(t) + n(t) \tag{3.19}$$

where m = the simulated measurement;

α = scaling coefficient that controls the SNR;

s = the clean signal;

n = the traffic noise signal.

Case 1: The recording of the primary microphone was simulated by directly adding the traffic noise to the impact signal. In this case, the scaling factor is 1. The recording of the secondary microphone was simulated as a filtered version of the same traffic noise used for the primary microphone. The length of the adaptive filter was the same as that of the filter used to create the reference signal. In this case, a filter length of 5 was used.

Case 2: The primary and reference signals were the same as in Case 1 but the length of the adaptive filter was 2.

Case 3: The primary and reference signals were the same as in Case 1 but the length of the adaptive filter was 4.

Case 4: The primary signal was the same as in Case 1, but the reference signal was also a mixed version of the filtered traffic noise and impact signal. The scaling factor was 0.316. The length of the filter was assumed to be 5 for both the adaptive filter and the actual filter.

Case 5: This case was identical to Case 4 except that the scaling factor in the reference signal was 0.0316.

The RLS algorithm performs very well if the length of the adaptive filter is equal to or greater than the actual filter (such as in Case 1), as shown in Figure 3.5. The source signal was masked in the noisy recording, but was effectively and rapidly recovered by the RLS algorithm. The SDR of Case 1 was calculated to be 10.51dB.

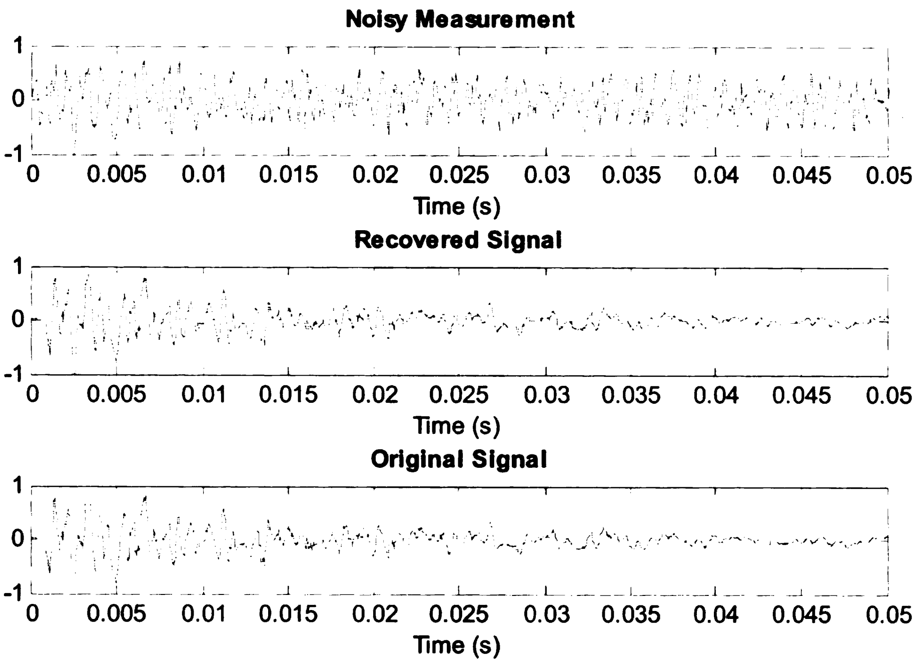


Figure 3.5 Performance of RLS Adaptive Filter (Case 1)

However, if the length of the filter is underestimated, the performance will drop because the effect of the actual filter cannot be fully represented by filters with a shorter length. Figure 3.6 shows the results for Case 2. The SDR for this case was -6.212 dB, which was considerably lower than for the ideal case. When the length of the filter was increased to 4 (Case 3), the performance index SDR increased to 1.658 dB. The results for this case are shown in Figure 3.7

The comparison above indicates that the SDR is a good performance measure. It is very difficult to judge which case has better performance from visual inspection. However, the SDR is able to detect the change in performance.

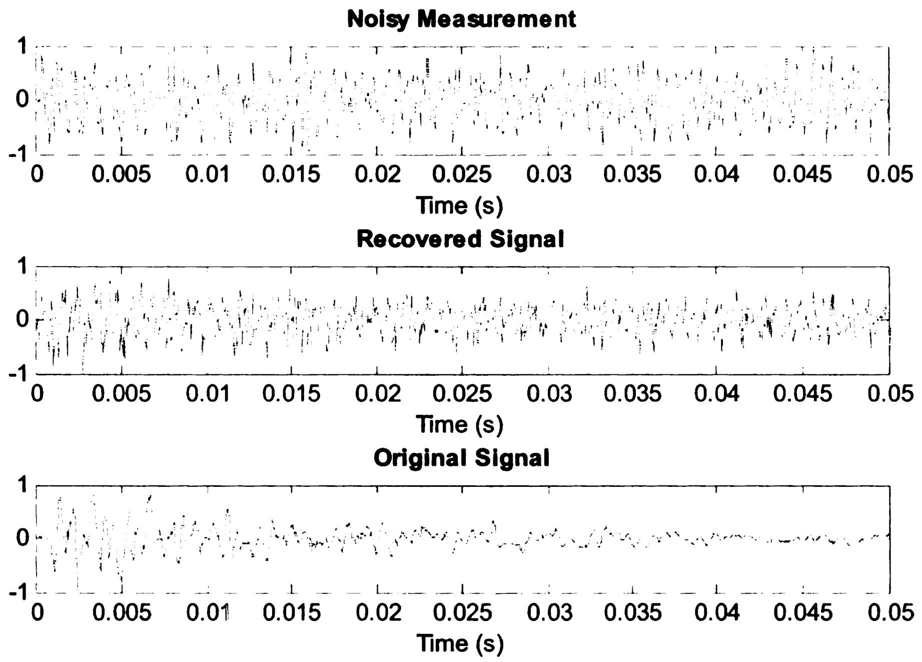


Figure 3.6 Performance of RLS Adaptive Filter (Case 2)

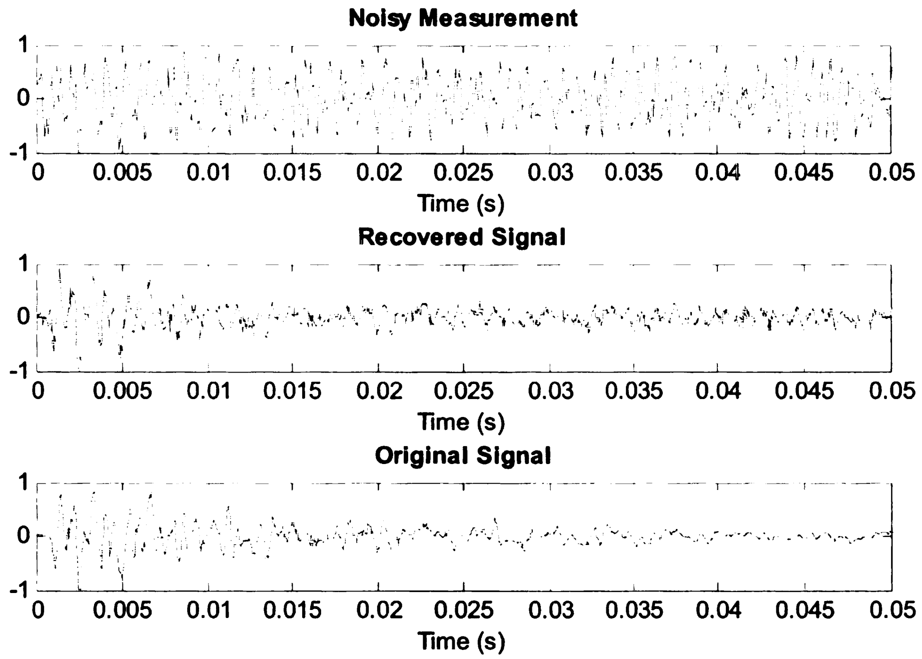


Figure 3.7 Performance of RLS Adaptive Filter (Case 3)

Another problem with this algorithm is that the source in the primary signal will inevitably be cancelled when some source components leak into the reference signal. This leads to distortion and reduces the performance. Figure 3.8 shows the performance of Case 4 where the source and the noise were mixed at the signal to noise ratio of 0.316 in the reference recording. It can be seen that the signal is significantly distorted. The SDR of the recovered signal was -6.212 dB. If the signal to noise ratio in the reference signal became $1X10^{-5}$ (Case 5), the performance improved since only a fraction of the source signal was cancelled. This increase in performance was also reflected in the waveform shown in Figure 3.9. The SDR in this case increased to -2.334 dB, but was still much lower than the SDR for the ideal case.

From the above discussion, it can be concluded that the adaptive filter can efficiently cancel the unwanted noise under ideal conditions, (i.e., when the length of the adaptive filter is equal to or greater than the actual filter and there is no signal component in the reference signal). However, the performance drops quickly if these requirements are not statisfied.

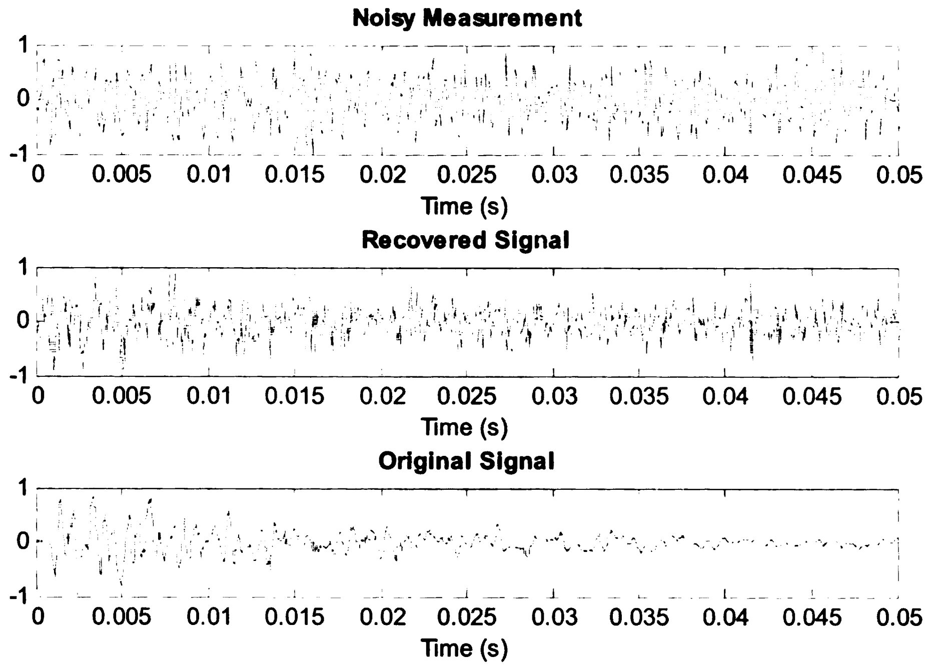


Figure 3.8 Performance of RLS Adaptive Filter (Case 4)

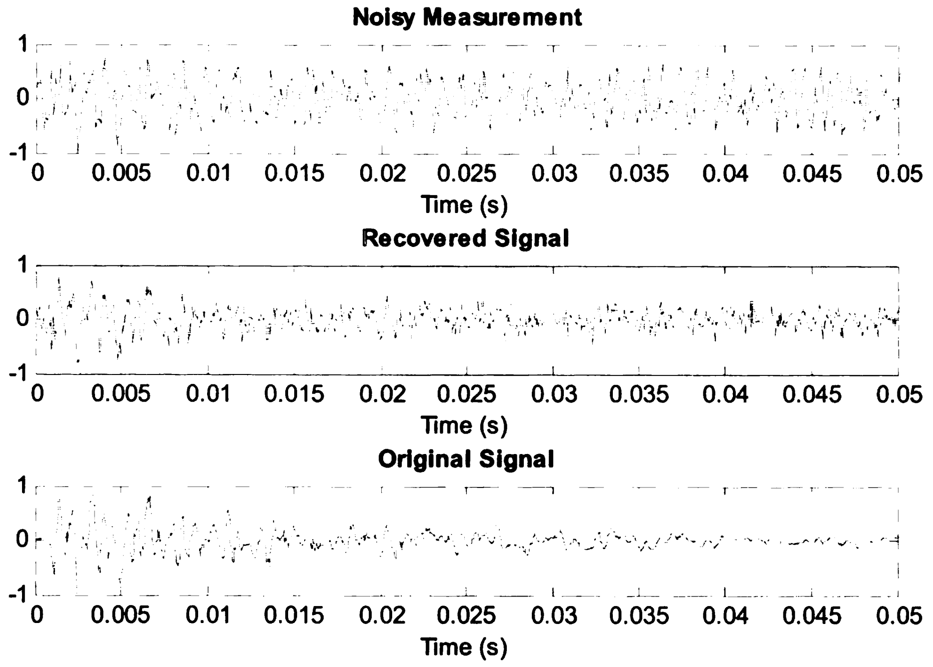


Figure 3.9 Performance of RLS Adaptive Filter (Case 5)

3.5 Independent Component Analysis

3.5.1 Theoretical Background

Case 5 in the previous section showed that the presence of a small portion of the source signal in the reference recording led to a significant decrease in the SDR of the reference signal. Independent component analysis (ICA) is employed in this section to solve the problem. The concept of ICA was first proposed by Comon in 1994 [35] and illustrated in Figure 3.10. The algorithm assumes that the sources are mutually independent. The de-mixing of the recordings can be performed by maximizing the independence between the outputs of the algorithm. Once the independence is maximized, the outputs will be scaled versions of the original sources. The maximization of independence is realized by the adaptation of a de-mixing matrix. The detailed derivation of the method is described as follows [36]:

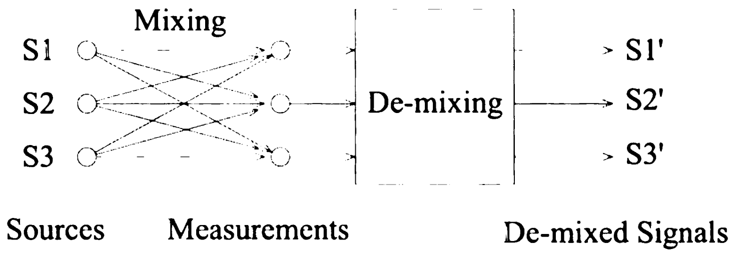


Figure 3.10 Independent Component Analysis

The output of the de-mixing matrix (W) can be expressed as:

$$Y = WX \tag{3.20}$$

where X = matrix of mixed signals.

Y = matrix of de-mixed signals.

W = de-mixing matrix.

The probability density function (PDF) of de-mixed signals is assumed to be $f_Y(y,W)$. The objective is to adjust the de-mixing matrix such that the output signals Y_i and Y_j are independent. If the output signals are independent, the PDF of de-mixed signal can be expressed as:

$$\tilde{f}_Y(y,W)=\prod_{i=1}^{m}\tilde{f}_{Yi}(y_i,W) \tag{3.21}$$

where $\tilde{f}_{Yi}(y_i,W)$ is the marginal PDF of the Y_i output signal.

The objective can therefore be achieved by minimizing the "difference" between $f_Y(y,W)$ and $\tilde{f}_Y(y,W)$. In a statistical sense, one common way to measure the difference between two PDFs is the Kullback-Leibler (KL) divergence. The KL divergence between two PDFs, f_X and g_X, is computed as:

$$D_{f_X\|g_X}=\iint f_X(X)\log\left(\frac{f_X(X)}{g_X(X)}\right)dx \tag{3.22}$$

Substituting the expressions for $f_Y(y,W)$ and $\tilde{f}_{Yi}(y_i,W)$ into Equation (3.22) yields:

$$
\begin{aligned}
D_{f_Y \| \tilde{f}_Y} &= \iint f_Y(Y) \log \left(\frac{f_Y(Y)}{\prod_{i=1}^{m} \tilde{f}_{Y_i}(y_i, W)} \right) dy \\
&= \int f_Y(Y) \log f_Y(Y)\, dy - \sum_{i=1}^{m} \int f_Y(Y) \log \tilde{f}_{Y_i}(Y_i)\, dy \\
&= -h(Y) - \sum_{i=1}^{m} \tilde{h}(Y_i)
\end{aligned}
\tag{3.23}
$$

where $h(\cdot)$ is the entropy of a random variable that can be calculated as:

$$
h(x) = \int_{-\infty}^{+\infty} f_X(X) \log f_X(X)\, dx \tag{3.24}
$$

The steepest descent method described in Section 3.4 can then be used to minimize the KL divergence. In order to derive the update law, the gradient of the KL divergence needs to be found. The gradient of the entropy $h(Y)$ is found by:

$$
\begin{aligned}
\frac{\partial h(Y)}{\partial W} &= \frac{\partial}{\partial W}\left(h(WX) \right) \\
&= \frac{\partial}{\partial W}\left(h(X) + \log\left|\det(W)\right| \right) \\
&= \frac{\partial}{\partial W}\left(\log\left|\det(W)\right| \right) \\
&= \frac{1}{\det(W)} \frac{\partial}{\partial W}\left(\log \left| \sum_{k=1}^{m} w_{ik} A_{ik} \right| \right) \\
&= \frac{A_{ik}}{\det(W)} = \left(W^{-1} \right)^{T}
\end{aligned}
\tag{3.25}
$$

$\tilde{h}(Y_i)$ can be expressed by truncating the Gram-Charlier (GC) expansion [37] of the corresponding PDF at different level. For example, PDF can be expressed as:

$$\tilde{f}_{Yi}(y_i) \approx \frac{1}{\sqrt{2\pi}} \exp\left(-y_i^2\right)$$
$$\cdot \left(1 + \frac{\kappa_{i,3}}{3!} H_3(y_i) + \frac{\kappa_{i,2}^2}{4!} H_4(y_i) + \frac{\kappa_{i,6} + \kappa_{i,3}^2}{6!} H_6(y_i) \right) \tag{3.26}$$

where $H_k(y_i)$ are Hermite Polynomials and $\kappa_{i,k}$ are the cumulants of Y_i.

By taking the log of the equation above and using the Taylor expansion, $\tilde{h}(Y_i)$ can be expressed as:

$$\begin{aligned}\tilde{h}(Y_i) \approx{}& \frac{1}{2}\log(2\pi e) - \frac{\kappa_{i,3}^2}{12} - \frac{\kappa_{i,4}^2}{48} - \frac{\left(\kappa_{i,6} + 10\kappa_{i,3}^2\right)^2}{1440} \\ &+ \frac{3\kappa_{i,3}^2\kappa_{i,4}}{8} + \frac{\kappa_{i,3}^2\left(\kappa_{i,6} + 10\kappa_{i,3}^2\right)}{24} + \frac{\kappa_{i,4}^2\left(\kappa_{i,6} + 10\kappa_{i,3}^2\right)}{24} \\ &+ \frac{\kappa_{i,4}\left(\kappa_{i,6} + 10\kappa_{i,3}^2\right)^2}{64} + \frac{\kappa_{i,4}^3}{24} + \frac{\left(\kappa_{i,6} + 10\kappa_{i,3}^2\right)^3}{432}\end{aligned} \tag{3.27}$$

The derivative of the cumulants $\kappa_{i,3}$ can be calculated as:

$$\begin{aligned}\frac{\partial \kappa_{i,3}}{\partial W} &= \frac{\partial}{\partial W}\left(E\left[Y_i^3\right]\right) \\ &= E\left[\frac{\partial}{\partial W}\left(Y_i^3\right)\right] = E\left[Y_i^2 \frac{\partial}{\partial W}\left(\sum_{j=1}^{m} w_{ij} X_i\right)\right] \\ &= E\left[Y_i^2 X\right]\end{aligned} \tag{3.28}$$

The derivatives of other cumulants can be derived in a similar way. The final update law for the ICA is:

$$\begin{aligned} W(n+1) &= W(n) + \Delta W(n) \\ &= W(n) - \eta \frac{\partial}{\partial W}\left(D_{f \| \tilde{f}}\right) \\ &= W(n) + \eta\left[W(n)^{-T} - \varphi(y) X^T\right] \\ &= W(n) + \eta\left[I - \varphi(y) X^T W'(n)^T\right] W(n)^{-T} \\ &= W(n) + \eta\left[I - \varphi(y) y^T\right] W(n)^{-T} \end{aligned} \tag{3.29}$$

where $\varphi(y)$ is the activation function derived from the Gram-Charlier expansion described above and η is the learning rate factor, controlling the rate of adaptation and convergence of the algorithm.

Depending on the order of the Gram-Charlier series, $\varphi(y)$ can have different expressions. One typical activation function is:

$$\begin{aligned} \varphi(y) &= \frac{1}{2} y_i^5 + \frac{2}{3} y_i^7 + \frac{15}{2} y_i^9 + \frac{2}{15} y_i^{11} - \frac{112}{3} y_i^{13} \\ &+ 128 y_i^{15} - \frac{512}{3} y_i^{17} \end{aligned} \tag{3.30}$$

The iteration is continued until convergence criteria are met. The resulting W is the de-mixing matrix that will separate individual sources from the mixture. The separated sources can be computed using equation (3.20). Although the algorithm is complex to derive, it is very simple to implement.

Recently, researchers have developed different ICA algorithms to improve performance. One of the main differences among these algorithms is the estimation of the PDF of Y_i (the estimated original sources). That is, different forms of Equation (3.30) will lead to different ICA algorithms. In this research, an ICA algorithm called EFICA [38] is used. The accuracy given by the residual variance reaches the Cramer-Rao lower bound [39] and therefore this algorithm is asymptotically efficient or Fisher efficient.

It should also be noted that the recordings from the microphones have to be "different" enough to contain enough "information" about the sources. Otherwise, it will lead to a singular or ill-conditioned correlation matrix and the ICA algorithm will become unstable or inaccurate.

3.5.2 Performance Evaluation

To test the performance of the linear ICA described above, both instantaneous/linear mixtures and convolutive mixtures were used. In the instantaneous mixtures, the recordings were simulated through linear superposition of the sources as:

$$x_i(n) = \sum_{i=1}^{m} A_{im} s_m(n) \tag{3.31}$$

In convolutive mixtures, the recordings are the combinations of filtered sources, shown below:

$$x_i(n) = \sum_{j=1}^{d} \sum_{\tau=1}^{M_{ij}} h_{ij}(\tau) s_j(n-\tau) \tag{3.32}$$

where, x_i is the i^{th} observed signal or recording, s_m is the m^{th} source signal, A is the mixing matrix, and h_{ij} is the filter between the i^{th} microphone and the m^{th} source.

The difference between Equations (3.31) and (3.32) is that the elements in the unknown mixing matrix A in (3.31) are replaced by an unknown filter h_{ij} and the matrix multiplication is replaced by convolution.

The performance of EFICA was tested using both the instantaneous mixtures and the convolutive mixtures. For the instantaneous mixtures, one input channel was obtained by the direct addition of the traffic noise and the impact signal; the other input channel was also a linear addition of the traffic noise and impact signal but at a different ratio. For the convolutive mixtures, the first channel was the same as that of the instantaneous channel, but the second channel was a mixture of the filtered version of the traffic noise and impact signal. The length of the filter was assumed to be 5. The outputs of EFICA for both cases are shown in Figure 3.11 and Figure 3.12, respectively. As can be seen from the results, EFICA performed very well for instantaneous mixtures and the original signal was successfully extracted from the noisy recordings. However, EFICA could not separate signals from the convolutive mixtures. The reason for this is that the convolution operation brings delayed versions of sources into the mixture. The delayed versions of sources are considered independent by the algorithm and there are now more independent sources than recordings. The ICA problem therefore becomes indeterminate. The difference in the performance of EFICA for these two cases was also reflected in the SDR of the recovered signals: the SDR was 71.34 dB for linear mixtures and was only -10.81 dB for convolutive mixtures.

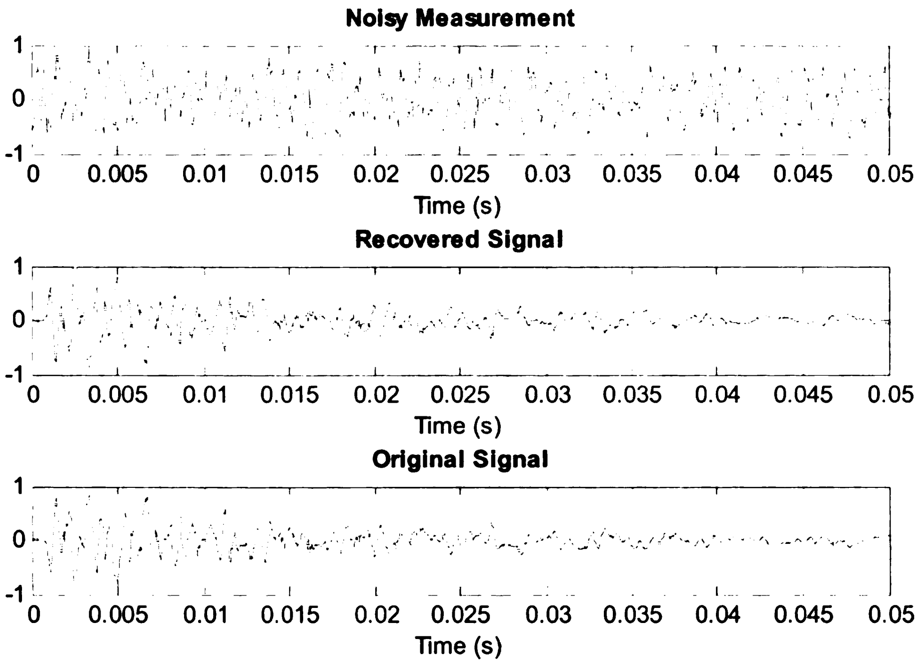


Figure 3.11 Performance of EFICA for Instantaneous Mixture

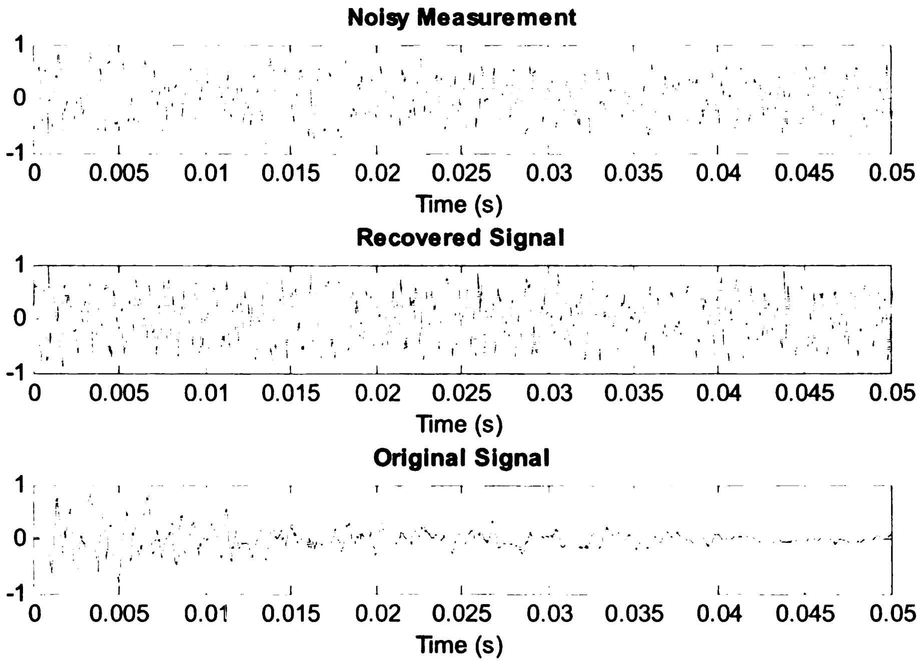


Figure 3.12 Performance of EFICA for Convolutive Mixture

3.6 Modified ICA

3.6.1 Theory Background and Procedures

As discussed in the previous section, traditional ICA requires: (1) the recordings must be a linear mixture of the sources; and (2) the recordings must be different enough. To satisfy the first requirement, the microphones should be placed as close as possible, but this conflicts with the second requirement since the microphones that are close are likely to record very similar signals. To meet the second requirement, the microphones should be placed at different locations, but this will make the recordings from the two microphones become convolutive mixtures, from which the sources cannot be separated by traditional ICA algorithms.

From Equation (3.32), it can be seen that the convolutive mixture is combination of scaling and shifting. If the shifted version of the recordings were de-mixed by instantaneous ICA (such as EFICA), the outputs are shifted and scaled versions of the original sources. Based on this concept, a modified ICA algorithm in the time domain [40] was used to perform the source separation of convolutive mixture. The procedures for the modified ICA algorithm are described as follows.

Step 1: A "convolutive sphering" on the recordings is performed, as shown below. In this step, the delayed version of the recording is used as additional recordings as shown below:

$$\tilde{X} = \begin{bmatrix} x_1(n) & x_1(n-1) & \dots & x_1(n-L+1) \\ x_1(n-1) & x_1(n-2) & \dots & x_1(n-L) \\ \dots & \dots & \dots & \dots \\ x_1(n-L) & x_1(n-L-1) & \dots & x_1(1) \\ x_2(n) & x_2(n-1) & \dots & x_2(n-L+1) \\ \dots & \dots & \dots & \dots \\ x_2(n-L) & x_2(n-L-1) & \dots & x_2(1) \\ \dots & \dots & \dots & \dots \\ x_m(n-L) & x_m(n-L-1) & \dots & x_m(1) \end{bmatrix} \tag{3.33}$$

where $\tilde{X}$ is the rearranged input with a dimension of $mL \times (n-L)$, x_i is the i^{th} observed signal or recording, L is the number of delays, n is the length of the block for the training of the traditional ICA, and m is the number of recordings.

Step 2: The rearranged input $\tilde{X}$ is de-mixed by traditional ICA algorithms (such as EFICA) and the de-mixed outputs c are a delayed and scaled source signals [41]:

$$c(n) = W\tilde{x}(n) \tag{3.34}$$

where W is obtained through an ICA algorithm.

Step 3: The similarities or the distance between each de-mixed output (or independent components) from Step 2 is calculated based on correlation-based criteria. To do this, vector $\tilde{c}_i$ and a time shift operation are defined as:

$$\tilde{c}_i = \left[c_i(L+1) \quad c_i(L+2) \quad \dots \quad c_i(N-L)\right]^T \tag{3.35}$$

$$D^k \tilde{c}_i = \left[\tilde{c}_i(L+1+k) \quad c_i(L+2+k) \quad \dots \quad c_i(N-L+k)\right]^T \tag{3.36}$$

Then the distance between the i^{th} and j^{th} independent components can be calculated as:

$$D_{ij} = \left\| \tilde{c}_j - \tilde{C}_i \left(\tilde{C}_i^T \tilde{C}_i \right)^{-1} \tilde{C}_i^T \tilde{c}_j \right\|^2 \tag{3.37}$$

where:

$$\tilde{C}_i = \left[D^{-L}\tilde{c}_i \quad D^{-L+1}\tilde{c}_i \quad \ldots \quad D^{L-1}\tilde{c}_i \quad D^{L}\tilde{c}_i \right] \tag{3.38}$$

Step 4: The independent components from Step 2 are grouped into m groups based on the similarity matrix D in Step 3, where m is the number of sources. Here, a hierarchical clustering algorithm using an average-linkage method is used. The method is described as follows:

(1) Assign each IC to a cluster. If there are n ICs, there are now n clusters;

(2) Find the closest (most similar) pair of clusters and merge them into a single cluster. The number of clusters is now reduced by one and becomes $n-1$;

(3) Compute distances between the new clusters using the average-linkage strategy (the new distance is the average distance of the two merged clusters);

Repeat (2) and (3) until the number of clusters is reduced to the target. In this case, the target is the number of sources.

Step 5: The contribution of source i to $\tilde{X}$ in Step 1 is determined by the inverse of the de-mixing process, in which the de-mixing matrix corresponding to source i is computed based on the similarity matrix in Step 3 and clustering:

$$\tilde{X}^i = W^{-1}\text{diag}\left[\lambda_1^i \quad \ldots \quad \lambda_{mL}^i \right] c \tag{3.39}$$

where $\tilde{X}^i$ is the contribution of source i to $\tilde{X}$ and c is the independent components computed in step 2. λ_k^i are the weighting factors computed from:

$$\lambda_k^i = \left(\frac{\sum_{j \in K_i, j \neq k} D_{kj}}{\sum_{j \notin K_i, j \neq k} D_{kj}} \right)^{\alpha} \tag{3.40}$$

where K_i are the indices belonging to cluster i and α is a positive factor that controls the "hardness" of the weighting.

The influence of source i on microphone k is defined as:

$$\hat{s}_k^i(n) = \sum_{p=1}^{L} \tilde{X}^i_{(k-1)L+p}(n+p-1) \tag{3.41}$$

Step 6: The sources are reconstructed from $\tilde{X}^i$ by inverting the "convolutive sphering" process in Step 1.

The process of the algorithm is briefly shown in Figure 3.13

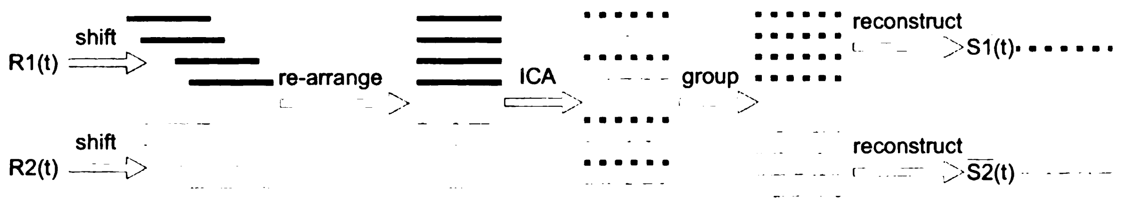


Figure 3.13 Modified ICA

3.6.2 Performance Evaluation

To compare the modified ICA algorithm and the traditional ICA described in Section 3.5, both an instantaneous mixture and a convolutive mixture was used to test the performance. The signal used for performance evaluation was the same as in Section 3.5. The results of linear and convolutive mixtures are shown in Figure 3.14 and Figure 3.15, respectively, and indicate that the impact signal was successfully recovered by the modified ICA algorithm for both types of mixtures. The SDRs of the algorithm for instantaneous and convolutive mixtures were 3.609 dB and 1.210 dB, respectively. Even though the SDRs of the modified ICA were not as high as that of the EFICA for the linear mixture case, the performance on convolutive mixtures exceeded that of EFICA. In fact, when the recovered signal was played back, no significant difference was detected between the original source and the recovered signal from the modified ICA.

This algorithm requires that the delay has to be predefined for successful separation. However, since both microphones are located on the impacting cart and are separated only by a small distance, the delay is not large and can be estimated through:

$$L = \frac{F_s}{v/(d_1 - d_2)} \tag{3.42}$$

where F_s is the sampling frequency (Hz), d_1 and d_2 are the distances between the microphones and the impact point and v is the velocity of sound in air.

If the distance difference of the two microphones is one meter and taking the velocity of sound in air to be 340 m/s, a delay of 25 samples is enough for successful separation at a sampling frequency of 8000 Hz.

The results showed that the modified ICA works on both instantaneous and convolutive mixtures. The predefined delay can be easily estimated.

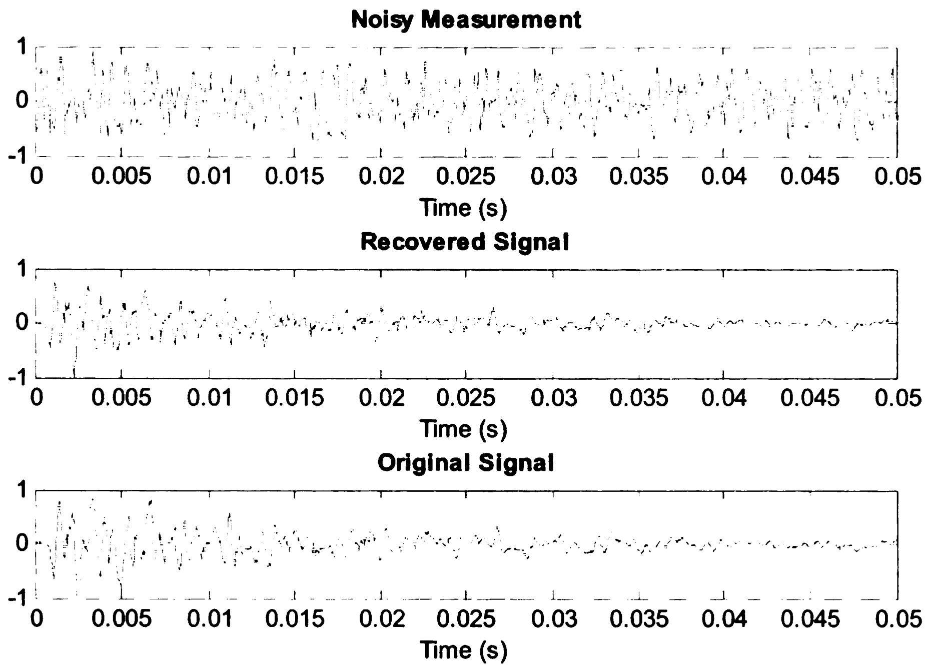


Figure 3.14 Performance of Modified ICA for Instantaneous Mixture

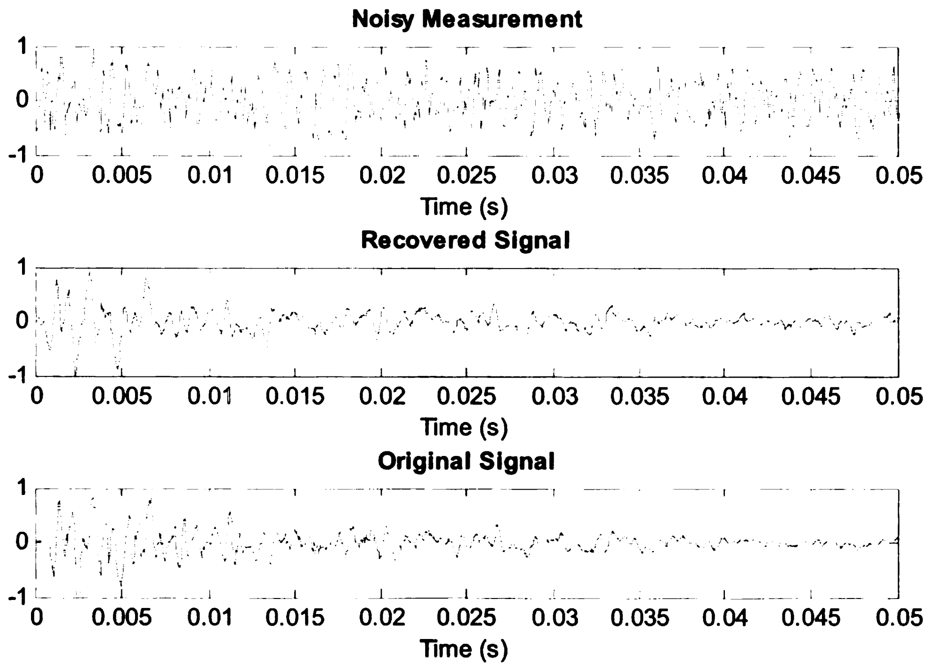


Figure 3.15 Performance of Modified ICA for Convolutive Mixture

3.7 Selection of Noise Cancelling Algorithms

Field inspection requires that a noise cancelling algorithm should meet the following criteria:

1. The algorithm must be able to cancel/separate non-stationary sources to be effective with changing the traffic noise in adjacent lanes.

2. No prior information about sources should be required since the traffic noise cannot be predicted or controlled and the impact sound changes from case to case.

3. There should be no strict requirement on the recording device, such as directionality, etc.

4. The algorithm should work for convolutive mixtures, if two or more microphones are used.

Table 3.1 compares different noise cancelling algorithms mentioned in this chapter with respect to the above four requirements.

Table 3.1 Comparison of Noise Cancelling Algorithms.

	Requirement 1	**Requirement 2**	**Requirement 3**	**Requirement 4**
Spectral Subtraction	X	√	√	N.A
RLS	√	√	X	√
ICA	√	√	√	X
Modified ICA	√	√	√	√

To quantitatively compare the performance of the noise cancelling algorithms, Table 3.2 lists the SDRs of the algorithms. Since the recordings at the site are usually convolutive mixtures, only the results for the convolutive mixture are listed.

Table 3.2 Performance for Convolutive Mixtures.

	Spectral Subtraction	**RLS**	**ICA**	**Modified ICA**
SDR (dB)	-6.514	-6.212 ~ -2.334	-10.81	1.210

It is obvious from Table 3.1 and Table 3.2 that the modified ICA meets all the requirements for field inspections and has the best overall performance under real scenarios (convolutive mixtures). Therefore, the modified ICA was selected as the noise cancelling algorithm for acoustic delamination detection.

3.8 Summary

This chapter described the evaluation criteria for noise cancelling algorithms and the technical details of commonly used noise cancelling algorithms.

The performance evaluation of noise cancelling needs to be simple and objective. An objective performance measure, the signal to distortion ratio (SDR), was introduced. Then, the technical details of four commonly used noise cancelling or source separation algorithms (spectral subtraction, recursive least square adaptive filter, independent component analysis and modified independent analysis) were described. The SDR and time-domain signal comparisons were used to evaluate the performance of each individual algorithm. Each algorithm performed differently under different types of recordings and has its own advantages and disadvantages.

Spectral subtraction is simple and easy to implement. It uses recordings from only one microphone. The noise in the recordings is estimated from the spectrum during the quiet period and is then subtracted from the spectrum of the impact period. However, this algorithm requires that the noise signal be short-term stationary, which is hard to guarantee in field inspections.

In the recursive least square (RLS) algorithm, an adaptive filter is used to estimate the noise recorded by the primary microphone using the recording from the reference microphone. Noise in the primary recording is estimated from the reference recording by minimizing the mean square error (MSE) between the output of the adaptive filter and the desired output. The signal is recovered by subtracting the estimated noise from the primary microphone. RLS adaptively adjusts the coefficients of the filter and can work with non-stationary signals, but requires the signal from the reference microphone to be pure noise, which is hard to satisfy in field inspection. It also needs a good estimate of the filter length.

A Fisher efficient independent component analysis (ICA) algorithm called EFICA was employed to release the requirement that there should be no signal in the recording by the reference microphone. This algorithm maximizes the independence between the outputs to separate sources from the mixture by an adaptive de-mixing matrix. The coefficients of the de-mixing matrix are adaptively changed. The algorithm can separate sources without prior information about them. However, it requires that the recordings from the two microphones be linear mixtures of the signal and noise, which is not the case for the signals recorded at bridge sites.

A modified ICA was therefore proposed to separate the signal from a convolutive mixture. This method can be used to separate both linear and convolutive mixtures. The algorithm requires that the maximum delay between the two microphones be predefined, and it can be estimated without much error.

The candidate algorithms were then compared and evaluated by considering both the requirements of field inspection and the performance for the convolutive mixtures, which is representative of signals recorded in the field. The modified ICA algorithm performed the best and was selected for the remainder of the research.

CHAPTER 4

FEATURE EXTRACTION ALGORITHMS

After the noise in the recordings was removed by implementing the noise canceling algorithm described in Chapter 3, the next step in the delamination detection is to relate the characteristics of the acoustic signals with the existence of delamination. As mentioned in Chapter 2, the delamination of the concrete bridge deck is characterized by a dull, hollow sound. This criterion is subjective and difficult to implement in an automatic detection algorithm. An objective criterion is needed to separate the "delaminated" sound and the "solid" sound. In this chapter, different feature extraction algorithms are compared and the best algorithm for delamination detection is selected.

4.1 Feature Extraction of Acoustic Signals

Figure 4.1 shows typical impact signals from solid and delaminated concrete. It is very difficult to differentiate the two signals in terms of their waveform. Therefore, other characteristics of the signal need to be extracted for the purpose of detection. The characteristics of the signal can be obtained by extracting features that quantify the acoustic signals. This step also reduces the dimension of the signal to avoid "the curse of dimensionality" [42]. The extracted features are further selected based on different selection criteria to eliminate features that are irrelevant to target concepts [43].

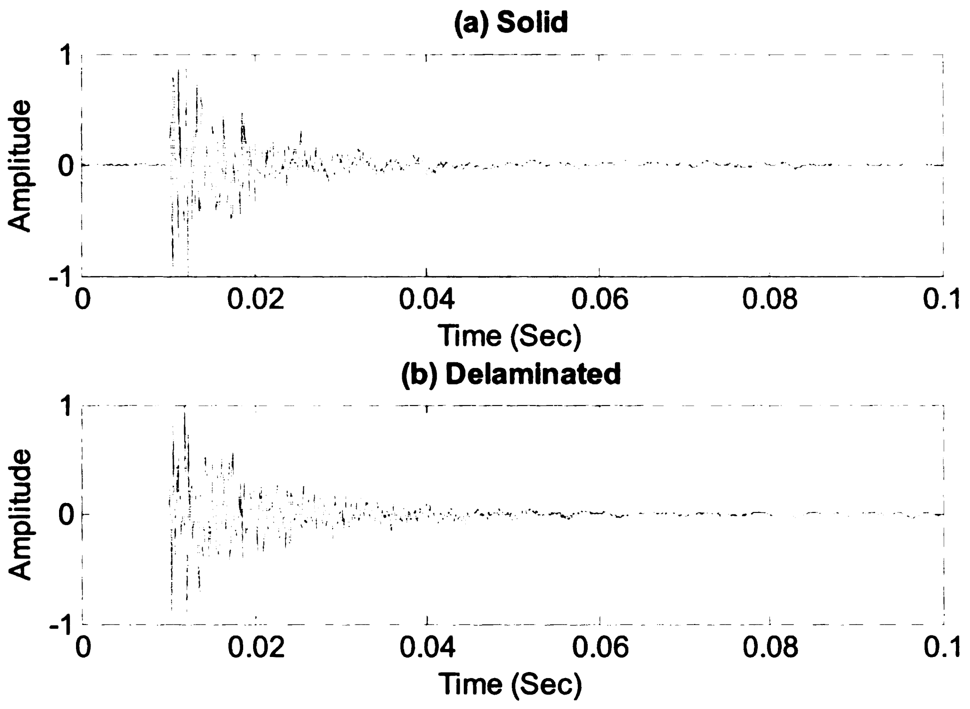


Figure 4.1 Example Impact Signals

Acoustic signals are usually parameterized using different models. The parameters in these models are called features of the signal and the process of parameterization is called feature extraction. Different models represent a signal in different ways and extract different features of the signal. For example, the Fourier Transform (FT) expresses the signal in the frequency domain and extracts frequency features of the signal while the Wavelet Transform represents the signal in the wavelet domain and extracts different features of the same signal. Several features for acoustic signals are described in the following sections.

4.1.2 Sub-band Energy

Frequency components are probably the most widely used features in the processing of acoustic signals since they have a clear physical meaning. To obtain the features in the frequency domain, the Fourier Transform (FT) that represents the signal in the forms of sinusoids with different frequencies is used. Since the signal used for detection is digitized and discrete, the discrete Fourier transform (DFT) is used. The computation of DFT is described below:

$$X_k = \sum_{n=0}^{N-1} x_n \exp\left(-\frac{2\pi i}{N}kn\right), \ k = 0,1,\ldots,N-1 \tag{4.1}$$

where $N/2+1$ is the number of discrete frequencies.

Upon taking the DFT of the signal, the signal is represented by frequency domain features and the dimension of the feature vectors is N. In order to reduce the dimension of the feature vectors, N should be small. However, a small N leads to a decrease of resolution in the frequency domain. A bigger N yields higher resolution in the frequency domain, but this increases the dimension of the feature space and defeats the purpose of dimension reduction. Also, due to the short duration of the impact signal used for delamination detection, the variance of the DFT can be large. Sub-band energy can be used to reduce the dimension and the variance of the frequency domain features. The entire frequency domain is evenly divided into several (for example, 20) sub-bands and the energy in each sub-band is calculated as follows and used as features for delamination detection:

$$E_i = \sum_{\omega \in F_i} |X(\omega)|^2 \tag{4.2}$$

This is equivalent to passing the signal through a different series of band-pass filters with different cut-off frequencies. This filter series is called a filter bank. The shape of the filter bank in this case is rectangular and the filters are evenly spaced on the frequency axis, as shown in Figure 4.2. The energy of the filtered signal is extracted as features.

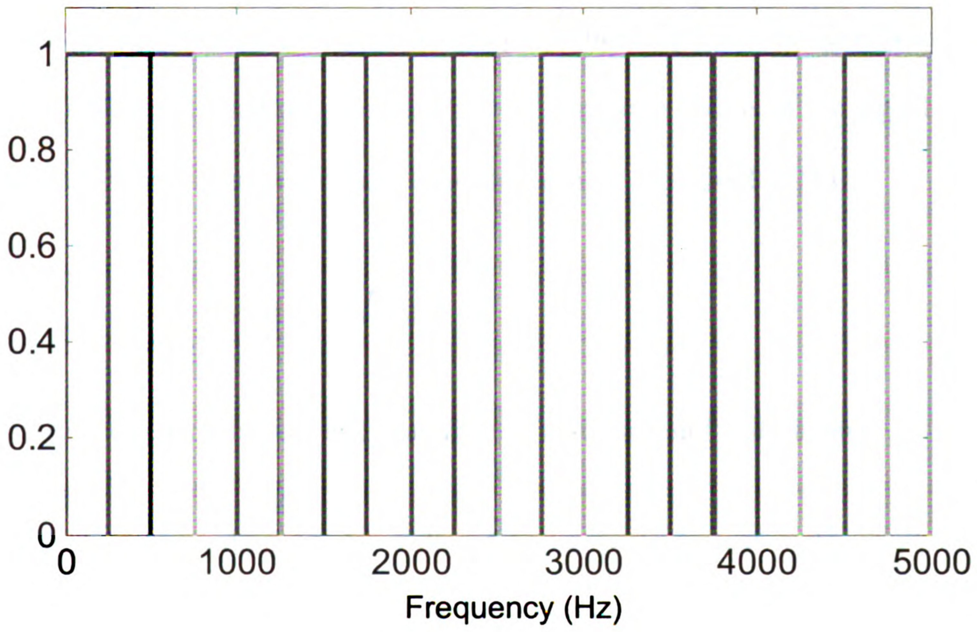


Figure 4.2 Rectangular Filter Bank

4.1.3 Energy of Wavelet Packet Tree

Another set of commonly used features in signal processing are obtained in the wavelet domain. Similar to the Fourier Transform, the Wavelet Transform (WT) decomposes the signal using different basis functions. The difference between the WT and the FT is in the selection of the basis functions. In the Fourier Transform, the basis functions are a family of sinusoids with infinite support in the time domain, while the basis functions for WT are scaled and shifted versions of wavelet functions, usually with a finite support. The wavelet transform provides more flexibility in choosing the type of basis functions. In addition, the short support of the basis functions can capture transient information about the signal. Also the scaling and shifting of basis functions make the wavelet representation capable of representing the signal at different resolution levels in both the time and frequency scales. The wavelet is defined by two bases, the scaling function and the wavelet function. The scaling function captures the base shape of the signal and the wavelet function is responsible for capturing details of the signal. Having defined the basis functions, the wavelet transform can be expressed as [44]:

$$X(a,b)=\frac{1}{\sqrt{a}}\int_{-\infty}^{+\infty} x(t)\psi\left(\frac{t-b}{a}\right)dx \tag{4.3}$$

where $\psi(\cdot)$ is the basis function, and a and b are the scaling and shift factors.

In the frequency domain, the scaling function is equivalent to filtering the signal through a low pass filter and the wavelet function is a high pass filter. The traditional wavelet transform decomposes the signal into approximation and detail coefficients by the scaling and wavelet functions, respectively. The approximation coefficients are further decomposed by the scaling and the wavelet functions at the second level. This process continues until the required level of decomposition is reached. In this way, the signal is expressed by the approximation coeffeicients and detail coefficients at different levels. The process of the wavelet transform is shown in Figure 4.3.

$$\begin{tikzcd}
 & & & S \arrow[dl] \arrow[dr] & \\
 & & CA1 \arrow[dl] \arrow[dr] & & CD1 \\
 & CA2 \arrow[dl] \arrow[dr] & & CD2 & \\
CA3 & & CD3 & &
\end{tikzcd}$$

Figure 4.3 Wavelet Decomposition

The wavelet decomposition is complete, meaning that there is no redundant information. This completeness of the wavelet transform is good for signal representation but may not be good for feature extraction. The purpose of the feature extraction is to find the features for classification purposes. Redundancy in this case provides flexibility in the selection of features. The wavelet packet is a redundant way of representing the signals and therefore is used in this research. The wavelet packet decomposition and the wavelet transform is the same except that in each step, both the approximation coefficients and the detail coefficients are decomposed. This process is equivalent to passing both approximation and detail coefficients from the previous level through a high pass and a low pass filter. The shape or the frequency response of each filter is dependent on the type of wavelet function used. Figure 4.4 [45] shows an example level 3 wavelet packet decomposition. After the signal was decomposed, the Shannon entropy of each sub-band at the lowest level in the wavelet packet tree can be used as features.

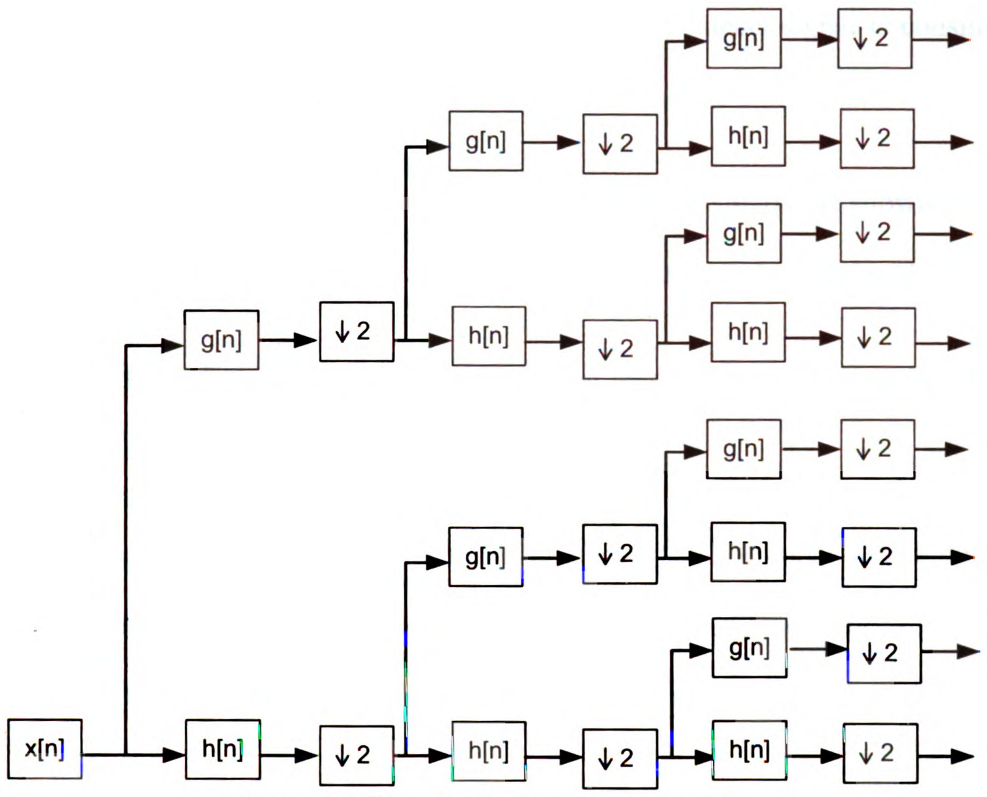


Figure 4.4 Wavelet Packet Decomposition

4.1.4 Psycho-Acoustic Features

Even though detection of delamination by the "hollowness" of the impact signal may be subjective, it is undeniable that the human ear is good at detecting differences in sounds. To understand how the human ear discriminates sound, research has been conducted in the field of psycho-acoustics [46]. Many feature extraction algorithms based on psycho-acoustic models have been proposed. Mel-Frequency Cepstral Coefficients (MFCC) [47] is one of the psycho-acoustics based features. MFCC approximates the human auditory system's response more closely than the linearly-spaced frequency bands.

MFCC can be calculated as follows:

Step 1: Split signal into frames; in this case, each impact signal is considered a frame.

Step 2: For each frame, smooth with the Hamming window as shown in Figure 4.5 and then compute the fast Fourier transform (FFT) of the windowed signal.

Step 3: Calculate the power spectrum of the framed signal by squaring the FFT spectrum.

Step 4: Filter the power spectrum obtained in Step 3 through a Mel-frequency filter bank. The filters are equally distributed on the Mel-scale. The relationship between the Mel-scale frequency and regular frequency measurement (Hz) is:

$$m = 1027.01048 \log_e \left(1 + \frac{f}{700}\right) \tag{4.4}$$

where m is the Mel-frequency and f is in Hz.

Different filter shapes may be used. In this study, the triangular shape is selected as shown in Figure 4.6 . Adjacent filters are overlapped over half of the bandwidth.

Step 5: Apply the discrete cosine transform (DCT) to the log of the spectrum filtered by the Mel-frequency filter banks. The DCT can be calculated as:

$$DCT_x(k) = w(k) \sum_{n=1}^{N} x(n) \cos \frac{\pi(2n-1)(k-1)}{2N} \text{ , } k = 1, \ldots, N \tag{4.5}$$

where: $w(k) = \begin{cases} 1/\sqrt{N}, & k = 1 \\ \sqrt{2/N}, & 2 \le k \le N \end{cases}$

Step 6: Obtain the MFCC by applying a filter to the output of the DCT (to smoothen the MFCC). In this case, the filter has a half sine wave shape.

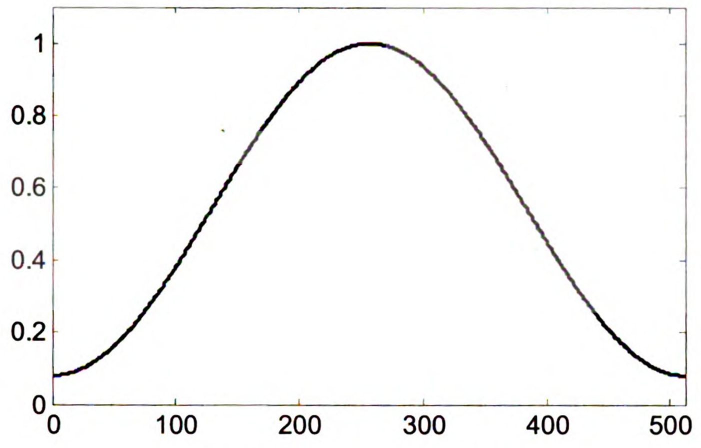


Figure 4.5 Hamming Window

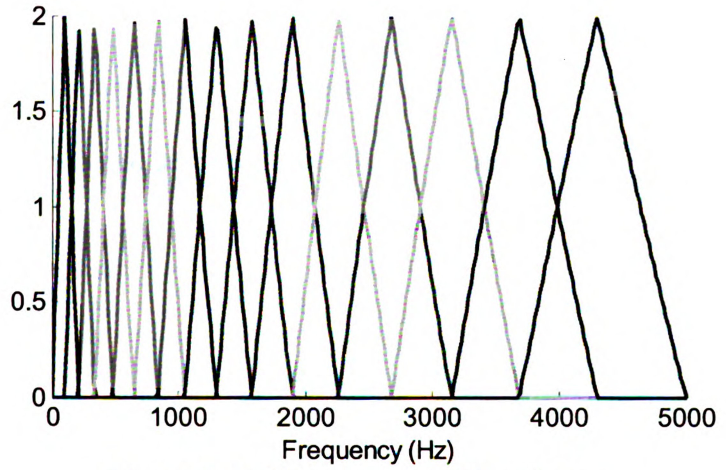


Figure 4.6 Mel-Frequency Filter Banks

There are different variations of the MFCC. For example, other types of frequency scales such as the Bark scale [46] which ranges from 1 to 24 Barks, corresponding to the first 24 "critical bands" of the human hearing system based on the results of psychoacoustic experiments, can be used. In addition to the different frequency scales, the shape of the filter bank can be different to optimize the performance of the feature extraction algorithms. For example, the shape of the filter bank at different frequencies can be optimized by Principal Component Analysis [48].

4.1.5 Principal Component Analysis

When the data is transformed from the "data space" to the "feature space", it is desirable that the original data is represented by "effective" features in a lower dimension with minimal loss. Energy loss is one commonly used cost function during dimension reduction. A good representation will keep as much energy as possible using the least number of features. Principal component analysis (PCA) [48] can be used to achieve this goal. It finds the optimal linear transformation such that the extracted features are the best representation of the original signal in the sense of mean square error.

Assume the dimension of the original data x is n, and that the data can be represented by n orthogonal unit basis vectors u_i, where $i = 1, \ldots, n$, as:

$$x = y_1 u_1 + \ldots + y_n u_n \tag{4.6}$$

Therefore:

$$y_j = x^T u_j \tag{4.7}$$

The extracted features can be found by truncating Equation (4.6) to:

$$\hat{x} = y_1 u_1 + \ldots + y_m u_m \tag{4.8}$$

where m is the dimension of the features and $m < n$.

The error between the original data and the extracted features is found to be:

$$\begin{aligned}\varepsilon &= x - \hat{x} \\ &= \sum_{j=m+1}^{n} y_j u_j\end{aligned} \tag{4.9}$$

The mean square error is:

$$\begin{aligned}MSE &= E\left(\varepsilon^2\right) \\ &= E\left[\left(\sum_{i=m+1}^{n} y_i u_i\right)^T \left(\sum_{j=m+1}^{n} y_j u_j\right)\right] \\ &= \sum_{i=m+1}^{n} E\left(y_j^2\right)\end{aligned} \tag{4.10}$$

where $E(\cdot)$ is the expectation operation and

$$\begin{aligned}E\left(y_j^2\right) &= E\left[\left(u_j^T x\right)\left(x^T u_j\right)\right] \\ &= u_j^T R u_j\end{aligned} \tag{4.11}$$

where R is the correlation matrix of x.

Substituting Equation (4.11) into (4.10) yields:

$$MSE = \sum_{j=m+1}^{n} u_j^T R u_j \tag{4.12}$$

where $u_j^T u - 1 = 0$

The minimization of the mean square error can be found by using a set of Lagrangian multipliers and setting the derivative of the mean square error to zero:

$$\frac{\partial \xi}{\partial u_j} = \frac{\partial}{\partial u_j}\left[\sum_{j=m+1}^{n} u_j^T R u_j - \lambda\left(u_j^T u - 1\right)\right] \tag{4.13}$$
$$= 2Ru_j - \lambda u_j = 0$$

It can be shown that u_j is the eigenvector of the correlation matrix R. The mean square error can then be found to be:

$$MSE = \sum_{j=m+1}^{n} \lambda_j \tag{4.14}$$

where λ_j are the eigenvalues of the correlation matrix R.

To minimize the mean square error, λ_j needs to be the smallest $n-m$ eigenvalues. Hebbian learning algorithms [36] can be used to find the eigenvectors and eigenvalues of the correlation matrix. The advantage of PCA is that it is a non-parametric analysis and the result is unique and independent of any assumption about the probability distribution of the data.

The procedure of extracting features using PCA is briefly described below:

Step 1: Obtain principal components of the training signals.

Step 2: Project the signals onto the principal components and use the projections as features of the signal. In this way, the signal is represented by the least amount of projections.

4.1.6 Independent Component Analysis

Independent Component Analysis (ICA) described in Chapter 3 can also be used to find the effective features of a signal. Unlike PCA which maximize the energy contained in the extracted features, ICA extracts features by maximizing the amount of information. In this approach, the loss of information is minimized in the process of dimension reduction. The procedure of feature extraction using ICA [49] is briefly described as follows:

Step 1: Extract independent components from the training signal.

Step 2: Select dominant independent components by relative importance of basis vectors. In this case, the L_2 norm of the column of the de-mixing matrix is used as the criteria for relative importance.

Step 3: Transform the dominant ICs into the frequency domain. Signals were filtered by the filters obtained from the dominant ICs.

Step 4: Scale the energy of the filtered signals logarithmically and compute the cepstral coefficients of the log-energy as features.

This method is similar to the MFCC described in Section 4.1.4, except that the shape of the filter bank is computed from the dominant ICs.

4.2 Performance of Different Features

Several commonly used feature extraction algorithms were described in the previous section. Different algorithms extract different features of the signal. In order to select the best features for the detection of concrete delamination, it is necessary to evaluate and compare the results of different feature extraction algorithms. This section first introduces the criteria for the evaluation of different feature extraction algorithms and then the performance of the algorithms is evaluated against these criteria.

4.2.1 Criteria for Evaluation

For acoustic methods of concrete delamination detection, the features of the acoustic signal need to have two properties. First, the features must be repeatable, i.e., the features of the signal obtained under the same test conditions must be consistent. Second, the features of the signal must be separable, i.e., the difference between features from the solid and delaminated concrete must be large so that they can be easily separated from one another. The evaluation criteria must be numerical and dimensionless so that an objective comparison can be made.

Assuming that the features from solid and delaminated concrete are random variables, repeatability can be measured by the coefficient of variation. For multiple random variables, the repeatability of the extracted feature can then be calculated as the weighted mean value of the coefficient of variation for solid and delaminated concrete:

$$RPT = \sqrt{\frac{(N_S - 1)\left(\frac{\sigma_S}{\mu_S}\right)^2 + (N_D - 1)\left(\frac{\sigma_D}{\mu_D}\right)^2}{N_S + N_D - 2}} \tag{4.15}$$

where μ_S and μ_D are the mean values of the extracted features for signals from solid and delaminated concrete; σ_S and σ_D are the standard deviations of the features for solid and delaminated signals; N_S and N_D are the number of samples in the groups of solid concrete and delaminated concrete, respectively. A high repeatability value indicates poor repeatability of the test.

The separability of the features can be formulated as a hypothesis test: whether the features from solid concrete and from delaminated concrete has the same mean. The t statistic-based separability measure is:

$$SEP = \frac{\left|\mu_S - \mu_D\right|}{\sqrt{\frac{(N_S - 1)\sigma_S^2 + (N_D - 1)\sigma_D^2}{N_S + N_D - 2}}} \tag{4.16}$$

A higher separability measurement indicates better separation.

Another separability criterion comes from information theory: the mutual information. Each feature contains a certain amount of information about the type of the concrete and mutual information is one way of measuring the amount of information. In delamination detection, it measures the information about the type of concrete or class label. The computation of this information theoretic measurement is briefly described below [50].

Suppose the type of concrete (or class label) is a random variable. The uncertainty of the class label can be calculated as:

$$H(C) = -\sum_{c} P(C)\log\left(P(C)\right) \tag{4.17}$$

where $P(C)$ is the probability density function (PDF) of the class label C.

After observing a feature vector $\underline{x}$, the conditional entropy becomes:

$$H(C\mid\underline{x}) = -\int_{\underline{x}} p(\underline{x})\left(\sum_{c} p(C\mid\underline{x})\log\left(p(C\mid\underline{x})\right)\right)d\underline{x} \tag{4.18}$$

where $p(\underline{x})$ is the PDF of the feature vector $\underline{x}$ and $p(C\mid\underline{x})$ is the conditional PDF of $\underline{x}$ given C.

The loss of uncertainty after the observation of a feature is called the mutual information between the feature and the class label and can be calculated as:

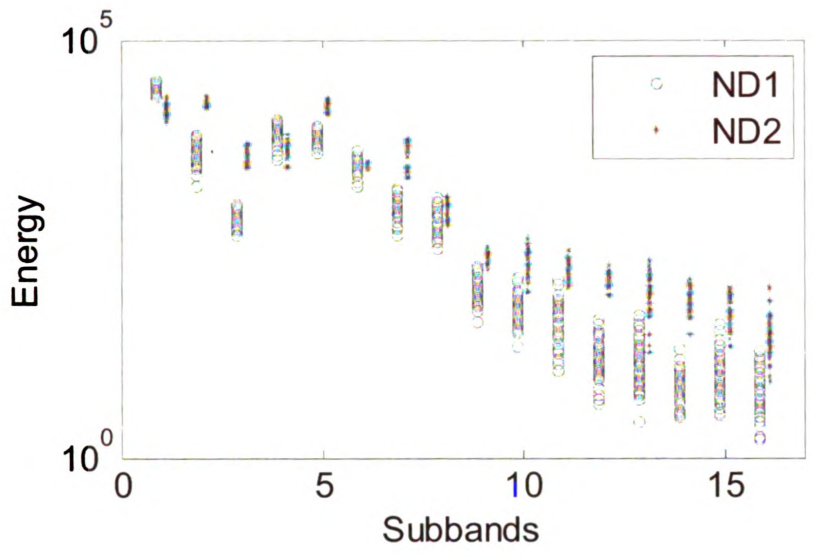
$$\begin{aligned} I(C,\underline{x}) &= H(C) - H(C\mid\underline{x}) \\ &= -\sum_{c}\int_{\underline{x}} p(c,\underline{x})\log\left(\frac{p(c,\underline{x})}{P(c)p(\underline{x})}\right)d\underline{x} \end{aligned} \tag{4.19}$$

In the actual computation, the probability distribution functions are obtained directly from the available samples and estimated from the histogram of each variable. The larger the value of mutual information, the more information is contained in the feature.

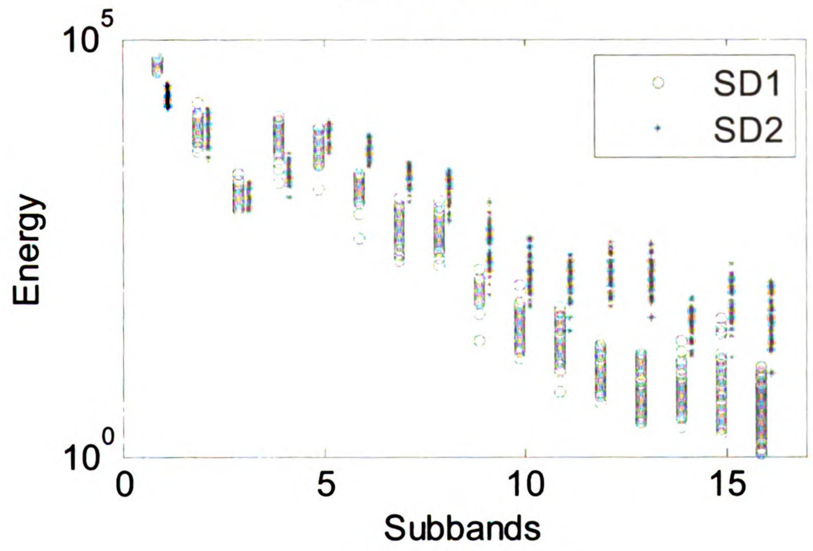
To test the performance of different features, data from lab experiments were used. The test setup and experimental process is discussed later in Chapter 6. The same experiments were performed on two different days to check the repeatability and to make the detection results more representative. On the first day, 53 impact signals were obtained from non-delaminated (ND1) or solid concrete and 66 impacts were recorded from a shallow delamination (SD1). On the second day, 52 impacts on solid concrete (ND2) were recorded and 66 impacts were from delaminated concrete (SD2). Therefore, there were 105 impact signals from solid concrete (ND) and 132 from delaminated concrete (SD). The measures of repeatability and separability were calculated for these signals.

4.2.2 Performance of Sub-band Energy

The performance of sub-band energy as a candidate feature was evaluated by repeatability and separability measures using the signals mentioned in the beginning of the section. Figure 4.7(a) compares the repeatability of the sub-band energy of signals from non-delaminated signals (ND1 vs. ND2) and Figure 4.7(b) compares the signal from delaminated signals (SD1 vs. SD2). Figure 4.8 shows the difference between the non-delaminated signals and the shallow delaminated signals. Due to the wide range of the feature values, the vertical axis (sub-band energy) was plotted on a log scale. As can be seen from Figure 4.7, the repeatability for signals obtained on the same day was good while the repeatability between days was not so good: the data obtained on different days forms two clusters for most of the sub-bands. From Figure 4.8, it can also be observed that the energy was mixed for all sub-bands, indicating poor separability.



(a) ND1 vs. ND2



(b) SD1 vs. SD2

Figure 4.7 Repeatability of the Sub-band Energy

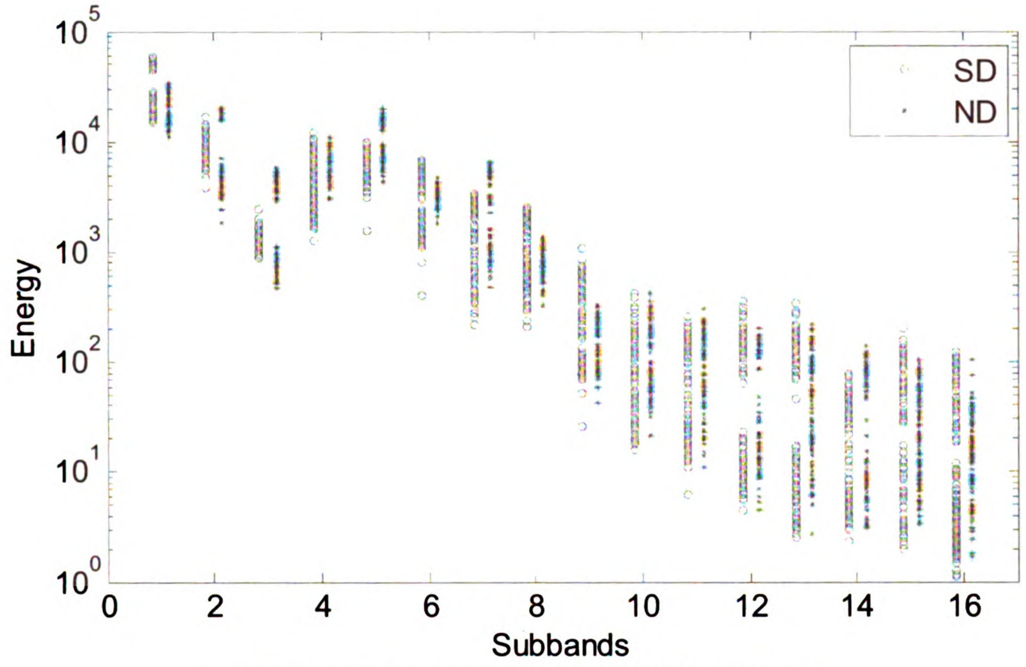


Figure 4.8 Separability of the Sub-band Energy

To quantitatively compare each individual feature, the numerical results of repeatability, separability and the mutual information between the class labels for different features were calculated and are shown in Figure 4.9 to Figure 4.11. Several observations can be made. First, different features have different separability and mutual information. The existence of the features that cannot effectively separate the two groups will lead to a decrease in the accuracy of the detection. It is therefore necessary to select the features that are useful for detection or classification purposes. Second, even though separability and mutual information measure the ease of separating the no delamination case from the delaminated case, they are inconsistent for some sub-bands. For example, the separability measure is very small for the first three sub-bands indicating poor repeatability, while the value of the mutual information for these sub-bands is high. The reason for this is that the separability measure provides the "distance" between the two classes while the mutual information measures the amount of the information.

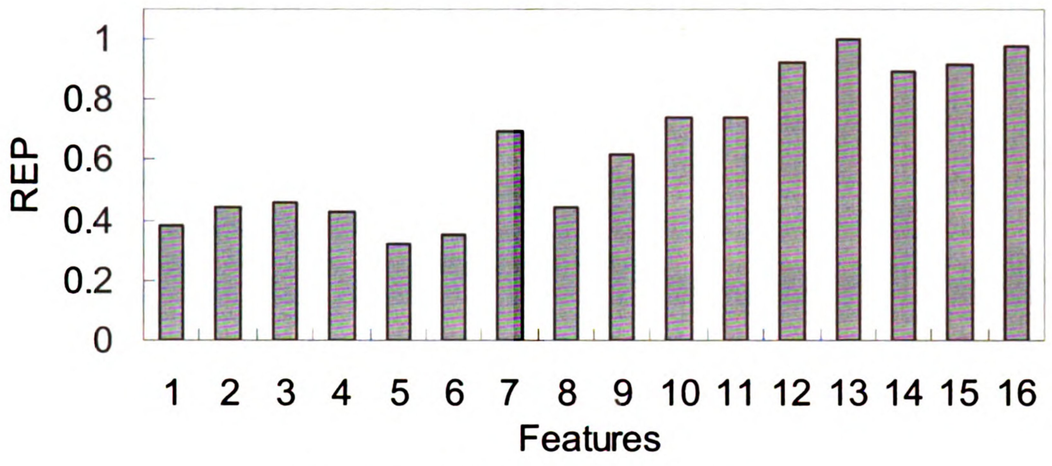


Figure 4.9 REP of the Sub-band Energy

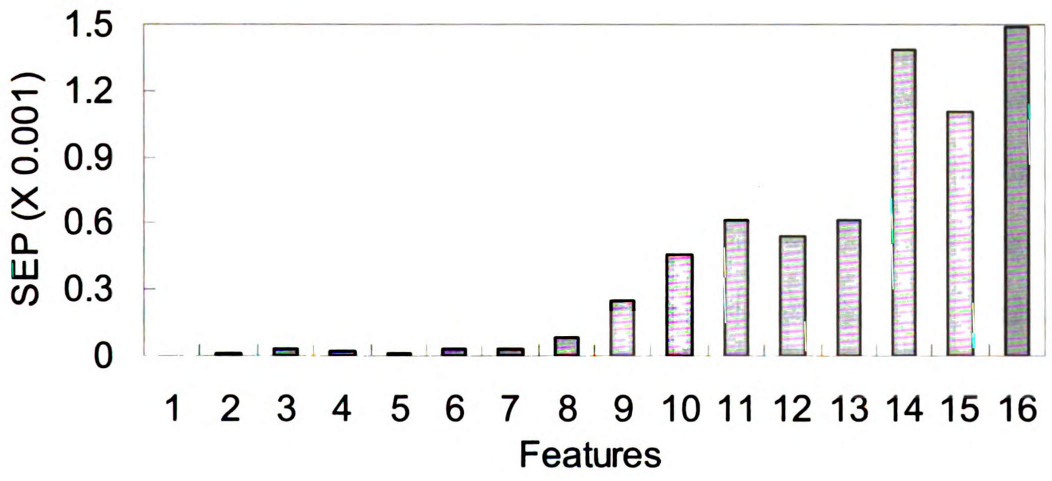


Figure 4.10 SEP of the Sub-band Energy

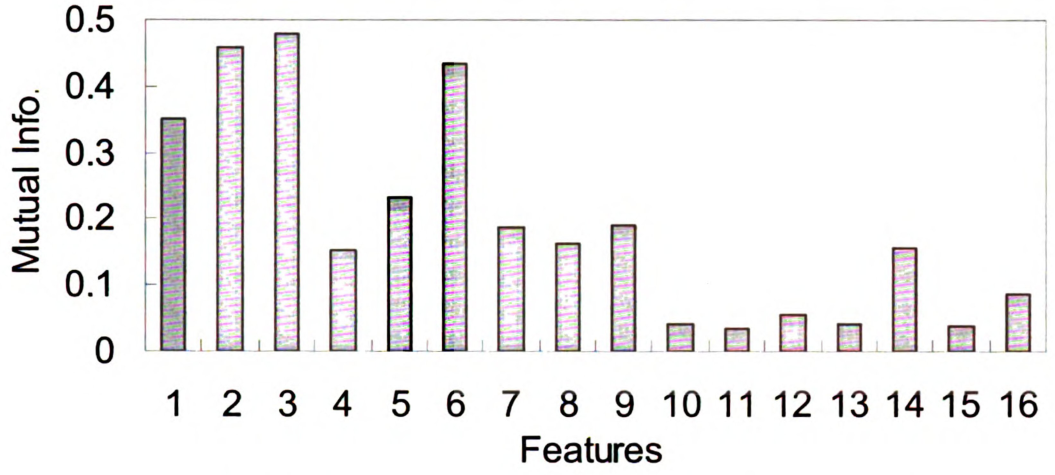
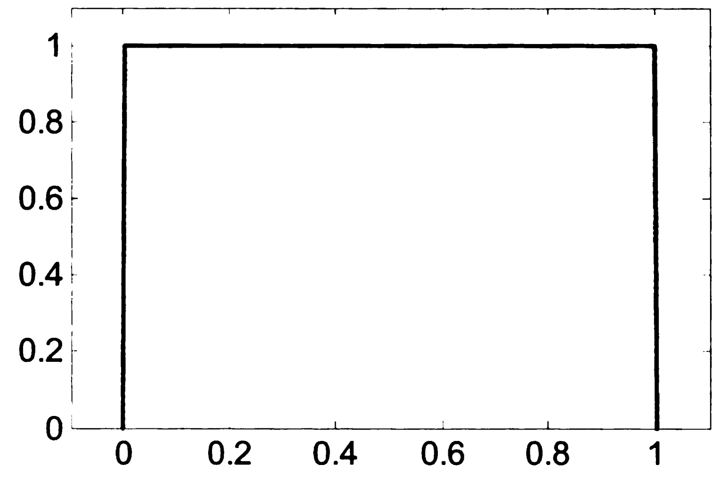


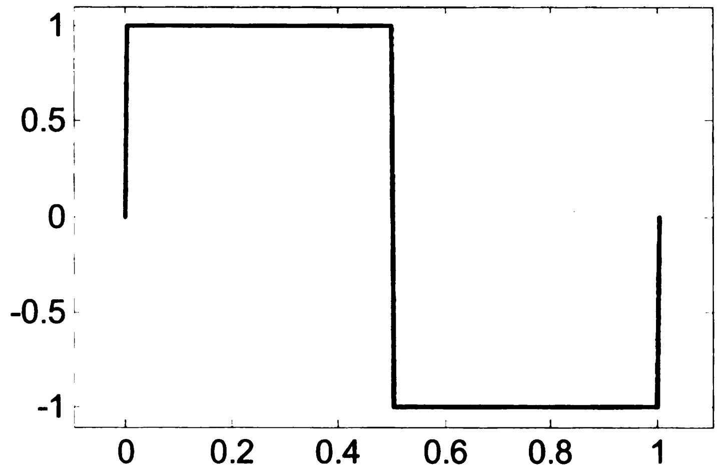
Figure 4.11 Mutual Information of the Sub-band Energy

4.2.3 Performance of the Wavelet Packet Tree

The performance of the wavelet packet decomposition is evaluated in this section. For simplicity, the HAAR wavelet is used. The shapes of the scale and wavelet functions for the HAAR wavelet are shown in Figure 4.12. The signals are decomposed to a level of 4 and the energy of 16 sub-bands are extracted as features. Figure 4.13 shows the repeatability of the features based on level 4 HAAR wavelet packet decomposition. Similar to sub-band energy, data points obtained on different days forms two clusters indicating good repeatability on the same day but poor repeatability between different days. The separability (shown in Figure 4.14) is very poor for energy of all sub-bands, indicating that the sub-band energy of level-4 HAAR wavelet packet decomposition is not a good option for differentiating signals from delaminated concrete and those from non-delaminated concrete.

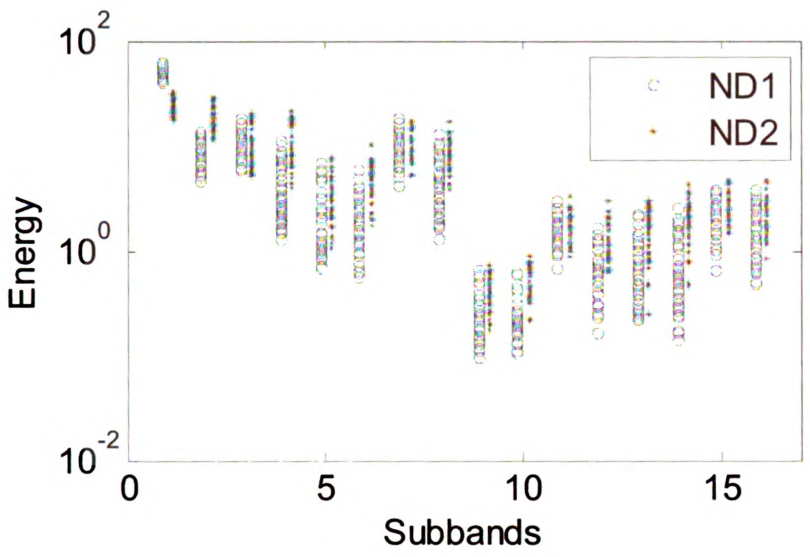


(a) Scaling Function

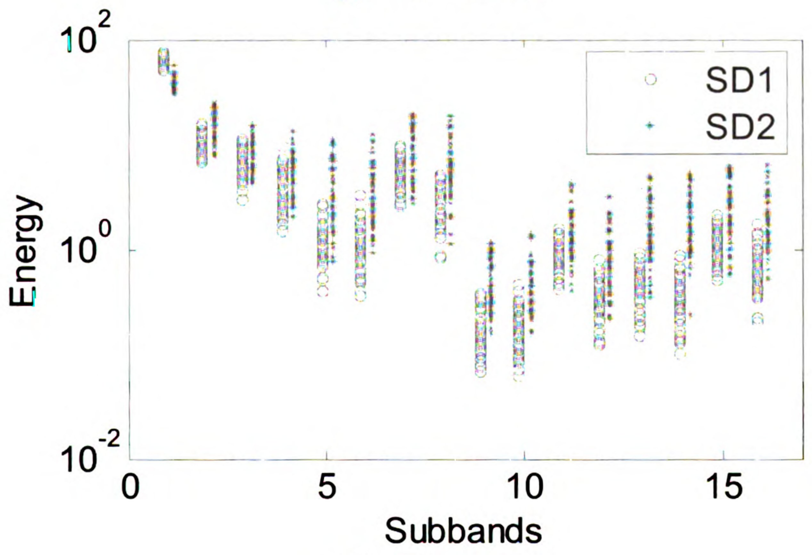


(b) Wavelet Function

Figure 4.12 HAAR Wavelet



(a) ND1 vs. ND2



(b) SD1 vs. SD2

Figure 4.13 Repeatability of the WP Tree

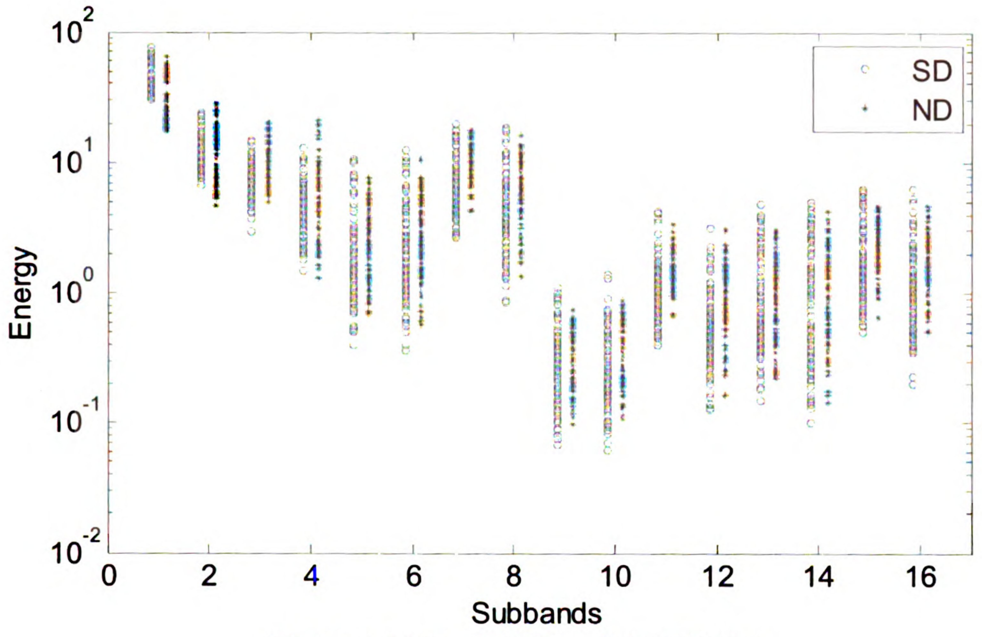


Figure 4.14 Separability of the WP Tree

The numerical performance measures are plotted in Figure 4.15 to Figure 4.17 for comparison. Similar to sub-band energy, the repeatability of different branches was different. The SEP values of branches 9-16 were much higher than the first eight branches, but the value of the mutual information did not show this pattern, indicating that the mutual information and separability measures are inconsistent.

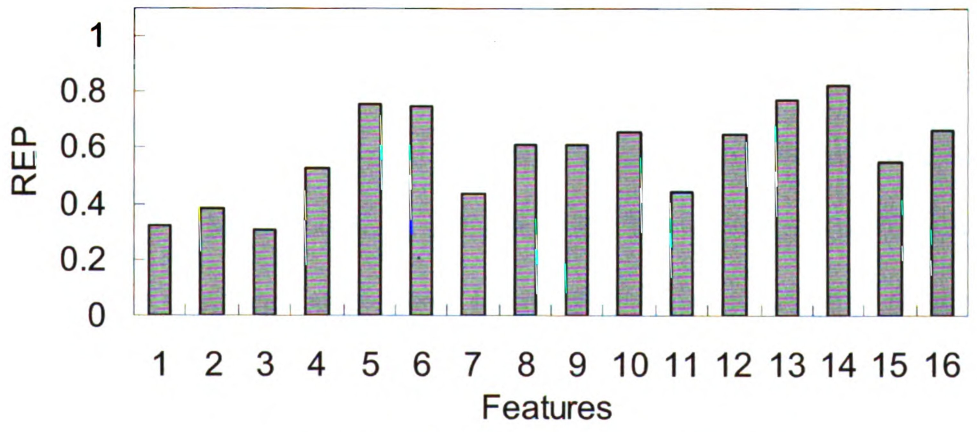


Figure 4.15 REP of the Wavelet Packet Tree

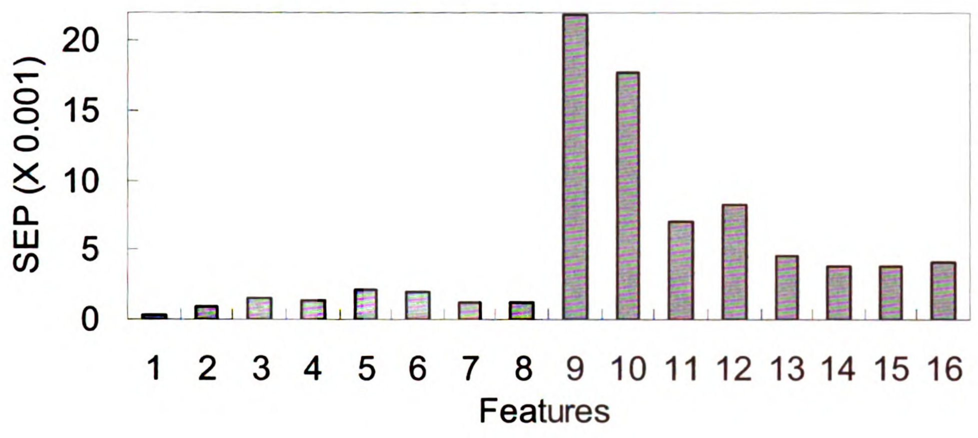


Figure 4.16 SEP of the Wavelet Packet Tree

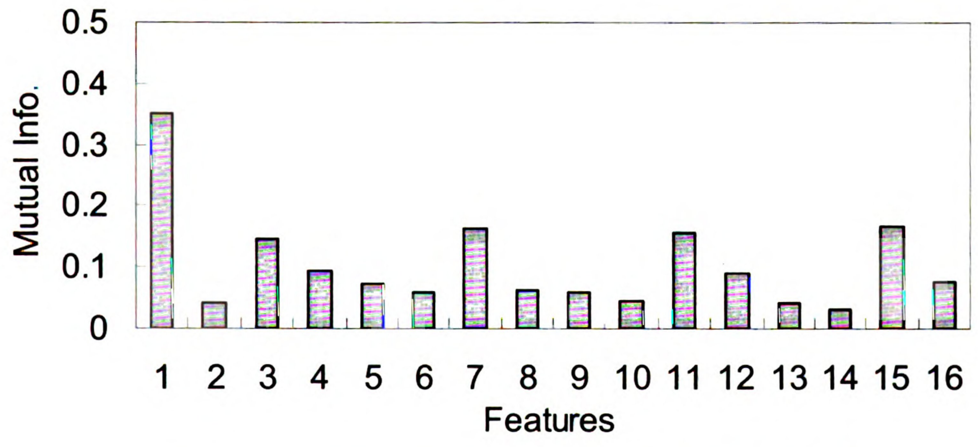
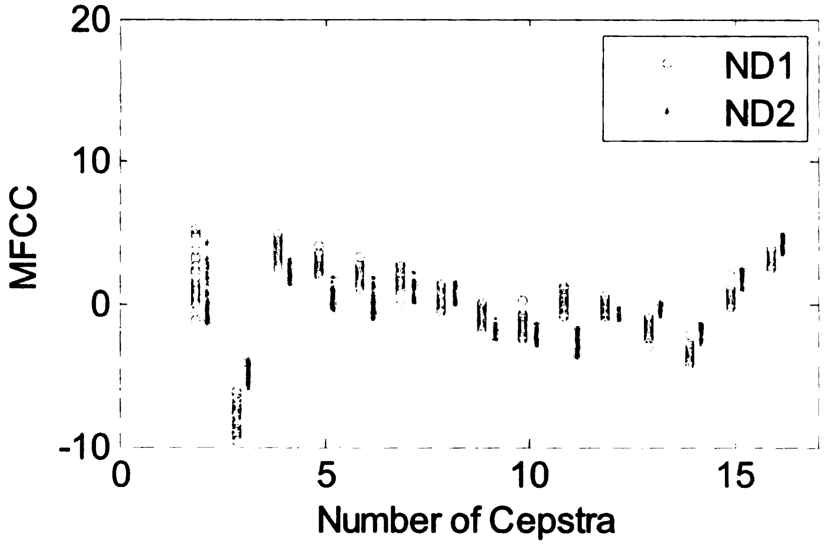


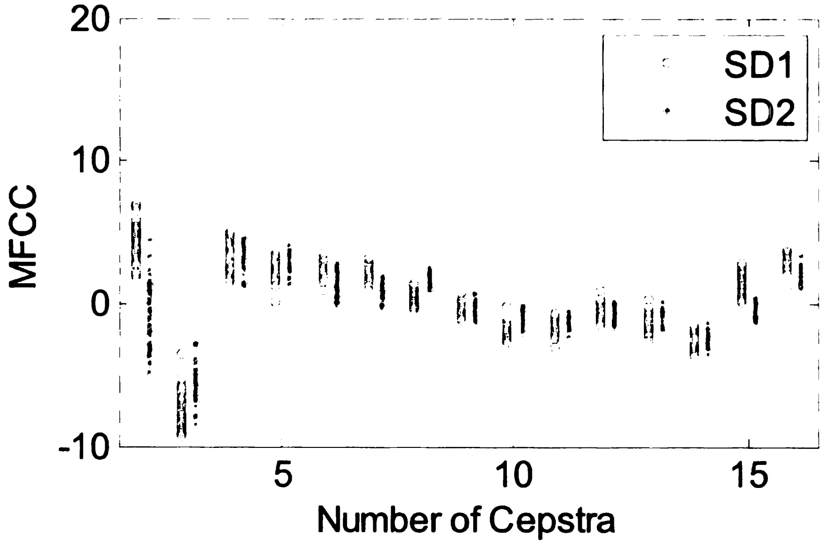
Figure 4.17 Mutual Information of the Wavelet Packet Tree

4.2.4 Performance of MFCC

In calculating the MFCC, a filter bank consisting of 50 triangular filters that are evenly spaced on Mel-scale was used. A total of 16 cepstra coefficients were computed. The repeatability of MFCC is shown in Figure 4.18. The variation of MFCCs was small across different days indicating good repeatability. The separability of the MFCC between the solid and delaminated concrete is shown in Figure 4.19. The difference between the solid concrete (ND) and delaminated concrete (SD) was small.



(a) ND1 vs. ND2



(b) SD1 vs. SD2

Figure 4.18 Repeatability of the MFCC

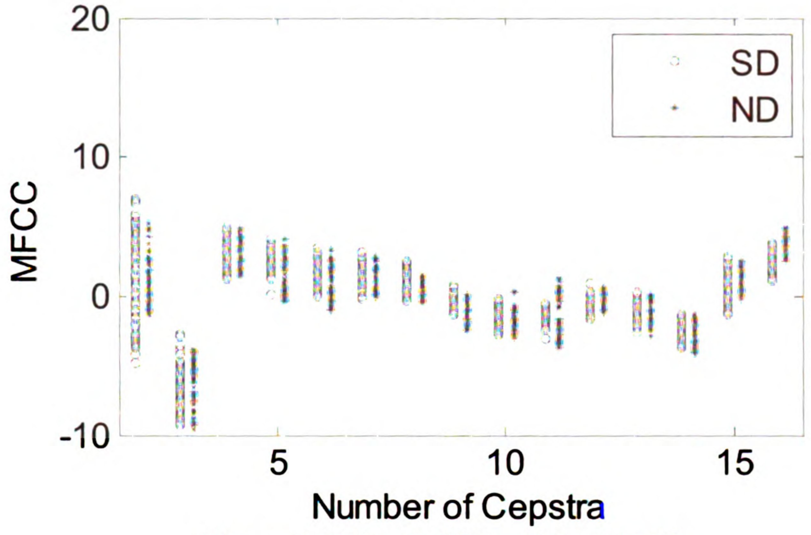


Figure 4.19 Separability of the MFCC

The numerical performance criteria for different coefficients are shown in Figure 4.20 to Figure 4.22. The high REP values of the second and fifteenth MFCC result from the high variation between different days, meaning that these features are not stable and may not be good choices for detection purposes. Inconsistency between SEP and mutual information measures also occurs for several features.

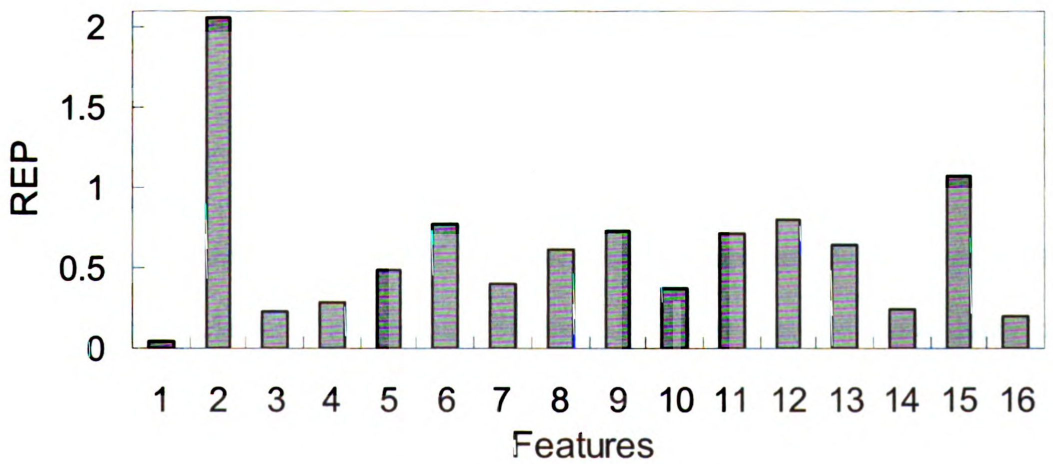


Figure 4.20 REP of the MFCC

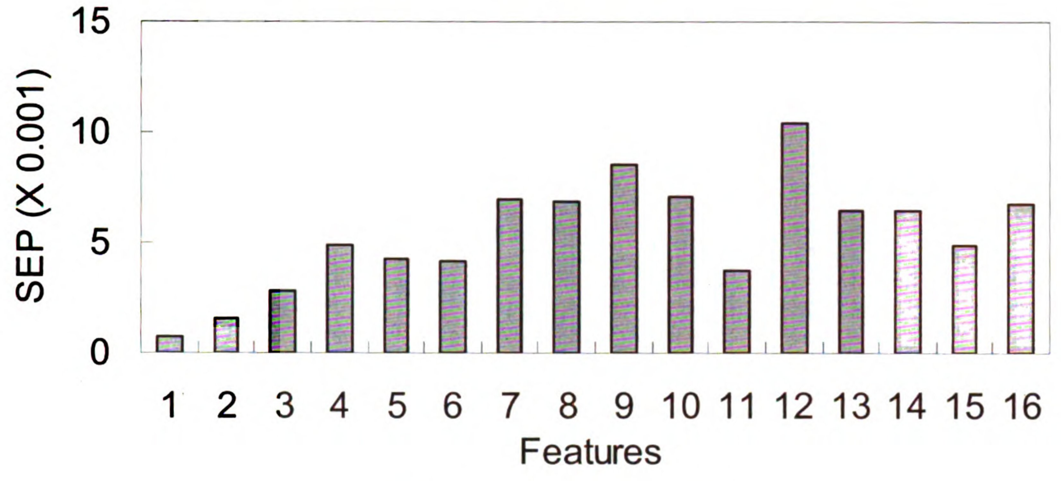


Figure 4.21 SEP of the MFCC

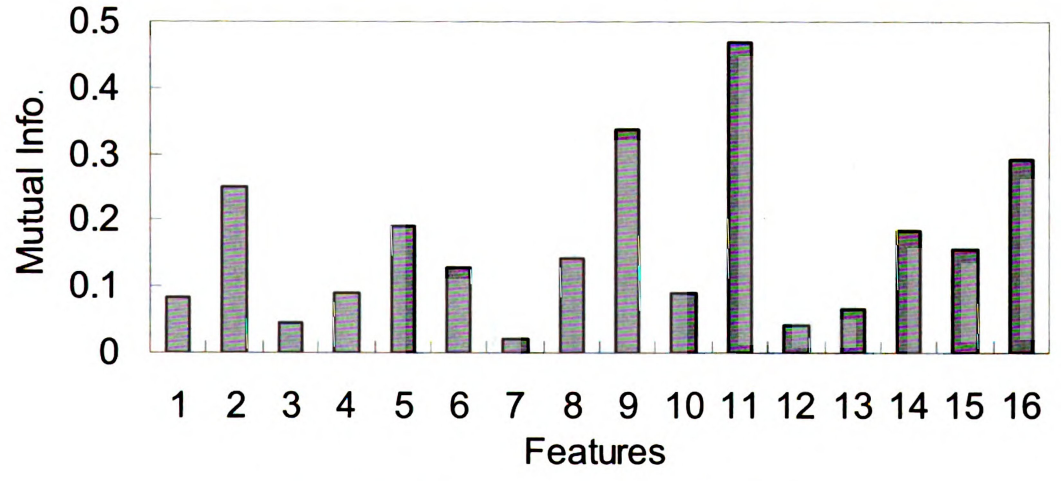
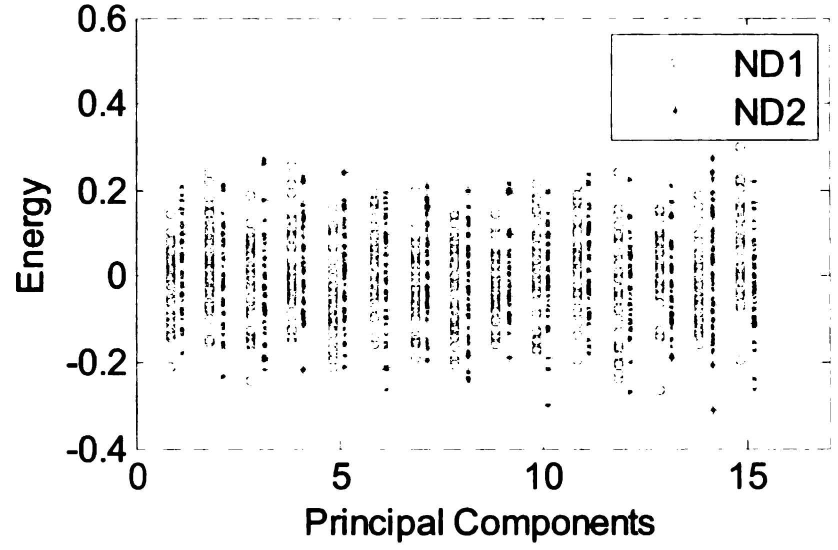


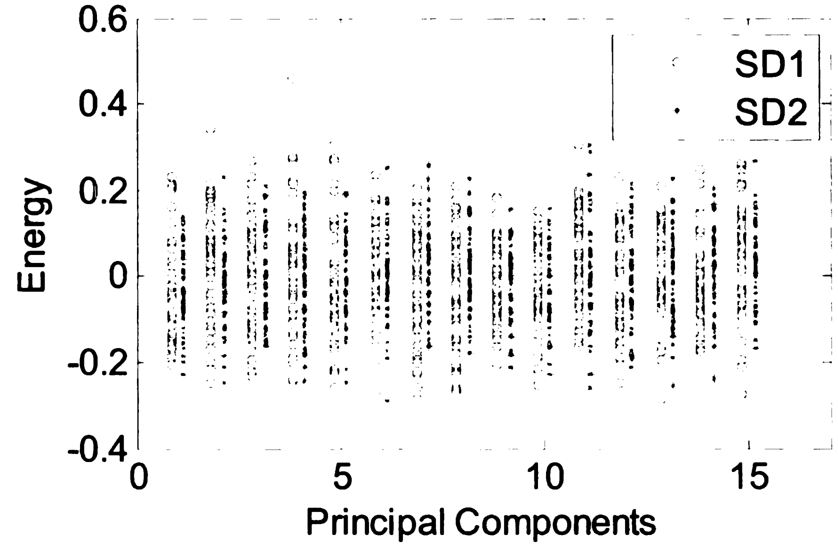
Figure 4.22 Mutual Information of the MFCC

4.2.5 Performance of Features Extracted by PCA

To check the performance of PCA, the first 16 dominant principal components of the signal were extracted as features. Figure 4.23 shows that the repeatability of the features was very good. However, the extracted features of the delaminated and solid signals had a large overlap as shown in Figure 4.24, indicating poor separability.



(a) ND1 vs. ND2



(b) SD1 vs. SD2
Figure 4.23 Repeatability of the PCA

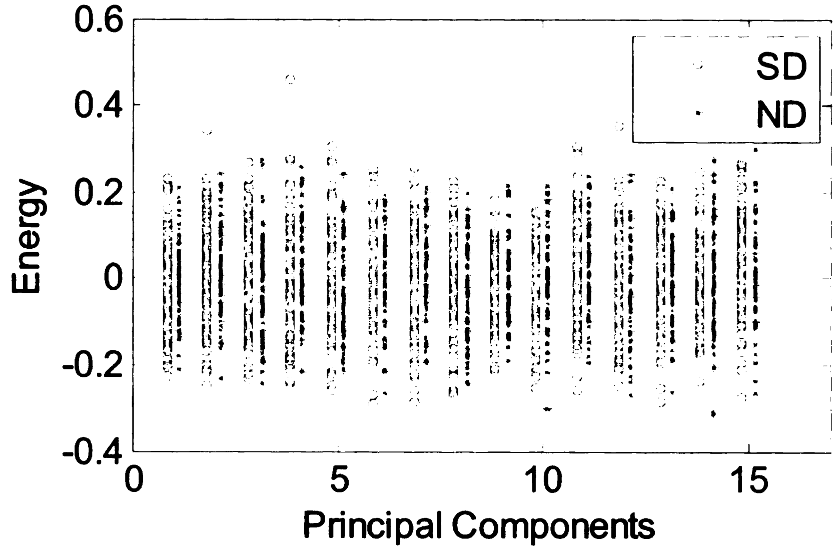


Figure 4.24 Separability of the PCA

The repeatability, separability and mutual information measures of the features extracted by PCA are shown in Figure 4.25 to Figure 4.27, respectively. The REP values of several features were too high and are not shown in Figure 4.25. Several conclusions can be observed here. The good repeatability indicated in Figure 4.23 was not supported by the numerical repeatability measure. The reason for this is that the absolute values of these features were close to zero and Equation (4.15) became ill-conditioned: a small variance may lead to a high repeatability measure. The separability of PCA was poor according to Figure 4.24, but the values of SEP are very high. This also results from the ill-conditioning problem of Equation (4.16) due to small variance of the features. The values of mutual information of most principal components were small indicating that these features did not contain much information about their class labels. The results were consistent with Figure 4.24. The number of principal components used was only 16 in this case and relataively small compared with the high dimension of the acoustic signal. Therefore, the amount of information contained in each features is not much.

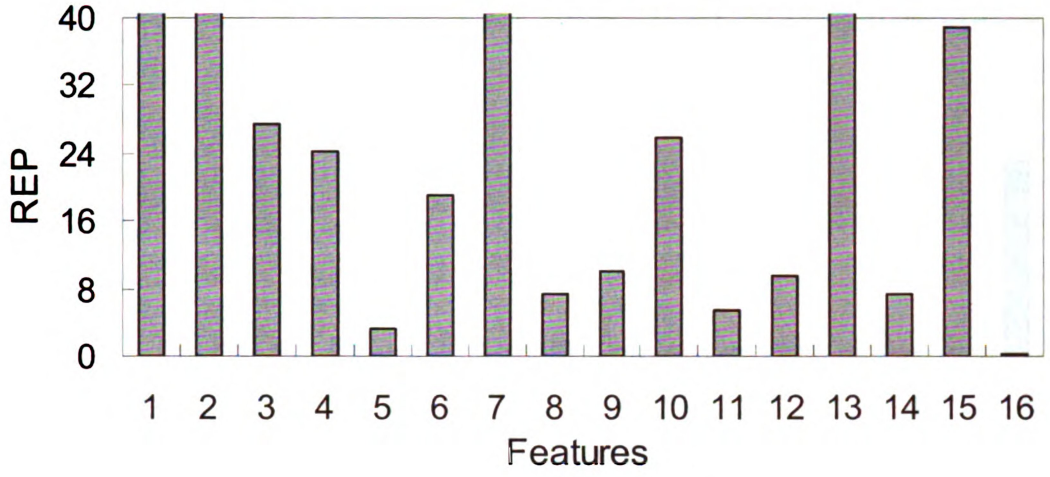


Figure 4.25 REP of the PCA

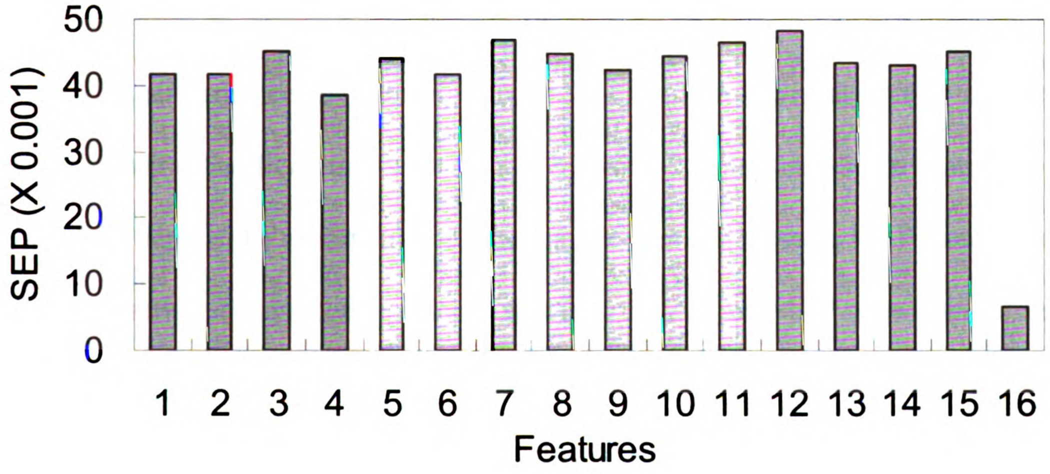


Figure 4.26 SEP of the PCA

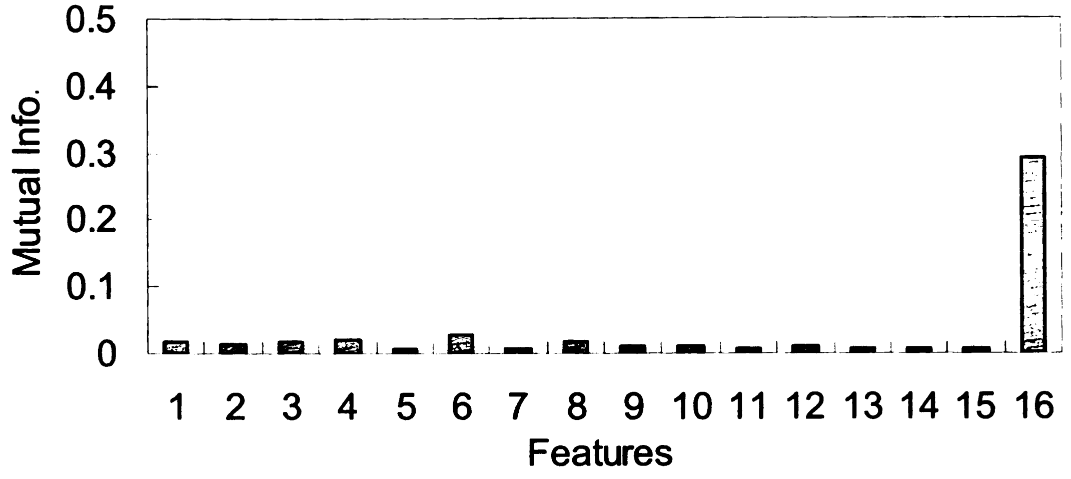
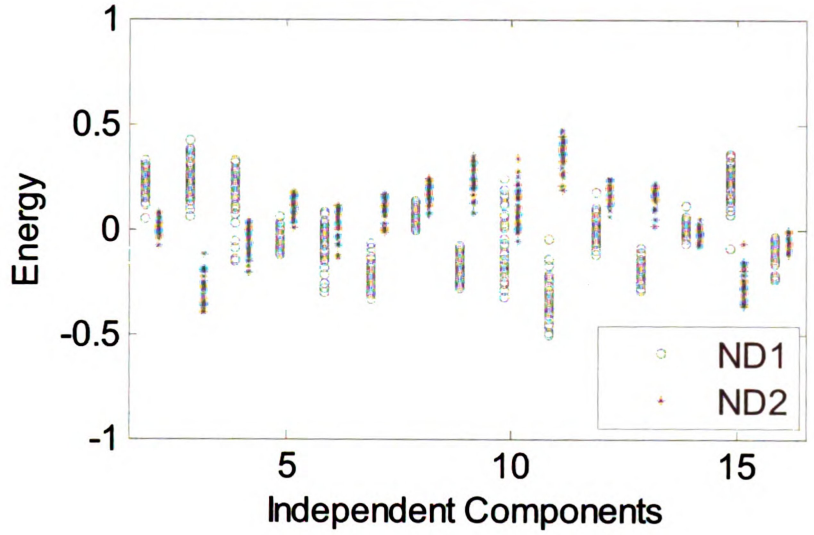


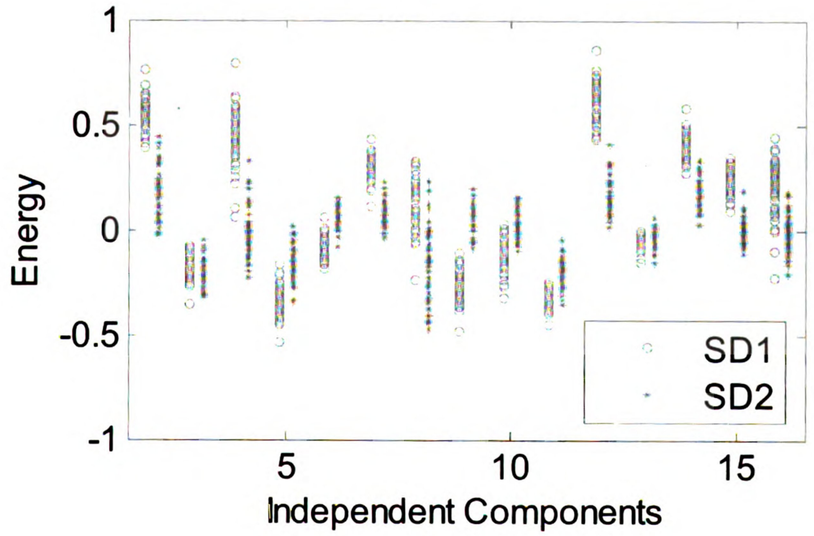
Figure 4.27 Mutual Information of the PCA

4.2.6 Performance of Features Extracted by ICA

The performance of features extracted by the ICA-based algorithm is discussed next. 16 cepstral coefficients were extracted by using 25 filter banks determined by ICA. Figure 4.28 depicts the repeatability of the features. As can be seen, the features were not repeatable between days but repeatable within a day. Figure 4.29 shows the difference between the signals from delaminated and solid concrete. The separation between the two types of signals is not clear. The reason for this is that the independent components of the input signal are computed by maximizing the independence of the output signals. In this process, no information about the deck condition or class label is considered, and features extracted by this method are indiscriminant.



(a) ND1 vs. ND2



(b) SD1 vs. SD2

Figure 4.28 Repeatability of the ICA

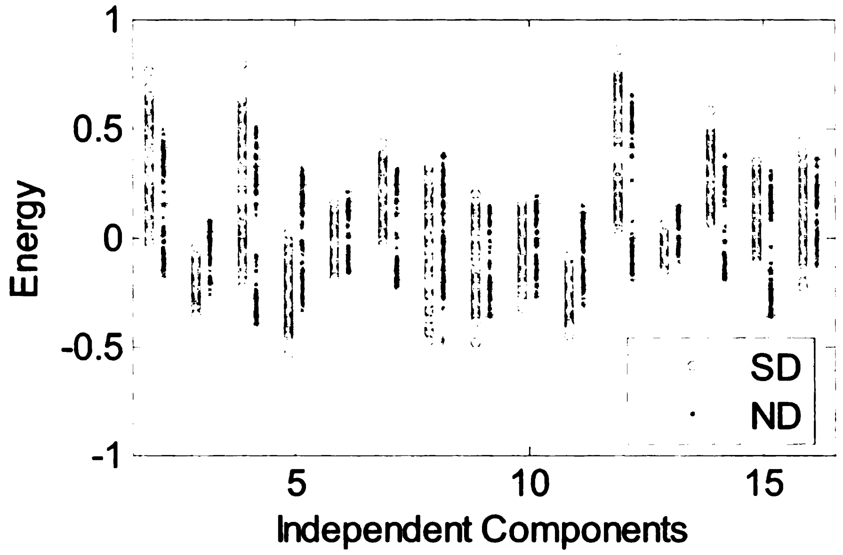


Figure 4.29 Separability of the ICA

Figure 4.30 to Figure 4.32 show the numerical measures for repeatability, separability and mutual information, respectively. The repeatability for most features was poor compared with features previously discussed. The high separability measure resulted from the ill-conditioned problem mentioned in the previous section and values of very high SEP features are not shown in Figure 4.31. The separability measure was not consistent with the results shown in Figure 4.29. The mutual information values indicate that ICA may have a better performance than PCA since ICA features have relatively high mutual information values.

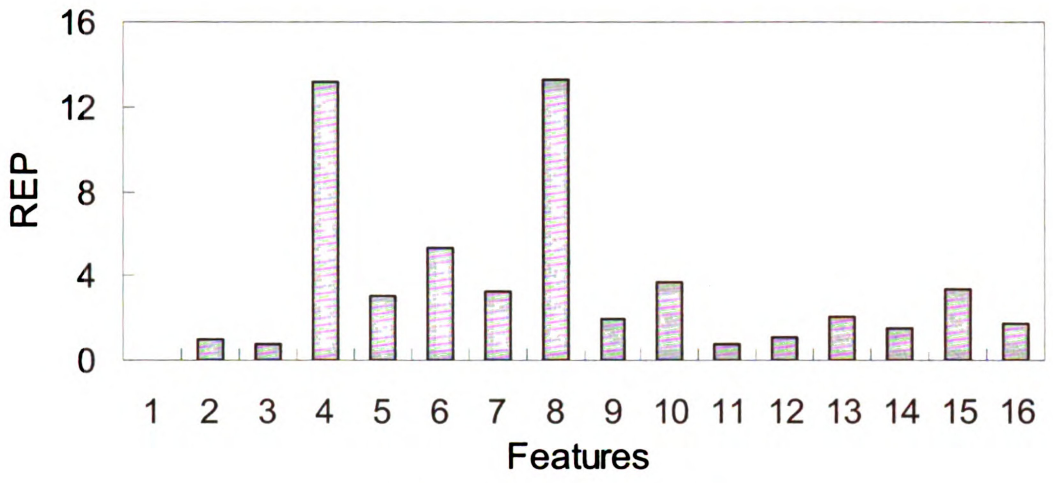


Figure 4.30 REP of the ICA

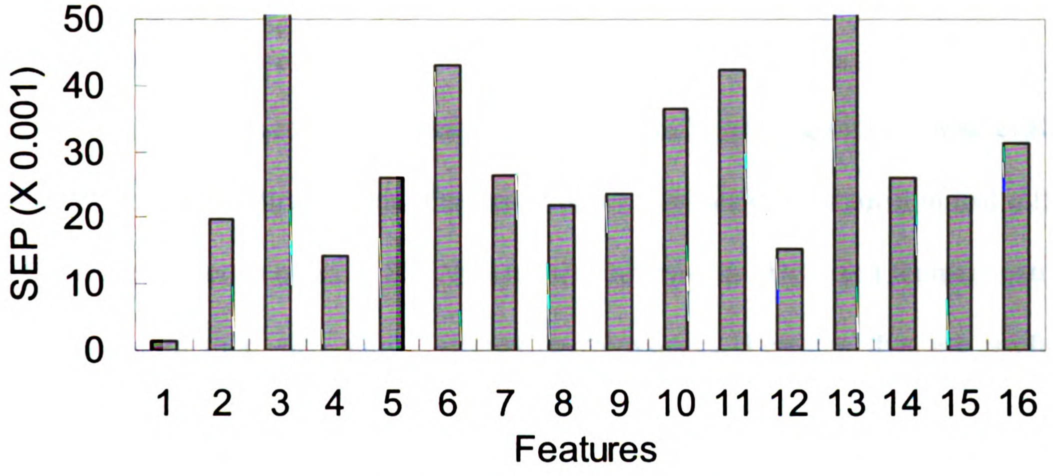


Figure 4.31 SEP of the ICA

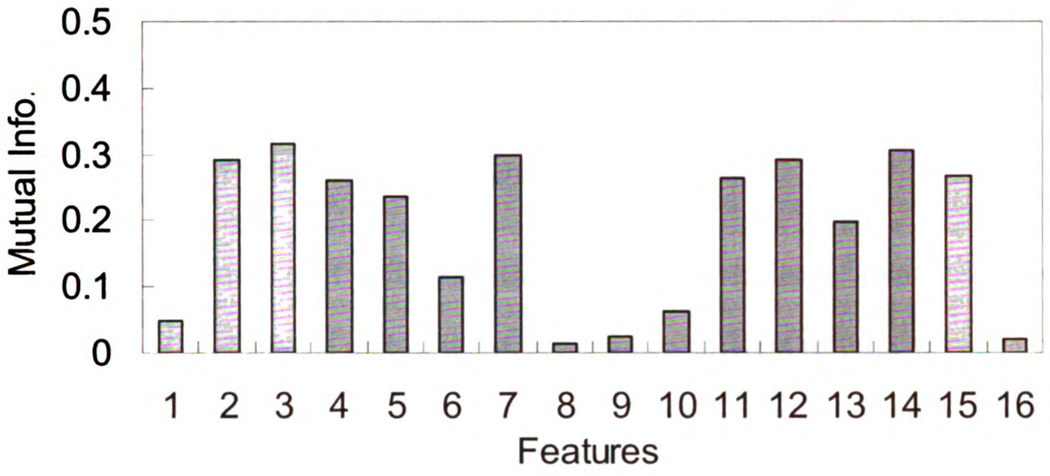


Figure 4.32 Mutual Information of the ICA

4.2.7 Summary of the Section

The performance of features extracted by different algorithms was evaluated using different evaluation criteria. Features extracted by different algorithm had different performances in terms of REP, SEP and mutual information. Different features extracted by the same method also show different performances. Features that are useful for delamination detection should be retained and others should be eliminated to facilitate delamination detection.

4.3 Selection of the Best Feature Extraction Algorithm

The previous section evaluated the performance of different feature extraction algorithms. Different algorithms performed differently when measured using different criteria. This section describes how the feature extraction algorithm for the delamination detection was selected. In the first section, applicant algorithms are evaluated based on their ranking in terms of repeatability, separability and mutual information. In the second section, the performance of feature extraction algorithms are tested using the data mentioned in Section 4.2.1 and the algorithms are further evaluated based on the rate of misclassification.

4.3.1 Algorithm Selection Based on Weighted Rank

The performance of different algorithms was calculated from the numerical performance: REP, SEP and mutual information. Since different features extracted by the same algorithm performed differently, it was necessary to select features that had the best performance. Based on the research on several commonly used classifiers described in Chapter 5, the performance reached or became close to the optimal performance when the number of features was four. The performance of feature extraction algorithms were calculated from four features that had the best performance for each criterion. For example, when comparing different algorithms in terms of repeatability, the four features with the lowest REP will be selected and the mean REP value of the selected features was calculated as the repeatability measure of this algorithm. After the performance of each algorithm was calculated, they were ranked from good to poor according to the numerical performance measure. The algorithm with the lowest rank was selected as being the best. Figure 4.33 to Figure 4.35 show the performance of different algorithms. The SEP values of PCA and ICA were too high and cannot be completely shown in Figure 4.34.

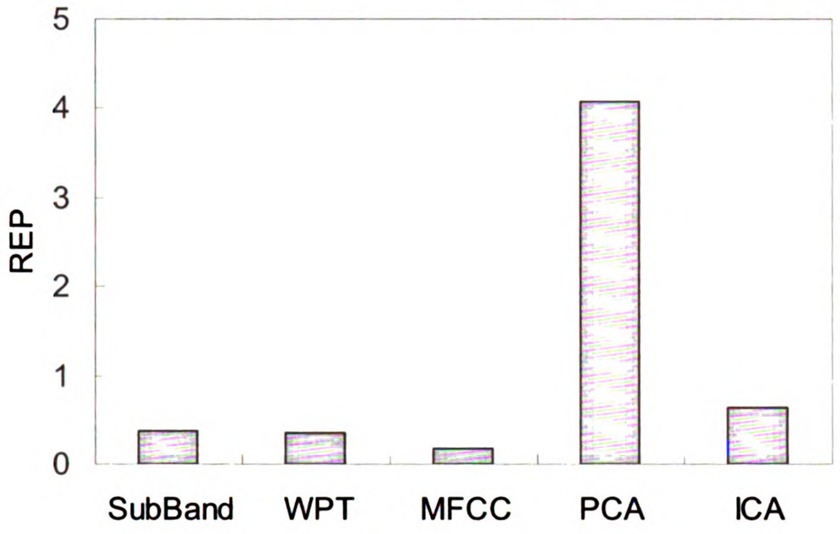


Figure 4.33 REP of Different Algorithms

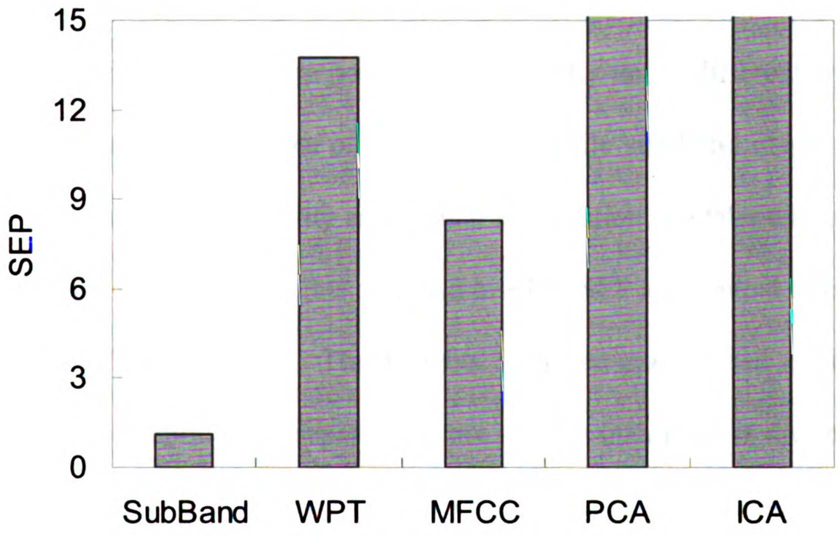


Figure 4.34 SEP of Different Algorithms

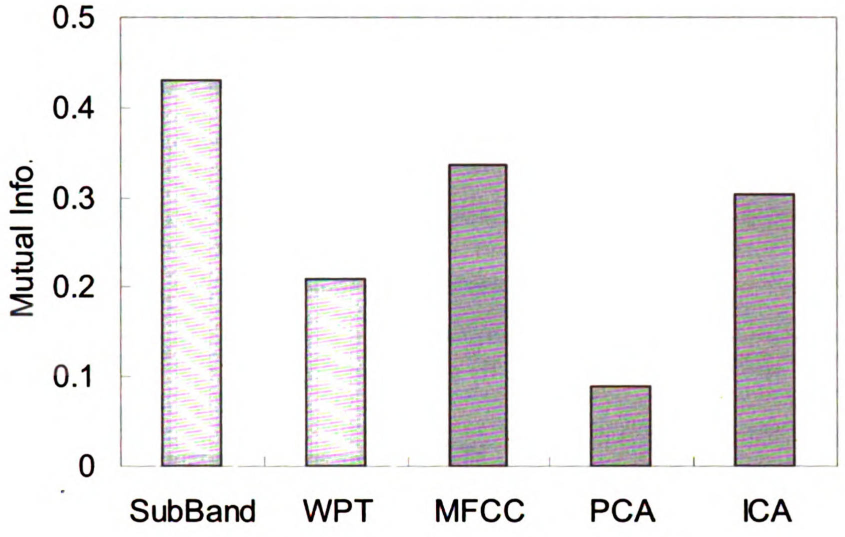


Figure 4.35 Mutual Information of Different Algorithms

Clearly, different algorithm performed differently under different criteria. MFCC had the best performance in terms of repeatability, while sub-band energy had the best performance in terms of separability and mutual information. To take this into account, a weighted rank was used. The weights assigned to REP, SEP and mutual information were 0.5, 0.25 and 0.25, respectively. The weights were assigned based on the following considerations: repeatability and separability are equally important in the selection of the algorithm and were assigned equal weight of 0.25. SEP and mutual information are essentially two different ways of measuring of separability. Therefore the weight for separability was equally distributed between SEP and mutual information. The rank of each individual criterion and the weighted rank of different algorithms are summarized in Table 4.1. It can be seen that MFCC has the best overall performance. The results are further confirmed using experimental data in the next section.

Table 4.1 Rank of Different Feature Extraction Algorithms

	Sub-band	**WPT**	**MFCC**	**PCA**	**ICA**
REP	2	3	1	5*	4
SEP	5	3	4	2*	1*
Mutual Info.	1	4	2	5	3
Weighted Rank	2.5	3.25	2	4.25*	3*

where * means the performance index has the ill-conditioning problem.

4.3.2 Algorithm Selection Based on Error Rates

The purpose of comparing different feature extraction algorithm is to find the algorithm that has the best accuracy in the detection of concrete delamnation. In this section, the performance of the feature extraction algorithms was measured using the error rate when classifying experimental data using a simple Bayesian classifier.

The first step was to select the features that are useful for delamination detection because "unwanted" features will decrease the accuracy of the detection and increase the computational load for classification. The mutual information was used to select "useful" features in this study. Similar to Section 4.3.1, the first four features with the highest mutual information values were selected as useful features and were used to train the classifier. The trained classifier was then used to classify different types of data and the error rate (defined below) of different algorithm was used as the criterion to evaluate the performance of the candidate feature extraction algorithms for the damage detection.

The data mentioned in Section 4.2.1was used. From each group (ND1, ND2, SD1 and SD2), 10 impact signals were randomly selected as the training signals. The remaining signals were used as testing signals. The features of these 40 signals were extracted and the mutual information was calculated based on the training signals. The four features with the highest mutual information were selected as the effective feature for detection. The features of the testing signals were also extracted and then selected based on the effective features obtained in the training step. The effective features of the training signals were used to train a linear Bayesian classifier. Detailed information about the classifier are described later in Chapter 5. The trained classifier was then used to classify the testing signals into two groups: solid or delaminated. The number of misclassifications was recorded and the error rate was computed as the ratio of the number of misclassification and the number of testing signals. Due to the variance of the signals, the effective features selected based on the mutual information may change with the selection of the training signals and the error rate may change accordingly. Considering this, the average error rate of 100 simulations was used to compare the performance between different feature extraction algorithms. The performance of different algorithms is summarized in Table 4.2. In this table, error 1 refers to the case where the concrete is solid but is classified as delaminated and, error 2 refers to the case where concrete is delaminated but was classified as solid.

Table 4.2 Error Rate of Different Feature Extraction Algorithms

Algorithm	**Sub-band**	**WPT**	**MFCC**	**PCA**	**ICA**
Error 1 (%)	6.95	13.19	3.98	21.02	8.55
Error 2 (%)	9.14	16.96	6.25	28.54	11.20
Total Error (%)	15.79	30.38	10.23	49.56	19.75

It can be observed that the weighted ranking in Table 4.1 agree well with the ranking of error rate in Table 4.2. This means that the repeatability and mutual information are both important in the selection of feature extraction algorithms. It can also be seen that MFCC has the lowest weighted rank and the smallest error rate. MFCC is therefore selected as the feature extraction algorithm to detect delamination in concrete bridge decks.

Conceptually, the difference in the performance of different feature extraction algorithms can be explained as follows. Sub-band energy and the wavelet packet tree essentially filter the sound using a series of filters distributed linearly on the frequency scale while MFCC filters the signal using filters that are evenly distributed on the log-frequency scale. The human ear is more sensitive to changes on a log-scale. Therefore MFCC has a better performance than sub-band energy and wavelet packet energy. PCA and ICA reduce the features by finding the optimal bases in different ways, but information about the class of the signal is not included. They are therefore indiscriminant in the process of feature extraction. In addition, the adaptiveness in the feature extraction will make the detection more dependent on the selection of the training data, which may also explains the high error rate for PCA and ICA.

4.4 Summary

This chapter focused on the problem of how to extract different features of the impact signals for the purposes of delamination detection.

Five commonly used feature extraction algorithms were introduced. The sub-band energy method extracts the distribution of the energy of the signal in different sub-bands; this is the same as finding the energy of the signal filtered through a series of rectangular filters in the frequency domain. Extracting features using the wavelet packet tree is equivalent to finding the energy of the signals filtered through filter banks at different levels. Mel-frequency cepstral coefficient (MFCC) is the spectrum of the log of the power spectrum filtered through a series of triangular filters in the frequency domain whose centers are evenly spaced on the Mel-scale. Principal component analysis (PCA) reduces the dimension by finding a linear transformation of the signal such that the mean square error between the signal with reduced dimension and the original signal is minimized. This keeps the features that contain most of the energy of the signal. Independent component analysis (ICA) finds the "basis" or independent components (ICs) of the signals in a statistical sense such that most of the "information" about the original signal is retained. The spectra of the ICs are used to replace the triangular filters in MFCC. The cepstral coefficients of the filter bank output were computed and used as the features.

The performance of the feature extraction algorithms were compared and evaluated against various criteria including repeatability, separability and mutual information. Different feature extraction algorithms had different performance and different features from the same feature extraction algorithm performed differently since information contained in each feature was different. Also, performance measures of repeatability and separability can be ill-conditioned in certain cases. Mutual information between extracted features and class labels are more consistent across all the algorithms.

The effectiveness of different feature extraction algorithm was evaluated using a weighted rank and the error rate of the classification because both repeatability and separability are important in the evaluation of the feature extraction algorithm. MFCC had the best overall performance and was therefore selected as the feature extraction algorithm for use in this study.

CHAPTER 5

PATTERN RECOGNITION AND DELAMINATION DETECTION

In Chapter 4, different feature extraction algorithms were compared to find the optimal features for delamination detection. Mel-frequency Cepstral Coefficients (MFCCs) were found to be most efficient. Once features of the acoustic signal are extracted and selected, the next task in the delamination detection is to differentiate the signals recorded on solid concrete from those recorded on delaminated concrete. This problem can be formulated as a classification problem and the task is to classify the recorded signals into two groups: signals from solid concrete and those from delaminated concrete. There are an infinite number of ways of drawing a dividing line between the two groups. Rather than drawing the boundary empirically "by eye", the line should be drawn optimally with respect to certain criteria. Different classification algorithms optimize different criteria. It is therefore necessary to compare and evaluate different algorithms to select the classifier that has the best performance for delamination detection.

As discussed in the previous chapter, the repeatability and separability of the features extracted from impact signals were not good, which made the classification task complicated. The classifier needs to accommodate the variance between different tests and to separate signals from different types of concrete. In this chapter, four commonly used classifiers including the Bayesian classifier, the support vector machine (SVM), the multi-layer perceptron (MLP), and the radial basis function (RBF) network are compared and evaluated. After comparing the performance of these classifiers, the best classifier is selected.

5.1 Detection Algorithms

This section briefly described the theoretical background of the four classifiers mentioned earlier. The classification is essentially an optimization problem: the classifier tries to minimize or maximize the cost function depending on certain criteria. The parameters of the classifiers are tuned using a training data set and the trained classifiers are then used to classify new data.

5.1.1 Bayesian-Based Classifier

Bayesian-based classifiers are derived from total probability and Bayes rules [51] and the target is to minimize the probability of classification error. For the case of delamination detection, there are only two classes: solid (labeled as C_1) or delaminated (labeled as C_2). Given the observed feature x_i, the classification problem can be formulated as:

$$d_i = \begin{cases} 1, & \text{if } x_i \in C_1 \\ -1, & \text{if } x_i \in C_2 \end{cases} \tag{5.1}$$

For the Bayesian classifier, if the feature is less than a certain threshold, the data is classified as C_1; if the feature is greater than the threshold, the data is classified as C_2. Based on this, the Bayesian decision rule can be expressed as:

$$d_i = \begin{cases} 1, & \text{if } x_i < x_0 \\ -1, & \text{if } x_i > x_0 \end{cases} \tag{5.2}$$

where x_0 is the threshold of the Bayesian classifier, as shown in Figure 5.1.

The goal here is to find an optimal x_0 such that the probability of misclassification (P_e) is minimized. Assuming that a priori probability of C_1 and C_2 are the same, the P_e can be derived as:

$$\begin{aligned} P_e &= P\left(C_1 \middle| x \in C_2\right) + P\left(C_2 \middle| x \in C_1\right) \\ &= P(C_1)\left[\int_{-\infty}^{x_0} P(x \mid C_2)\,dx + \int_{x_0}^{\infty} P(x \mid C_1)\,dx\right] \end{aligned} \tag{5.3}$$

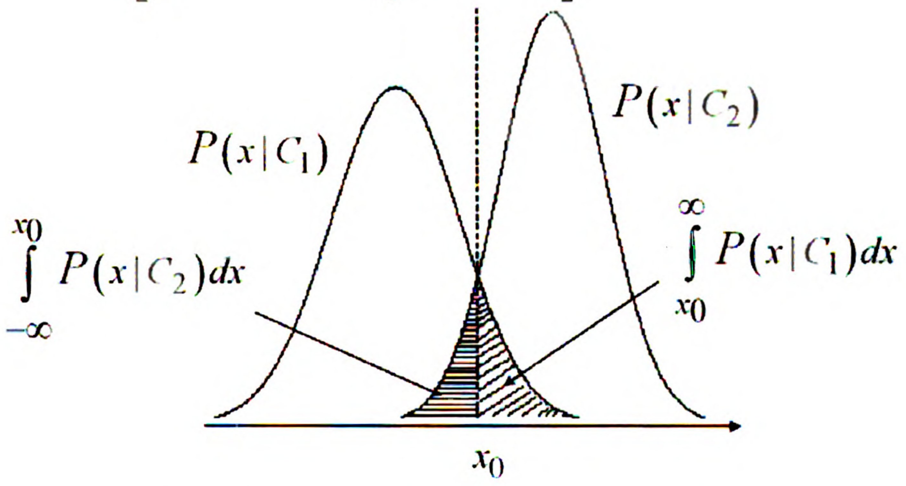

$P(x|C_2)$
$P(x|C_1)$
$\int_{-\infty}^{x_0} P(x|C_2)dx$
$\int_{x_0}^{\infty} P(x|C_1)dx$
x_0

Figure 5.1 Threshold of Bayesian Classifiers

$P\left(C_1 \middle| x \in C_2\right)$ can be expressed by the area on the left of the threshold under the curve of $P\left(x \middle| C_2\right)$ in Figure 5.1. Similarly, $P\left(C_2 \middle| x \in C_1\right)$ is the area on the right of the threshold under the curve of $P\left(x \middle| C_1\right)$. The probability of misclassification is then proportional to the shaded area shown in Figure 5.1. From the figure, it is obvious that when the threshold is at the intersection of the two curves, the shaded area is minimized and therefore, the misclassification probability is minimized. Therefore, the Bayesian rule can be expressed as:

$$d = \begin{cases} 1, \text{ if } P(x \mid C_1) > P(x \mid C_2) \\ -1, \text{ if } P(x \mid C_1) < P(x \mid C_2) \end{cases} \tag{5.4}$$

The decision surface of the Bayesian classifier can be expressed as the zeros of the following equation:

$$g(x) = P(x \mid C_1) - P(x \mid C_2) = 0 \tag{5.5}$$

The most commonly encountered probability density function in practice is the normal (Gaussian) distribution. In this case, the expression of the threshold can be further simplified.

The conditional probability density function of a jointly normal vector x can be expressed as:

$$P(x \mid C_i) = \frac{1}{(2\pi)^{l/2} |\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) \right) \tag{5.6}$$

where Σ_i is the covariance matrix of each class and μ_i is the mean value.

The decision surface for class i can be expressed as:

$$g_i(x) = -\frac{1}{2} (x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i) - \frac{l}{2} \ln(2\pi) - \frac{1}{2} \ln |\Sigma_i| \tag{5.7}$$

For uncorrelated x, the covariance matrix is diagonal and the decision surface becomes a quadratic function. Further, if the variance of all elements of x is equal, the decision surface reduced to be a hyper-plane, and the Bayesian classifier becomes a linear classifier.

From the above description, it can be seen that the decision surface can be found if the underlying distribution is known. In practice, this assumption is nct necessarily true and the distribution or the parameters of the underlying distribution need to be estimated. There are different ways to estimate unknown information such as maximum likelihood estimation [52] or expectation maximization [53].

5.1.2 Support Vector Machine

The Bayesian classifier described in the previous section tries to find the decision surface by minimizing the probability of misclassification. However, it requires prior information about the underlying distribution. Even though this information can be obtained through hypothesis testing or parameter estimation, the performance depends on how well the information about the underlying distribution is estimated. Linear classifiers which are not dependent on the underlying distribution of the training data provide one solution to this problem [54]. Linear classifiers try to classify data into different groups by a hyper-plane. As can be seen from Figure 5.2, there are infinite numbers of hyper-planes that can separate the two classes. The problem then is to find the optimal hyper-plane. One commonly used decision surface is the hyper-plane that maximizes the margin of separation as shown in Figure 5.2.

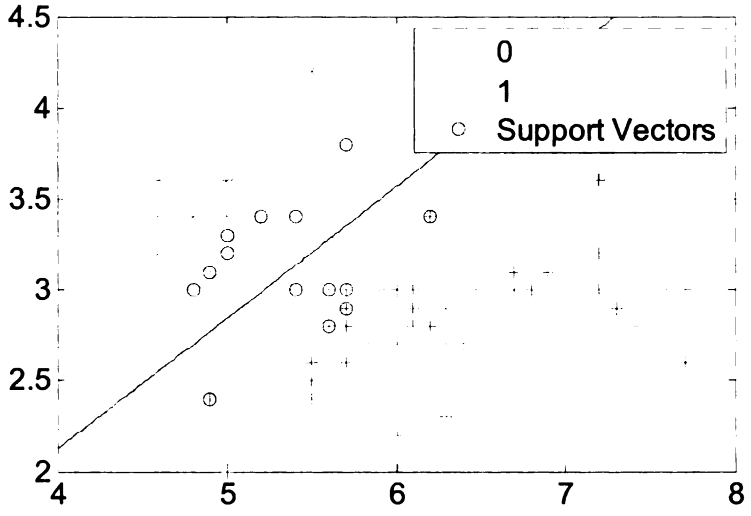


Figure 5.2 Support Vector Machine

For a linear classifier, given data point x_i, the decision can be expressed as:

$$d_i = \begin{cases} 1, \text{ if } w^T x_i + b > 0 \\ -1, \text{ if } w^T x_i + b < 0 \end{cases} \tag{5.8}$$

where, $g(x) = w^T x + b$ is the decision surface (hyper-plane in this case) and, w and b are the weighting and bias vectors, respectively.

The distance of the data point x_i to the decision surface can then be written as:

$$r = \frac{g(x_i)}{\|w\|} \tag{5.9}$$

If the training data is linearly separable, it is always possible to find a hyper-plane that satisfies Equation (5.10) below by scaling the weighting and bias vectors.

$$g(x_i) = \begin{cases} w^T x_i + b \geq 1, \forall x \in C_1 \\ w^T x_i + b \leq -1, \forall x \in C_2 \end{cases} \tag{5.10}$$

The equality is satisfied for the points that are closest to the decision plane. These points are called support vectors.

The margin between the two classes can then be expressed by the distance between the support vectors and the decision plan as:

$$r = \frac{g(x_{s1}) + g(x_{s2})}{\|w\|} = \frac{2}{\|w\|} \tag{5.11}$$

where, x_{s1} and x_{s2} are the support vectors in the two classes.

From Equation (5.11), it can be seen that maximizing the margin between the two classes is equivalent to minimizing the norm of the weight vectors under the constraint of Equation (5.10). In fact, Equation (5.10) can be combined with Equation (5.8) to yield:

$$d_i\left(w^T x_i + b\right) \geq 1 \tag{5.12}$$

This optimization problem can be solved using Lagrange multipliers. The Lagrangian function can be constructed as:

$$J(w, b, \lambda_i) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \lambda_i \left[d_i \left(w^T x_i + b \right) - 1 \right] \tag{5.13}$$

where, λ_i are the Lagrange multipliers and N is the number of training data sets.

The Lagrangian function in Equation (5.13) becomes stationary at the optimal solution and therefore:

$$\frac{\partial J}{\partial w} = w - \sum_{i=1}^{N} \lambda_i d_i x_i = 0 \tag{5.14}$$

$$\frac{\partial J}{\partial b} = \sum_{i=1}^{N} \lambda_i d_i = 0 \tag{5.15}$$

Substituting (5.12) and (5.14) into (5.13) and using the Kuhn-Tucker conditions [55], the optimization of the Lagrangian function is equivalent to minimizing:

$$J(\lambda_i) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j d_i d_j x_i^T x_j \tag{5.16}$$

under the constraints of (5.15) and non-negative Lagrange multipliers λ_i. From the training data, the optimal Lagrange multipliers λ_i can be obtained. Substituting this into Equations (5.14) and (5.10), the optimal hyper-plane can be uniquely determined.

If the classes are not linearly separable, it is always possible to construct a non-linear transformation that transforms the input features to a higher dimensional space in which the two classes becomes linearly separable. Such a transformation function is called a kernel function. Different kernel functions may have different effects on the performance of the classifier. In this study, a quadratic kernel function and a linear kernel function were used.

5.1.3 Multi-Layer Perceptron

The SVM described in the previous section is a linear classifier and linearly non-separable data needs to be transformed to a higher dimensional space such that the data can be linearly separable. A non-linear classifier, on the other hand separates the linearly non-separable data using a non-linear transformation. The multi-layer perceptron (MLP) [36] is such non-linear classifier.

A multi-layer perceptron model is a network of several layers of neurons, as shown in Figure 5.3. It consists of three parts: an input layer, one or more hidden layers, and an output layer. The input layer consists of a series of neurons that receive the input data, in this case, the extracted features of the impact signals. The hidden layer consists of several layers of neurons, which take the outputs of the previous layer as inputs and compute the output and feed it to neurons in the next layer. The neurons in the output layer receive the outputs from the hidden layer and compute the final output.

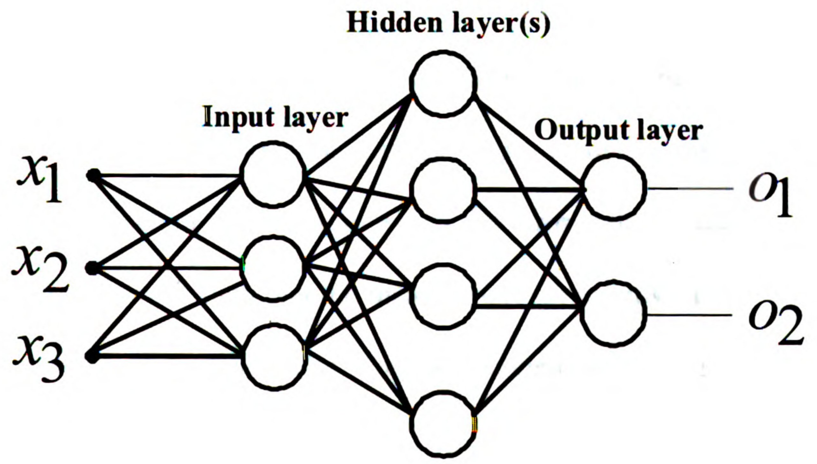


Figure 5.3 Multi-Layer Perceptron

Each neuron in the system is a computation unit. The neuron can compute the weighted sum of the inputs and the summation is fed into an activation function that produces the output. Figure 5.4 shows the signal flow of a perceptron in the output layer connected with a perceptron in the hidden layer. The difference between a perceptron in the output layer and perceptrons in other layers is that the output layer perceptron has a desired output. The difference between the actual and desired output is called the error signal and is used to update the synaptic weights in the back propagation step described below.

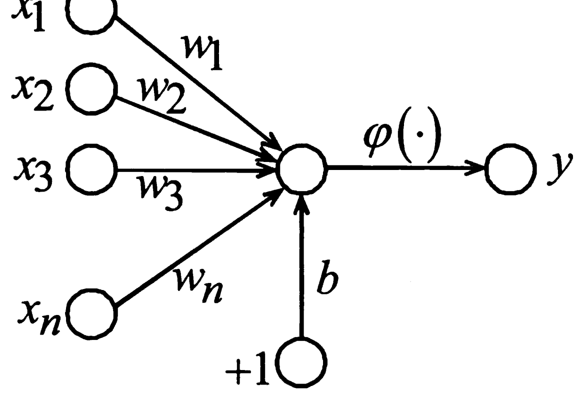


Figure 5.4 Signal-Flow Graph of the Perceptron

The computation consists of two phases. The first phase is called the feed forward process, in which the input is fed into and passed through the system. The output is computed and compared with the desired output to obtain the error signal. The second phase is called back propagation, in which the error signal is used to update the synaptic weighs in the network such that the mean square error between the system output and the desired output is minimized. The steepest gradient decent algorithm is used to find the update law of the synaptic weights.

The feed forward process is simple and straightforward. The problem is to find how the synaptic weight is updated by using the error signals. Assume that the output layer is the k^{th} layer, the mean square error between the actual output and the desired output can be computed as:

$$\varepsilon(n) = \frac{1}{2}\sum_{k} e_k^2 \tag{5.17}$$

where e_k is the error signal given by:

$$e_k = d_k - y_k \tag{5.18}$$

where d_k is the desired output and y_k is the actual output. The actual output can be computed from:

$$y_k = \varphi(v_k) \tag{5.19}$$

where, $\varphi(\cdot)$ is the activation function of the layer and v_k is the weighted sum of the input from the previous layer, in this case the j^{th} layer. v_k is calculated through:

$$v_k = \sum_j w_{j,k} y_j \tag{5.20}$$

Using the gradient descent method, the update law of the weight can then be derived as:

$$\begin{aligned} \Delta w &= -\eta \frac{\partial \varepsilon}{\partial w} \\ &= -\eta \frac{\partial \varepsilon}{\partial v} \frac{\partial v}{\partial w} \end{aligned} \tag{5.21}$$

where, η is the factor that controls the learning rate and convergence of the MLP.

From Equation (5.20), it follows that:

$$\frac{\partial v}{\partial w_{j,k}} = y_j \tag{5.22}$$

where y_j is the output of the *previous* layer.

For the output layer:

$$\frac{\partial \varepsilon}{\partial v_k} = \frac{\partial \varepsilon}{\partial e_k}\frac{\partial e_k}{\partial y_k}\frac{\partial y_k}{\partial v_k} = -\sum_k e_k \varphi_k'(v_k) \tag{5.23}$$

Therefore, the update law for the output layer is:

$$\Delta w_{j,k} = \eta y_j \sum_k e_k \varphi_k'(v_k) \tag{5.24}$$

For the layer that is one layer prior to the output layer, say the j^{th} layer, the derivative can be expressed as:

$$\begin{aligned} \frac{\partial \varepsilon}{\partial v_j} &= \frac{\partial \varepsilon}{\partial v_k}\frac{\partial v_k}{\partial v_j} = \frac{\partial \varepsilon}{\partial v_k}\frac{\partial}{\partial v_j}\left(\sum_j w_{j,k} y_j\right) \\ &= \frac{\partial \varepsilon}{\partial v_k}\sum_j w_{j,k}\frac{\partial y_j}{\partial v_j} \\ &= \frac{\partial \varepsilon}{\partial v_k}\sum_j w_{j,k}\varphi_j'(v_j) \end{aligned} \tag{5.25}$$

where $\frac{\partial \varepsilon}{\partial v_k}$ is the local gradient in the *next* step.

Therefore, the update for the j^{th} layer is:

$$\Delta w_{j-1,j} = \eta y_{j-1}\frac{\partial \varepsilon}{\partial v_{j+1}}\sum_j w_{j,j+1}\varphi_j'(v_j) \tag{5.26}$$

The second phase - back propagation - starts with the output layer using Equation (5.26), and then iteratively using Equation (5.23) the synaptic weights of the entire network can be updated from the output layer to the input layer.

In this research, the MATLAB [32] neural network toolbox was used to design the classifier.

5.1.4 Radial Basis Function

The Radial Basis Function (RBF) network is another way to transform the input using a non-linear transformation for classification purposes. It also consists of three layers: the input layer, the hidden layer and the output layer. The difference between RBF and MLP is that RBF has only one hidden layer while MLP may have one or more hidden layers, and the transformation function for the RBF is symmetric about a point (that is why the classifier is called radial basis function) while the transformation of MLP is determined by the weighed sum from the previous layer and the activation function. Figure 5.5 shows the architecture of the RBF.

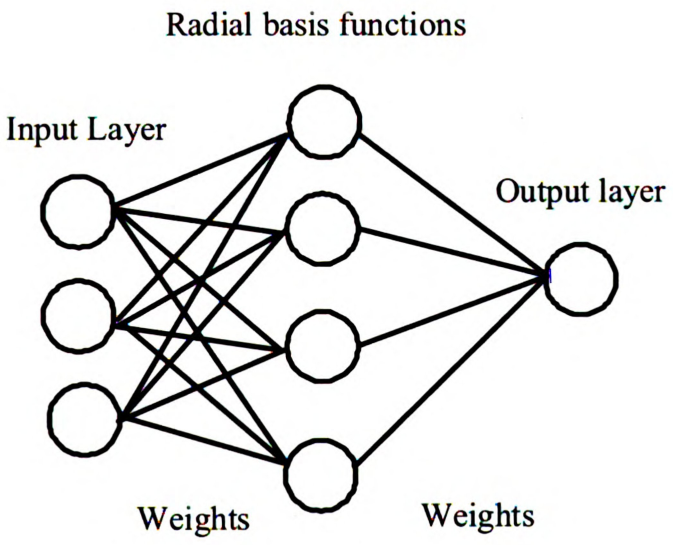


Figure 5.5 Architecture of Radial Basis Function Network

The output of the RBF network can be expressed as:

$$F(x) = \sum_{i=1}^{N} w_i \varphi(\|x - x_i\|) \tag{5.27}$$

where $\varphi(\|x-x_i\|)$ is a set of functions symmetric about the center x_i, w_i is the weight for each function, and N is the number of basis functions.

In reality, the number of training data may be greater than the number of underlying basis functions, the information provided is over-complete and the problem becomes ill-conditioned. In this case, the results of the RBF network described above may become an "over-fit", meaning that the network works very well for the training data but may not work well for other data.

To solve this problem, regulation theory [56] was proposed. The basic idea was to provide more degrees of freedom to the solution by adding some functions that embed prior information about the solution. The most commonly used function is the linear differential of the solution. This comes from the assumption that the mapping from input to the output is smooth or differentiable. The problem becomes the minimization of the regulated error given by:

$$\varepsilon = \sum_{i=1}^{N} \left[d_i - F(x_i) \right]^2 + \lambda \|DF\|^2 \tag{5.28}$$

where d_i is the desired output, λ is a positive real number called the regularization parameter, and D is a linear differential operator.

To solve this optimization problem, the output of the RBF, $\varphi(\cdot)$, is approximated by a family of Green's functions, $G(\|x-t_i\|)$, centered at t_i. The Green's functions can be derived from the RBF, $\varphi(\cdot)$. In this way, Equation (5.28) can be reformulated as [36]:

$$\varepsilon = \sum_{i=1}^{N} \left[d_i - \sum_{i=1}^{ml} w_i \underline{G}\left(\left\| x_i - t_i \right\|\right) \right]^2 + \lambda w^T G_0 w \tag{5.29}$$

where,

$$d = \begin{bmatrix} d_1 & d_2 & \dots & d_N \end{bmatrix}^T;$$

$$w = \begin{bmatrix} w_1 & w_2 & \dots & w_{ml} \end{bmatrix}^T$$

$$\underline{G} = \begin{bmatrix} G(x_1,t_1) & G(x_1,t_2) & \dots & G(x_1,t_{ml}) \\ G(x_2,t_1) & G(x_2,t_2) & \dots & G(x_2,t_{ml}) \\ \dots & \dots & \dots & \dots \\ G(x_N,t_1) & G(x_N,t_2) & \dots & G(x_N,t_{ml}) \end{bmatrix}$$

$$G_0 = \begin{bmatrix} G(t_1,t_1) & G(t_1,t_2) & \dots & G(t_1,t_{ml}) \\ G(t_2,t_1) & G(t_2,t_2) & \dots & G(t_2,t_{ml}) \\ \dots & \dots & \dots & \dots \\ G(t_{ml},t_1) & G(t_{ml},t_2) & \dots & G(t_{ml},t_{ml}) \end{bmatrix}$$

Once framed as a minimization problem in Equation (5.29), the unknown parameters in the RBF network can then be updated by an optimization algorithm such as the steepest gradient descent method.

The important parameters in the RBF network are: the number of RBFs (N), the location of the center, the width of the RBF (or the variance, σ), and the synaptic weights that connect the hidden RBF with the output layer. Usually, the number of RBFs and the width of the RBF are selected by the user, while the location of the center and the synaptic weights are optimized by using the training data.

5.2 Performance Evaluation

As described in the previous section, there are different ways to classify the extracted features into different groups. Different algorithms map the input features using different methods and optimize the different discriminant criteria. This section compares the performance of different algorithms and evaluates the effect of such parameters as the number of features. After the performance is evaluated, a decision can then be made as to which algorithm should be used for delamination detection.

As mentioned in Chapter 4, Mel-frequency Cepstral Coefficients (MFCC) had the best discriminant capacity and therefore were used as input features to test the performance of the different classifiers. In Chapter 4, the error rate was computed from 4 extracted features having the highest value of mutual information. In this section, a more thorough evaluation of the number of features is presented. The training samples need to be representative of the entire population. Therefore, training samples were randomly selected from each group (ND1, ND2, SD1, and SD2) and the number of training data was 10 (around 20% of the total population). The remainder of the signals was used to test the performance of the classifier.

Even though the number of training samples was the same, the training samples were randomly selected. Since the selection of training samples was random, the results could be different. Figure 5.6 shows the error rates of 100 simulations. To accommodate the variance due to the random selection of the training samples and to have a more fair comparison between different cases and algorithms, the upper limit for the 95% confident interval (CL) of the error rates was used.

The number of features also plays an important role in the performance of the classifier. The error rates of different numbers of features were compared in this section to find the optimal number of features for the purpose of delamination detection.

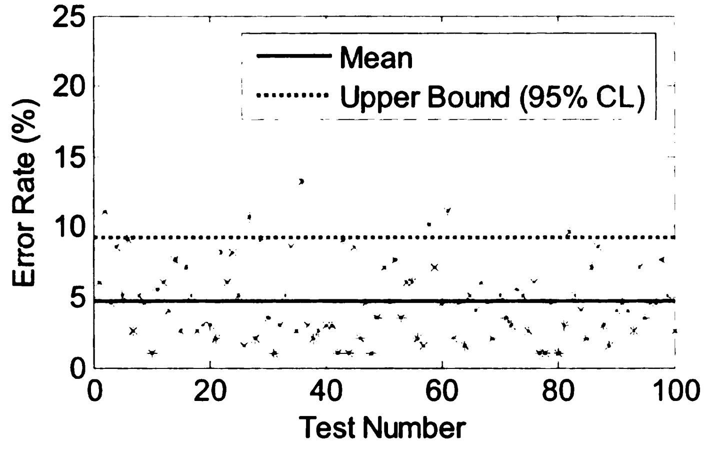


Figure 5.6 Variation of Error Rate due to Random Selection

5.2.2 Performance of Bayesian Classifier

This section evaluates the performance of the Bayesian classifier. To simplify the computation, several assumptions are made in this section. First, since the prior probability of the delaminated concrete and the solid concrete were unavailable, they were assumed to be 50% for both. Second, the underlying distribution of the extracted features was assumed to be normal. Third, the covariance matrix described in Section 5.1.1was taken to be diagonal.

In this section, two cases are considered. Case one (Linear Bayesian Classifier) assumes that the diagonal elements of the covariance are equal, meaning that the extracted features are independent and have the same variance. As described in the previous section, the Bayesian classifier in this case became a linear classifier. The second case (Quadratic Bayesian Classifier) is more general and only assumes that different features are independent of each other, the resulting decision surface is of quadratic form.

The error rate of linear the Bayesian classifier (Case 1) with different number of features is plotted in Figure 5.7.

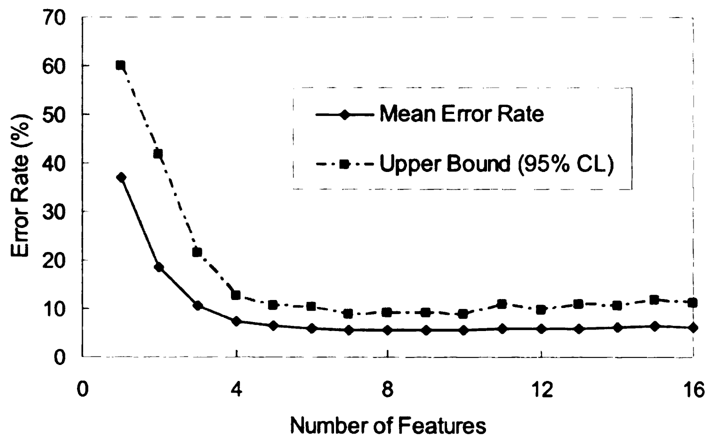


Figure 5.7 Performance of Linear Bayesian Classifier

The error rate dropped with an increase in the number of features, but the increase in the performance is limited. The optimal performance is reached when the number of feature is seven and the optimal performance has a mean error rate of 5.46%, and the upper bound of the 95% CL is 8.85%.

The error rate versus the number of features for the quadratic Bayesian classifier is plotted in Figure 5.8.

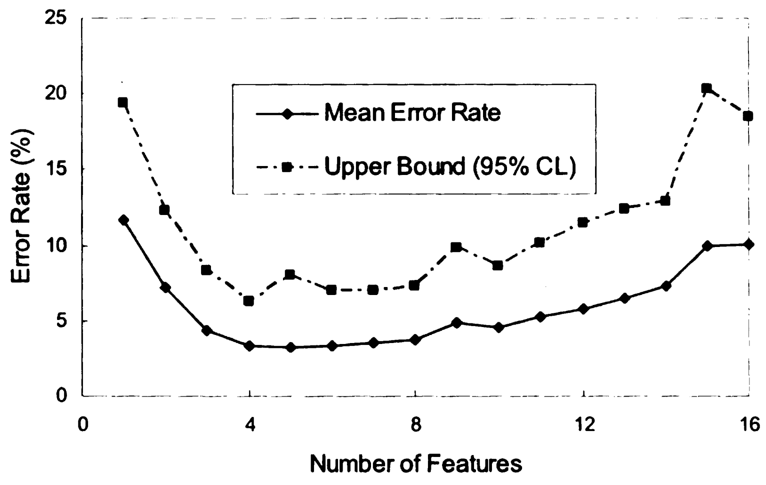


Figure 5.8 Performance of Quadratic Bayesian Classifier

The results show that with an increase in the number of features, the error rate first decreases and then increases. The reason for this is that information contained in the redundant features is not useful for or has a negative effect on delamination detection. The different trends for linear and quadratic Bayesian classifiers may come from their properties: the linear classifier is not sensitive to the redundant features. The optimal performance for the quadratic Bayesian classifier is achieved when the number of features is six and the error rate is 3.30% with an upper bound of 6.99%, both of which are lower than that of the linear Bayesian classifier. The quadratic Bayesian classifier has a better performance than the linear classifier because the quadratic Bayesian classifier only assumes the independence of the features and is therefore more general than the linear Bayesian classifier.

5.2.3 Performance of Support Vector Machine

The performance of the SVM was evaluated in this section. To compare the effect of the kernel function, two types of kernel functions were used in this section: a linear kernel function and a quadratic kernel function. Similar to the previous section, the average error rate and the upper 95% CL of 100 simulations are used as performance indices.

The performance of the linear kernel SVM is shown in Figure 5.9. The error rate decreases with an increase in the number of features. However, the improvement in performance was not significant when the number of features exceeded 5. The optimal performance for linear kernel SVM classifier was reached when the number of features was 12. The optimal error rate was 5.13% and the upper 95% CL was 8.42%.

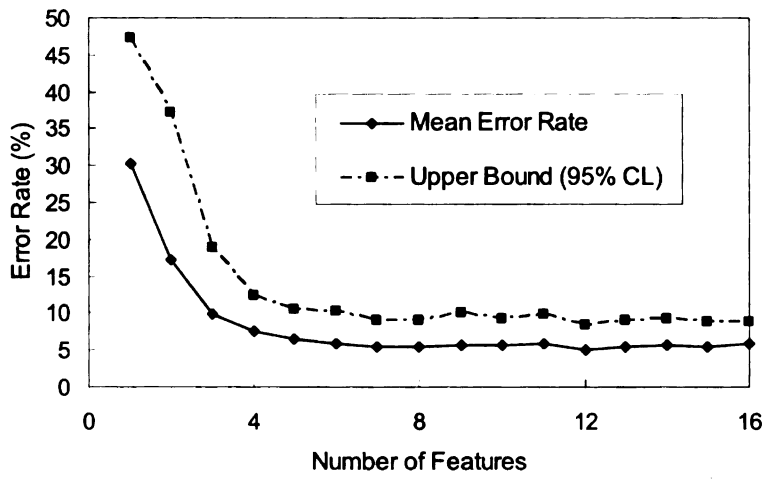


Figure 5.9 Performance of Linear Kernel SVM Classifier

The performance of the quadratic kernel SVM is shown in Figure 5.10. The relationship between the error rate and the number of features was similar to the linear kernel SVM. However, the performance of the quadratic kernel SVM is better than that of the linear kernel, indicating that the data tended to be better classified by a linear classifier using a quadratic transformation. The minimum error rate was only 2.55% and the upper 95% CL was 5.40%. The optimal number of features was 12.

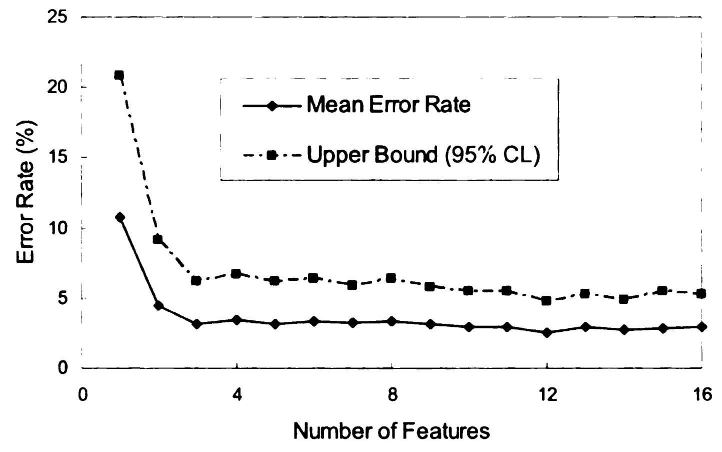


Figure 5.10 Performance of Quadratic Kernel SVM Classifier

5.2.4 Performance of Multi-Layer Perceptron

This section discussed the performance of the multi-layer perceptron (MLP). There are several factors that can affect the performance of an MLP classifier such as the number of hidden layers, the number of perceptrons in each hidden layer, and the choice of activation functions. When combining these factors, the MLP can have an infinite number of architectural structures. It is difficult to find the optimal structure by trial and error. Due to the relatively low dimension of the input (maximum dimension is 16), a 2-layer MLP was used and the number of perceptrons in each hidden layer was assumed to be the same. The activation function for all perceptrons was chosen to be the log-sigmoid function shown in Figure 5.11. The performance of the MLP with different numbers of neurons for each hidden layer is plotted in Figure 5.12, in which "MLP22" refers to the case where the number perceptrons in the two hidden layers were 2 and 2, respectively. From Figure 5.12, it can be seen that MLP22 had the highest error rate and the performance of MLP44 was the best among all the MLPs compared. Therefore, MLP44 was selected to represent the performance of MLP. Detailed information about the performance of MLP44 is plotted in Figure 5.13.

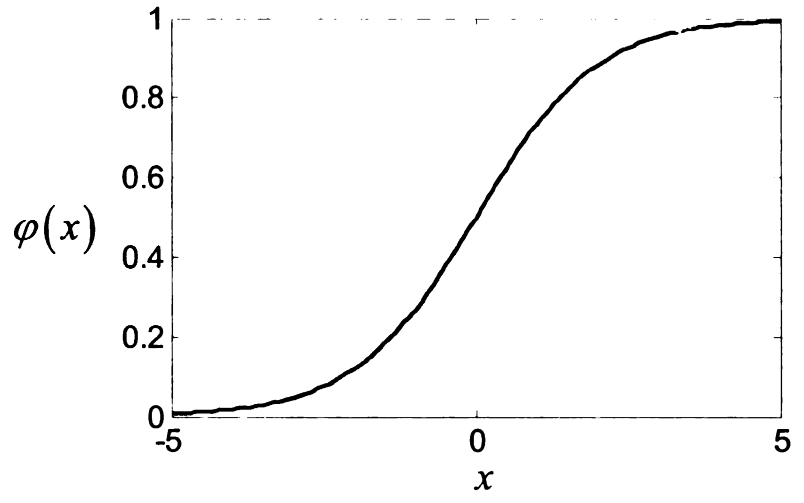


Figure 5.11 Log-Sigmoid Activation Function

Figure 5.13 indicated that the performance of the MLP was unsatisfactory when the number of features was too small or too large. This further confirmed that redundant features had a negative effect on the performance. The optimal number of features for MLP44 was 8 and the optimal mean error was 1.05% with an upper bound of 95% CL of 2.53%.

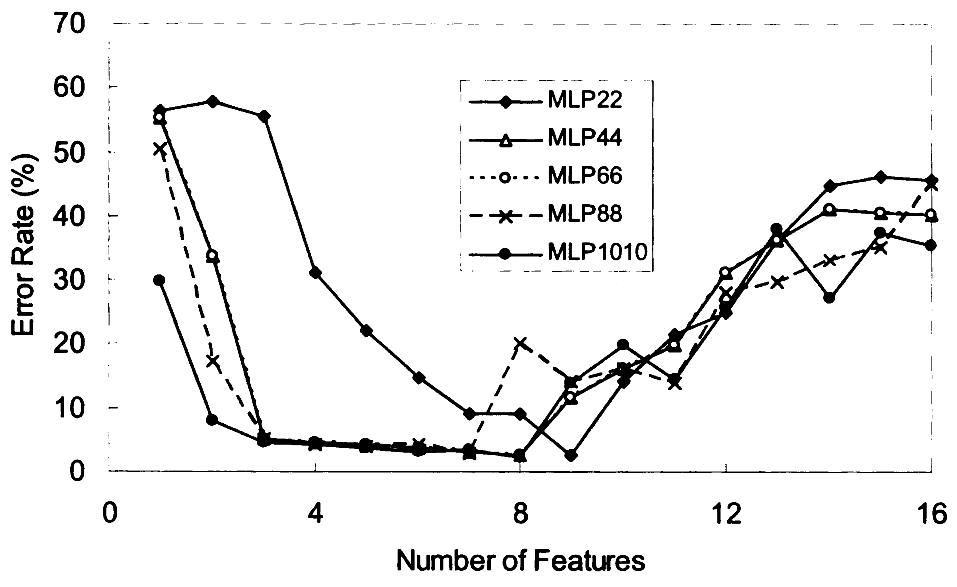


Figure 5.12 Performance of MLP with Different Structures

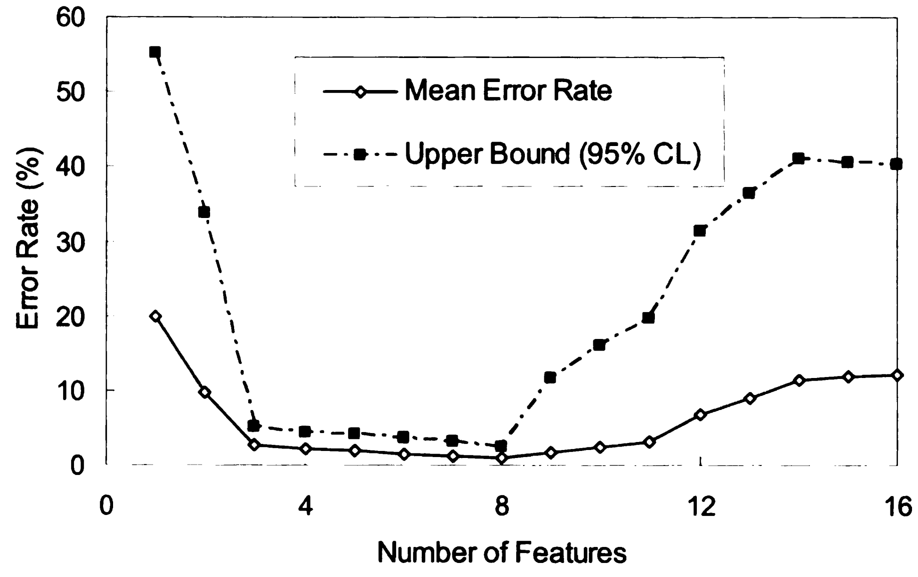


Figure 5.13 Performance of MLP44

5.2.5 Performance of Radial Basis Function

The performance of the RBF classifier can be affected by two factors: the width of the RBF and the number of neurons. Similar to the MLP, it is difficult to find the optimal structure for the RBF. Combinations of limited number of parameters were tried. The radial basis functions for all the hidden neurons were Gaussian functions.

Figure 5.14 shows the effect of the number of neurons. In this section, a Gaussian function with a spread of 100 was selected as the radial basis function. A large value was assumed here to prevent the RBF classifier from capturing only the local effect. Better performance was achieved when N increased from 5 to 10. However, the increase in the performance was not significant when N increased beyond 10. By comparing the error rate for different cases, the optimal number of neurons was found to be 20.

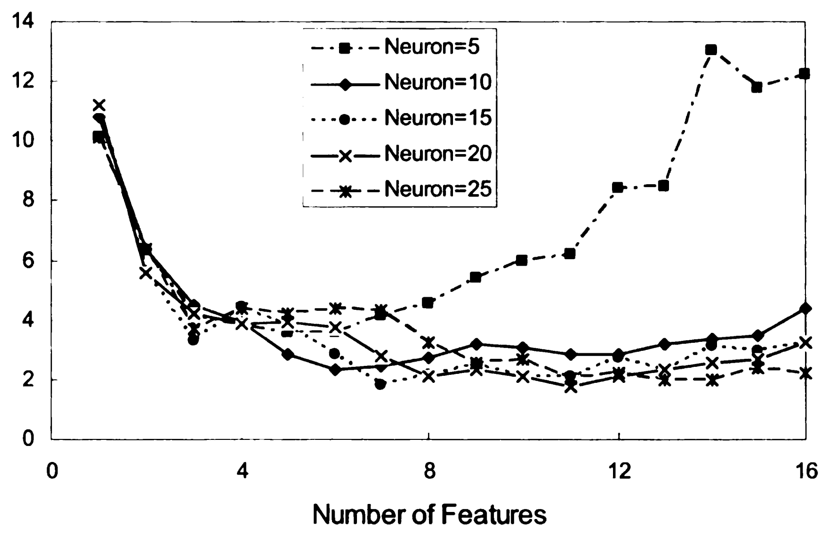


Figure 5.14 Effect of Number of Neurons on RBF

Figure 5.15 compares the error rate for different σ values. The number of neurons was 20 for the performance shown in Figure 5.14. For the smaller variance, the performance became worse as the number of features increased. For the larger variance, the performance was better and more stable. Although the performance was sensitive when the variance was small, the effect of variance on the behavior of the RBF was not significant when it was greater than 10. Based on this analysis, the optimal variance was selected to be 10.

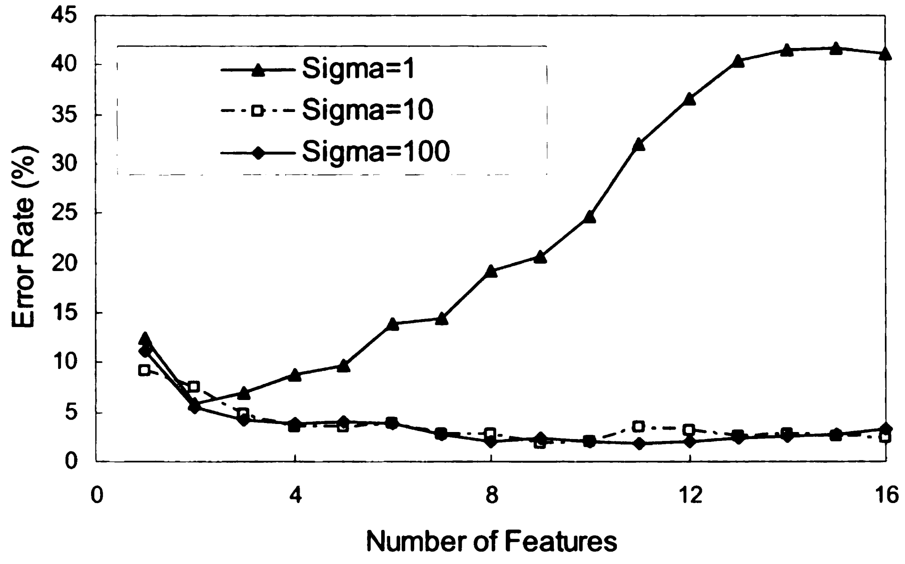


Figure 5.15 Effect of the Variance of RBF

Having optimized the number of neurons and the variance, the optimal RBF was a network with $\sigma = 10$ and $N = 20$. The performance of this classifier is shown in Figure 5.16. The number of features yielding the best performance was 9, the lowest error rate was only 0.59% and the upper 95% CL was 1.80%.

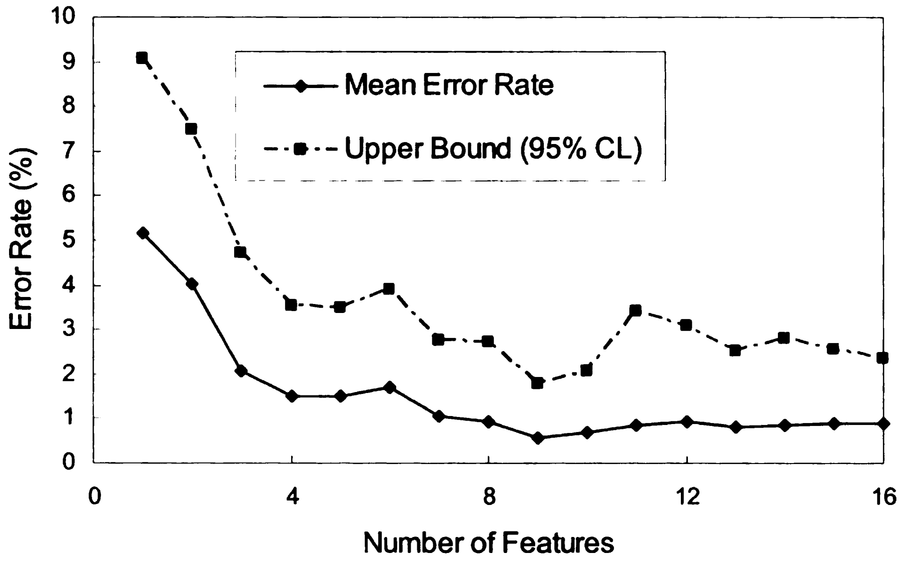


Figure 5.16 Performance of RBF Classifier

5.2.6 Selection of Detection Algorithm

In order to select the best classifier for delamination detection, the performance of the best classifiers from each of the categories is compared in Figure 5.17. SVM performed better than the quadratic Bayesian classifier because SVM does not pre-assume the distribution of the input data while the quadratic Bayesian classifier assumes that the extracted features are independent and have a normal distribution. Compared with linear mixtures such as the Bayesian classifier and SVM, MLP and RBF are non-linear classifiers which transform the data using non-linear transformations and make it easier to separate and therefore have greater flexibility in classifying the data. The optimal performance for non-linear classifers (MLP and RBF) is better than those of linear classifiers. The reason for the better performance of RBF than MLP is that the activation functions of hidden and output neurons in RBF can be different, while the activation functions of different neurons of MLP have to be the same. This provides more flexibility to RBF in finding the optimal transformation and dividing line.

After comparing the performance, the RBF with $\sigma = 10$ and $N = 20$ had the smallest error rate and was therefore selected for use in delamination detection. The optimal number of features for the RBF with these parameters was 9.

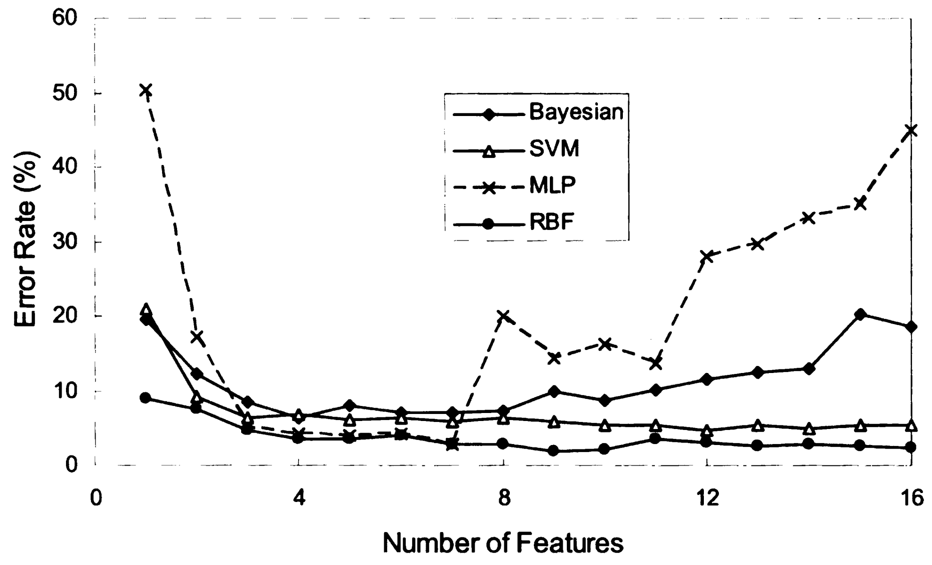


Figure 5.17 Comparison of Different Classifiers

5.2.7 Error Rate for Multiple Impacts

The error rates in previous sections were calculated from tests in which only one impact was performed at each location. In field inspection, multiple impacts would increase the accuracy of the detection as the error resulting from misclassification of one impact may be compensated by correct classification of other impacts at the same location. If N is the number of impacts, and N_1 and N_2 are the number of impacts classified as solid and delaminated, respectively, the final result of the class can be expressed as:

$$d = \begin{cases} solid & \text{if } N_1 \geq N_2 \\ delaminated & \text{if } N_1 < N_2 \end{cases} \tag{5.30}$$

In this case, an error occurs when the number of misclassification is greater than half of the total number of impacts. Assuming that different impacts are independent of each other and the error rate of an impact signal is ε_s, the error rate for multiple impacts will be:

$$\varepsilon = 1 - \sum_{i=0}^{\lfloor N/2 \rfloor} C_N^i \varepsilon_s^i (1-\varepsilon_s)^{N-i} \tag{5.31}$$

where, $\lfloor N/2 \rfloor$ is the maximum integer smaller than $N/2$.

Figure 5.18 shows the envelope of the error rate for multiple impacts. The error rate drops as the number of impacts increases. If the error rate of an individual impact is 20%, the final error rate for 5 impacts is approximately 6%. If the single impact error is 10%, the final error rate after 5 impacts is very small.

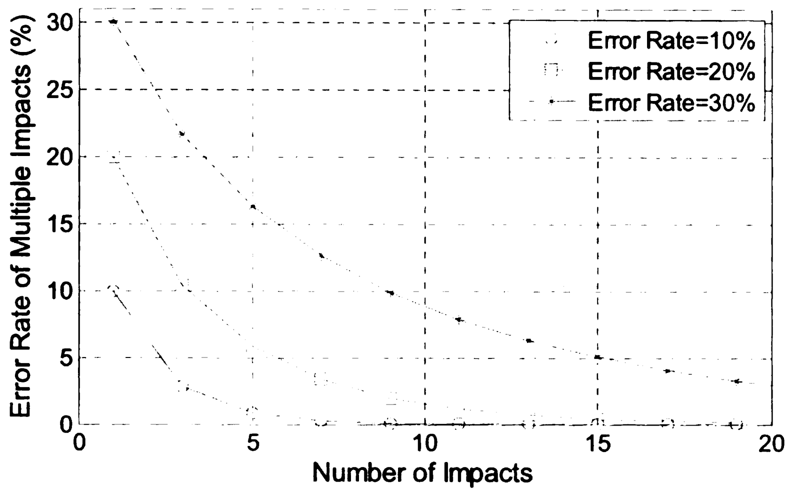


Figure 5.18 Error Rate of Multiple Impacts

5.3 Summary

This chapter evaluated and compared four types of commonly used classifiers: the Bayesian classifier, the support vector machine, the multi-layer perceptron network and the radial basis function network. The classifier for delamination detection was then selected based on performance.

This chapter first briefly described the theoretical background of four classifiers.

1. The Bayesian classifier finds the decision surface by minimizing the probability of misclassification. This classifier requires prior information about the underlying distribution of the input. If the underlying distribution is normal and if the covariance matrix is diagonal, the decision surface of the Bayesian classifier has a quadratic form. If the diagonal elements of the covariance matrix are equal, the decision surface is further reduced to a hyper-plane.

2. The support vector machine classifies different classes by finding a hyper-plane that maximizes the margin of separation and it does not require prior information about the underlying distribution of the input. The optimal hyper-plane can be found by using a Lagrange multiplier. For the case where the input data are not linearly separable, a non-linear kernel function must be used to transform the input data to a higher dimensional space where the classification becomes a linearly separable problem.

3. The multi-layer perceptron network consists of several layers of perceptrons. By adaptively changing the synaptic weights that connects different perceptrons, the mean square error between the desired output and the network output is minimized. This is equivalent to finding an optimal mapping between the inputs and the desired outputs.

4. The radial basis function network is another way to find the optimal mapping between the inputs and outputs. The difference is that the RBF consists of only one hidden layer and the mapping function is symmetric around the center. The classifier is trained by adjusting the centers of the RBF and the synaptic weights that connects the hidden layer and the output layer such that the error between the desired outputs and the actual outputs of the system is minimized.

The second part of the chapter evaluated the performance of the classifier because different classifiers use different optimization criteria and different approaches to find the optimal mapping. To compare the performance of different classifiers, the error rate and the upper 95% confidence interval under different numbers of features were plotted and used to compare performance to compensate for the variance due to the selection of training data.

1. By comparing two types of Bayesian classifiers, it was found that the quadratic Bayesian classifier had a better overall performance than the linear Bayesian classifier. "Redundant" features (when the number of features exceeds a certain number) had a negative effect on the quadratic Bayesian classifier, while the linear Bayesian classifier was not sensitive to redundant features.

2. For the SVM, both quadratic and linear kernel functions were compared. For both types of kernel functions, the SVM was not sensitive to the redundant features and the SVM with quadratic kernels had a better performance than the SVM with linear kernel functions.

3. Performance of the MLP increased significantly when the number of perceptrons in each hidden layer increased from 2 to 4. However, further increase in the number of perceptrons in the hidden layer did not yield significant improvement in performance. The MLP network was also sensitive to redundant features.

4. The performance of the RBF network was poor for small values of N and σ, especially when the number of features was high. Increasing N and σ improved the performance. However, improvement in the performance was not significant when the values of N and σ exceeded certain values. By comparing the performance of different classifiers, it was found that a RBF with $\sigma = 10$ and $N = 20$ had the best performance. The optimal performance of this classifier was achieved when the number of features was 9. Due to its superior performance, this classifier will be used for the delamination detection.

Lastly, the chapter also discussed the error rate when multiple impacts were performed at the same spot. The error rate dropped quickly with an increase in the number of impacts.

CHAPTER 6

SYSTEM DEVELOPMENT AND VERIFICATION

Detailed information about the delamination detection algorithms were described in previous chapters. After selecting the algorithms, it is necessary to test the performance of the combined system under different conditions. This chapter briefly describes how the different components, i.e., noise cancellation, feature extraction and selection and pattern recognition were incorporated to develop an automatic impact-based delamination detection (AIDD) system. After the parameters of the system were tuned, its performance was tested using both experimental and field data.

6.1 Hardware Development

An impact machine was designed and fabricated to automatically impact a concrete surface with constant energy. The impact is created by the free fall of the impactor from a constant height. The impactor is a stainless steel bar of diameter 25 mm (1 inch) with a ball-shaped head. The impactor is lifted by a pin on a flywheel. When the flywheel rotates to a certain location, the pin releases the impactor, which falls freely to impact the ground. An electro-magnetic catching mechanism was mounted on the cart to prevent multiple impacts due to the rebound of the impactor. When the impactor is lifted to a certain height, the magnet turns off to allow a consistent free fall. When the impactor drops to a pre-defined location, the magnet turns on to hold the impactor and prevent a double bounce. The impact and ambient sound are recorded by a condenser microphone. This microphone is directional and records the sound within a short range, which helped limit extraneous noise. Two microphones were mounted on the cart. The primary microphone (Mic. 1) was mounted under the base of the cart, pointing toward the impact point, to record the impacting sound. A sound proofing curtain was mounted as a physical barrier to block traffic and wind noise. The secondary microphone (Mic. 2) was mounted on the frame to measure the ambient noise. The scheme and a proto-type of the cart are shown in Figure 6.1 and Figure 6.2, respectively.

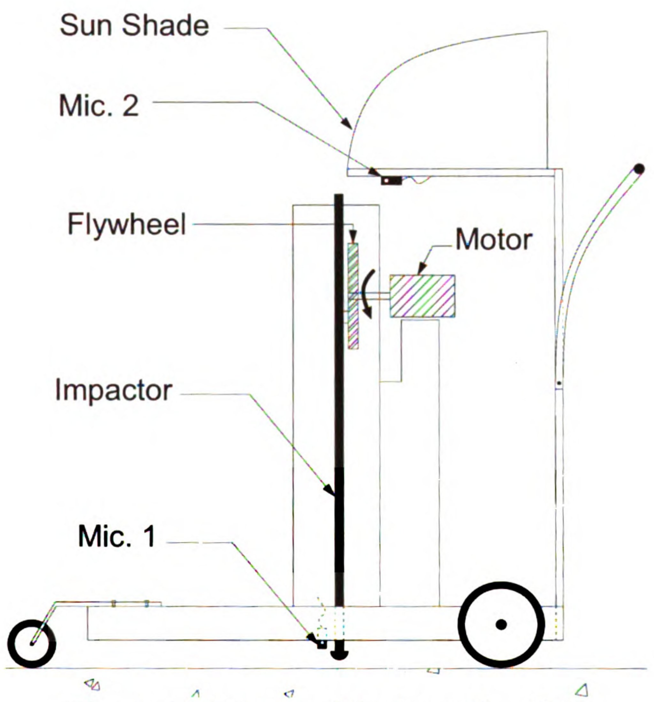


Figure 6.1 Scheme of the Impacting Cart

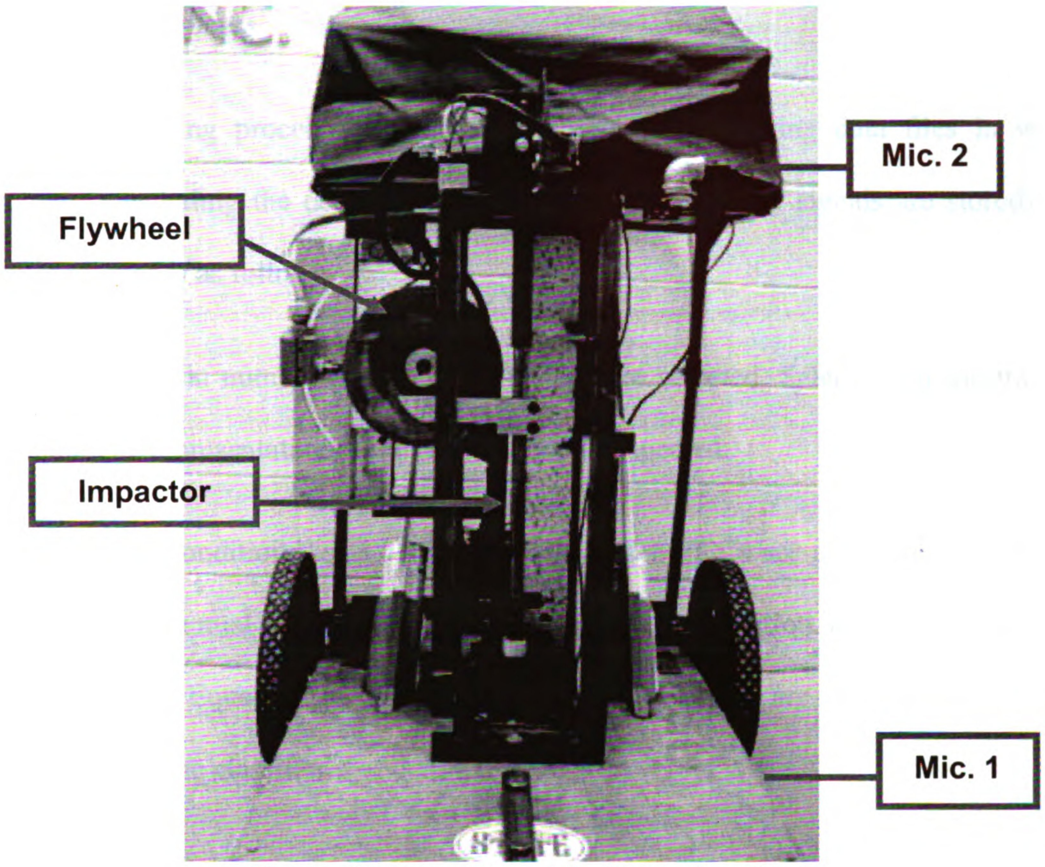


Figure 6.2 Proto type of the Impacting Cart

6.2 Software Development

There are two major components in the AIDD system: training and inspection. For practical implementation, training is conducted offline where selected signals from previous tests are used to train the classifier and to find effective features. Once the classifier is trained and features are selected, the information can be saved as a classifier file for future inspection. During inspection, a data acquisition system is used to record the sound. The recorded signal is first filtered through the modified ICA algorithm described in Chapter 3. The features, in this case, MFCCs were calculated and then classified using the detector obtained in the training process. This section describes the training and detection process in detail.

6.2.1 Training Process

The training process is performed offline using existing data files in which information including the deck condition, features and original signals are stored. The training process is as follows:

1. A certain number of training data files are selected. Selection of the training data should be representative of the structure to be inspected.

2. Deck condition (solid or delaminated) and MFCCs are read directly from the data file. The mutual information between the deck condition and each MFCC is calculated using Equation (4.19). MFCCs with high values are selected as effective features to train the classifier.

3. The classifier (RBF neural network) is trained using the effective features of the training signals. The training is performed using the artificial neural network tool box in MATLAB [32].

4. Once the training is completed, the classifier and the indexes of the effective features are saved to a classifier file that could be used for future inspection.

The flow chart of the training algorithm is shown in Figure 6.3.

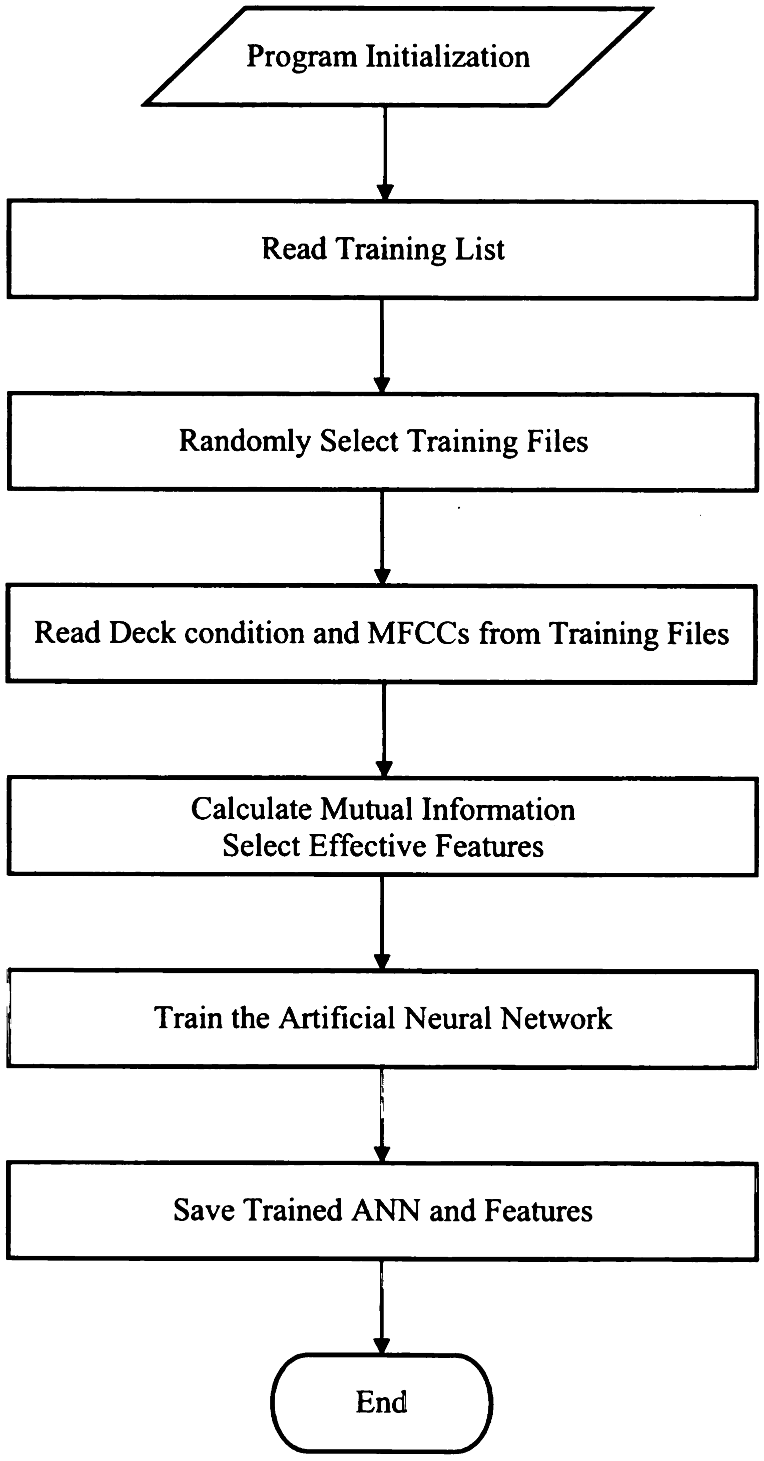

Program Initialization
Read Training List
Randomly Select Training Files
Read Deck condition and MFCCs from Training Files
Calculate Mutual Information
Select Effective Features
Train the Artificial Neural Network
Save Trained ANN and Features
End

Figure 6.3 Flow Chart of the Training Process

6.2.2 Inspection Process

The inspection process is performed at a bridge site and the analysis (including filtering and detection) is completed in a semi-real-time manner. The signal is first recorded and then processed by the computer. After the process at one location is completed, the system is able to process the data at a new location. The estimated time needed to perform the analysis (filtering, feature extraction and detection) for 3 seconds of signal sampled at 10 kHz is about 4 seconds on a laptop computer (1.8Ghz CPU and 3Gb RAM).

The inspection process consists of the following steps:

1. The impact signal is recorded by two microphones and digitalized by a data acquisition card.

2. The impact sound and the noise are separated from the recordings using the modified ICA described in Chapter 3.

3. The impact is identified from the recorded signal based on the voltage across the magnet described in Section 6.1 since the time delay between the impact and the instant the magnet is turned on is fixed.

4. The MFCCs of the impacts in the filtered signal are calculated as features.

5. The MFCCs obtained in Step 4 are used by the classifier obtained during the training process to determine whether the concrete deck is delaminated or solid.

6. Information about the recording, such as the concrete deck condition, calculated features and the original recording, etc. is saved as a data file for future use.

7. Steps 1 to 6 are repeated until the inspection is completed.

The flow chart for the inspection process is shown in Figure 6.4.

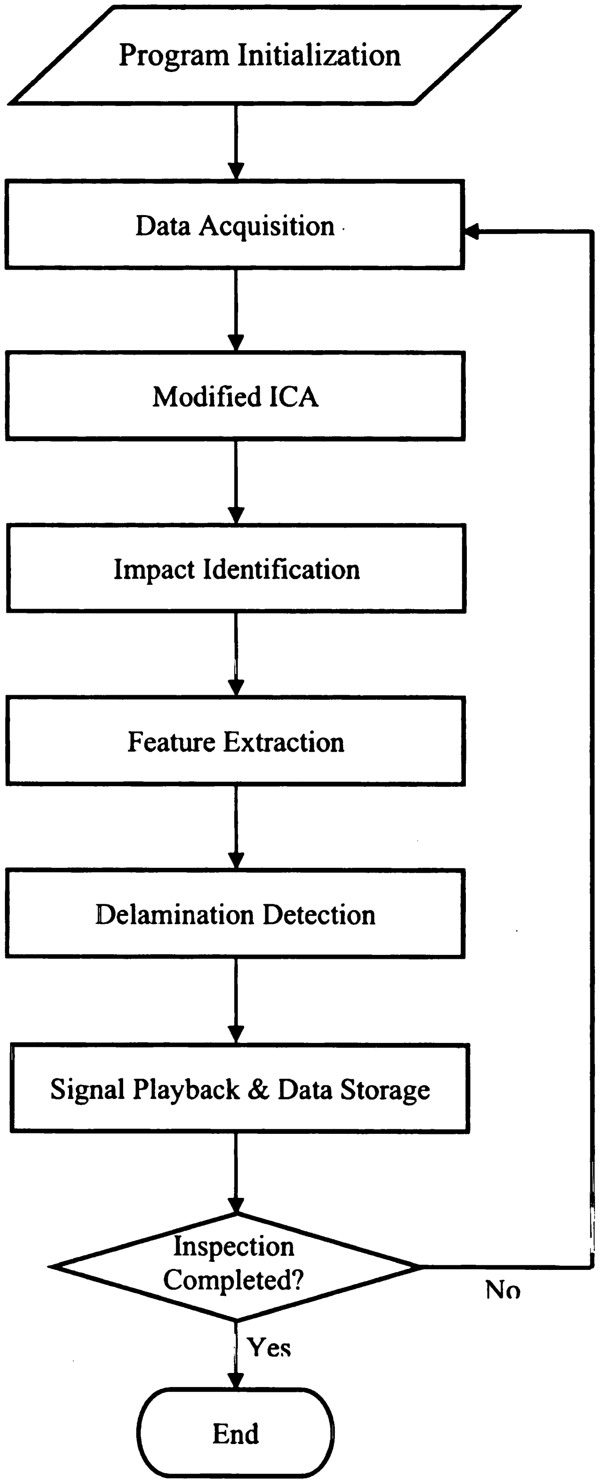


Figure 6.4 Flow Chart of the Inspection Process

6.2.3 Implementation of the Algorithms

Detailed information about the algorithms for training and inspection were described in previous chapters. One more step is needed to develop a practical tool that can be used in field inspection: to implement the algorithms into an executable program with a proper user interface.

Mixed-language programming in MATLAB [32], C++ and LabVIEW [57] was used to implement the algorithm. LabVIEW provides a simple interface between the computer and the data acquisition hardware. It includes many built-in libraries and instruments drivers, and no programming at the hardware level is needed. In addition, it has a graphical programming environment that makes it easy to create a graphical user interface (GUI). MATLAB is a convenient programming language and has powerful toolboxes and built-in functions. It is also capable of converting scripts into executable (.exe) files or dynamic link library (.dll) files, both of which can be run outside the MATLAB environment. However, MATLAB and LabVIEW cannot communicate data directly. Therefore, C++ programs were used as wrappers or bridges to enable the data communication between LabVIEW and MATLAB. The data communication inside the system is shown in Figure 6.5.

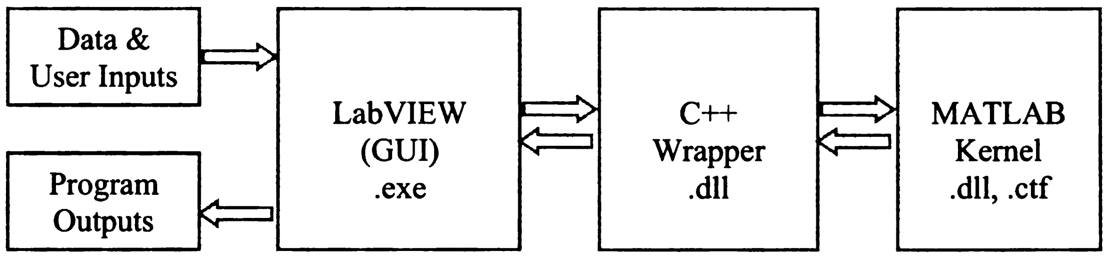
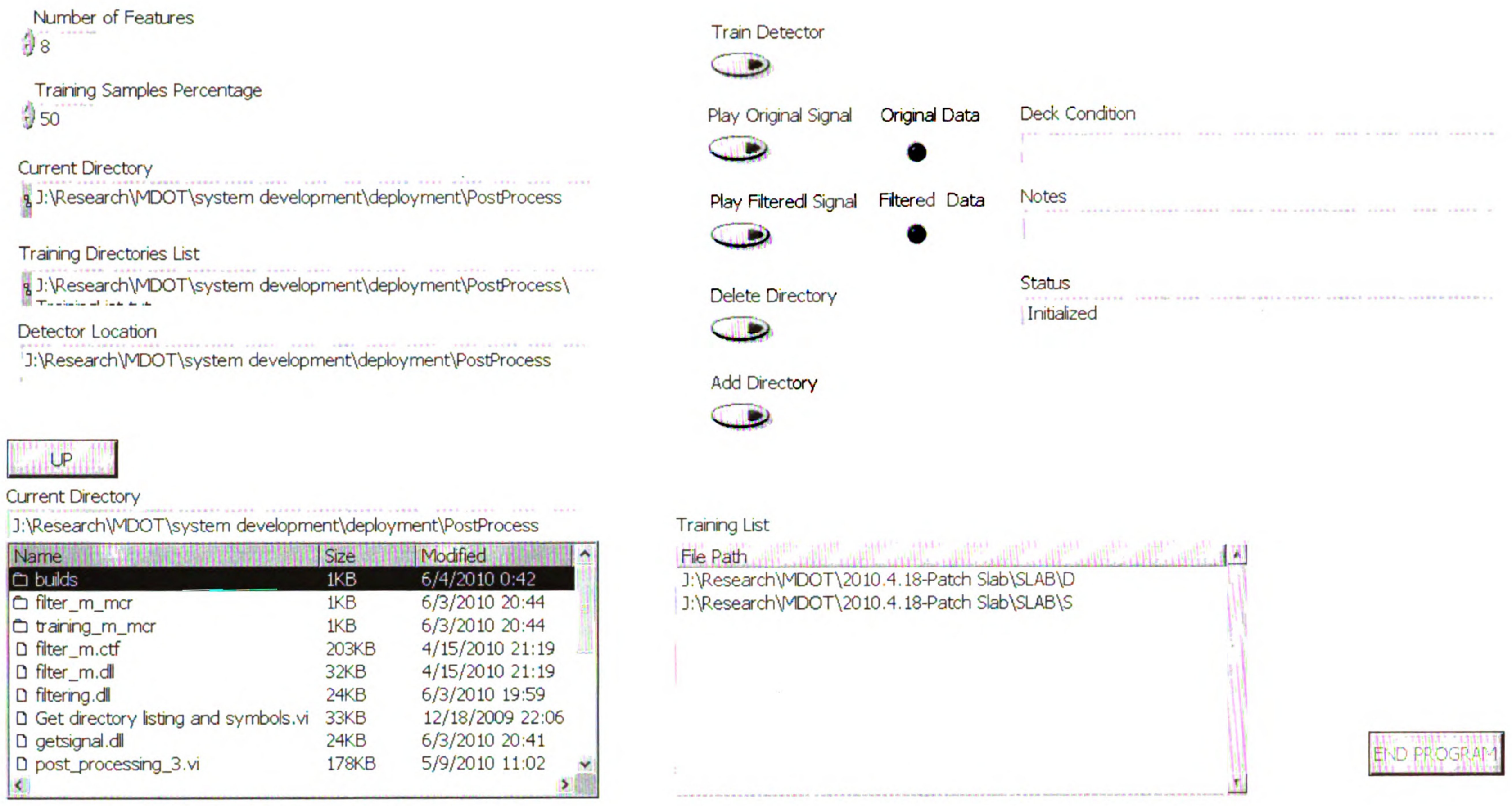


Figure 6.5 Data Communication

Figure 6.6 and Figure 6.7 show the GUI for the training module and the inspection module, respectively.


Number of Features
8

Training Samples Percentage
50

Current Directory
J:\Research\MDOT\system development\deployment\PostProcess

Training Directories List
J:\Research\MDOT\system development\deployment\PostProcess\

Detector Location
J:\Research\MDOT\system development\deployment\PostProcess

Current Directory
J:\Research\MDOT\system development\deployment\PostProcess

Name	Size	Modified
builds	1KB	6/4/2010 0:42
filter_m_mcr	1KB	6/3/2010 20:44
training_m_mcr	1KB	6/3/2010 20:44
filter_m.ctf	203KB	4/15/2010 21:19
filter_m.dll	32KB	4/15/2010 21:19
filtering.dll	24KB	6/3/2010 19:59
Get directory listing and symbols.vi	33KB	12/18/2009 22:06
getsignal.dll	24KB	6/3/2010 20:41
post_processing_3.vi	178KB	5/9/2010 11:02

Deck Condition

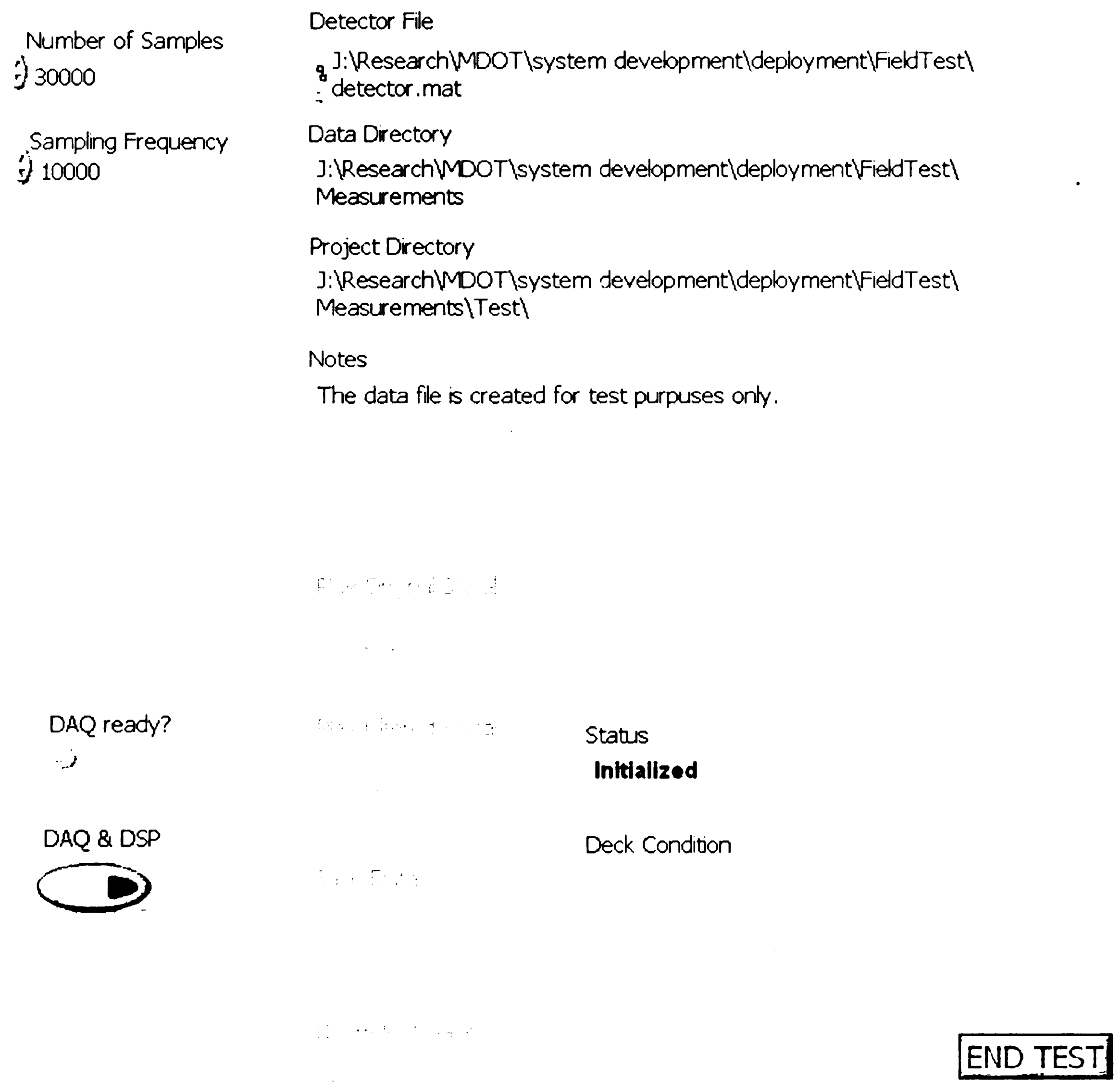
Notes

Status
Initialized

Training List

File Path
J:\Research\MDOT\2010.4.18-Patch Slab\SLAB\D
J:\Research\MDOT\2010.4.18-Patch Slab\SLAB\S

Figure 6.6 GUI forTraining Module


Number of Samples
30000

Sampling Frequency
10000

Detector File
J:\Research\MDOT\system development\deployment\FieldTest\detector.mat

Data Directory
J:\Research\MDOT\system development\deployment\FieldTest\Measurements

Project Directory
J:\Research\MDOT\system development\deployment\FieldTest\Measurements\Test\

Notes
The data file is created for test purpuses only.

DAQ ready?

Status
Initialized

DAQ & DSP

Deck Condition

Figure 6.7 GUI for Inspection Module

Detailed information about the algorithms and the wrapper as well as the LabVIEW flow chart are included in Appendix B.

6.3 Algorithm Verification

The performance of the proposed algorithms was tested using data from both laboratory and field tests. The performance under different noise levels and the effectiveness of the algorithms on different types of repair patches were verified using laboratory tests. Field data was used to test the performance of the system under real environments. The results of the tests are described in this section.

6.3.1 Performance under Different Noise Levels

To verify the performance under different noise levels, impact testing was carried out on a slab constructed in the laboratory. Figure 6.8 and Figure 6.9 show a photograph and an elevation view of the slab with artificial delamination. The depth of the delamination was controlled by the location of the separation. The thickness of the slab was 229mm (9 inches) and the depth of the two "delaminated" parts were 76.2 mm (3 inches) and 152 mm (6 inches), respectively to simulate different delamination depths. The design strength of the concrete was 27.6 MPa (4 ksi), typical of concrete used in bridge decks. Trial tests showed that the sound produced from the 6-inch delamination was very similar to that produced by the solid concrete. This is because the energy of the impact was not large enough to excite the mode where the difference between the 152 mm (6-inch) delamination and the solid concrete (229 mm (9 inches) in thickness) could be clearly observed. Therefore, in the analysis of the result, signals from these two cases were combined and labeled as "solid".

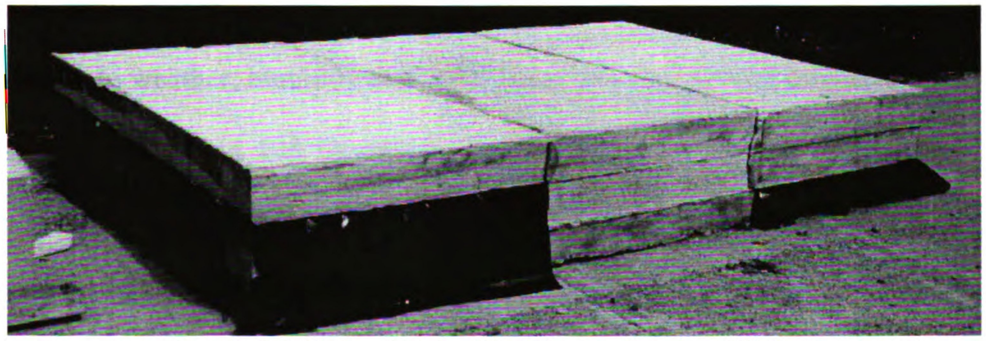


Figure 6.8 Slab with Artifical Delamination (Slab 1)

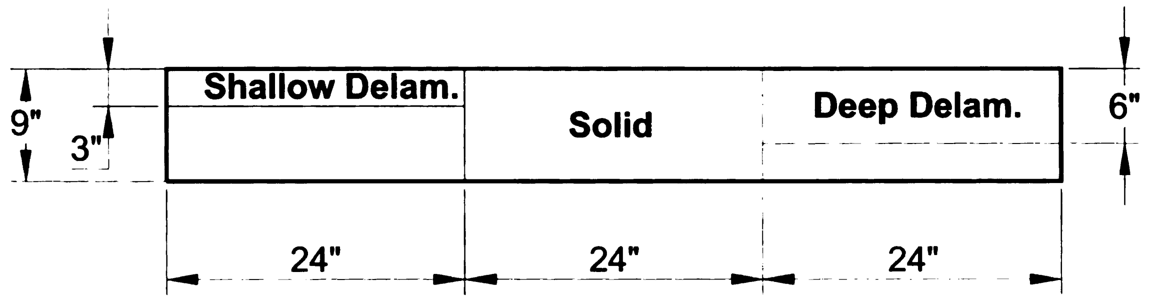


Figure 6.9 Elevation View of Slab 1

The noise level was low in the laboratory environment and the recordings from the primary microphone were clean enough to perform the analysis. To evaluate the performance of the algorithm with noisy input, recordings with different signal-to noise ratios (SNRs) were simulated by mixing recorded traffic noise signals with the impact signal obtained in a quiet laboratory environment. Different noise levels were obtained by mixing the scaled impact signal with noise signals as shown in Equation (3.19), except that the impact sound and the noise were convoluted versions. Four noise levels were considered: quiet condition ($\alpha = \infty$), low noise level (α =10), medium noise level (α = 1), and high noise level (α = 0.1). The modified ICA was used to perform noise elimination and then the MFCCs of both noisy recordings and the filtered signals were computed for comparison.

A total of 228 impacts were recorded on two different days to account for variance due to weather, humidity and wind between days. 120 recordings were obtained from solid concrete and 108 recordings were obtained from delaminated concrete. 40 randomly selected impacts were used for feature extraction and classifier training. The remaining signals were classified by the trained classifier. The average error rates under different conditions were calculated using the method described in Chapter 5. The results are listed in Table 6.1. As in Table 4.2, error 1 refers to the case where the concrete is solid but is classified as delaminated, and error 2 refers to the case where the concrete is delaminated but was classified as solid. This also applies to the remainder of the tables in this chapter.

Table 6.1 Percent Error Rate under Different Noise Levels

SNR (α)	Measurements	Filtered Signals			Noisy Signals		
		Error 1	**Error 2**	**Total**	**Error 1**	**Error 2**	**Total**
∞	$m = s$	1.31	0.46	1.77	N/A	N/A	N/A
10	$m = 10s + n$	4.37	0.98	5.35	4.53	3.97	8.50
1	$m = s + n$	4.75	0.10	4.85	9.14	5.25	14.39
0.1	$m = 0.1s + n$	5.21	0.08	5.39	15.34	4.98	20.32

The results showed that MFCCs performed well in a quiet environment (large SNR), yielding an error rate of only 1.8%. However, the accuracy of the algorithm dropped (error rate increases) as the noise level increased if the signals were not pre-processed with the modified ICA noise cancelling algorithm. When the signals were filtered with the modified ICA algorithm, the detection algorithm became less noise sensitive and the error rate remained constant (around 5%) for all noise levels considered.

6.3.2 Performance on Repair Patches

After years of service, bridge decks deteriorate due to mechanical and environmental loads such as the corrosion of steel reinforcement and freeze-thaw cycling. The deteriorated bridge decks need to be repaired to improve the condition and extend service life. Commonly used repair methods consist of different types of overlays: epoxy overlay, shallow concrete overlay, hot-mix-asphalt overlay with waterproofing membrane, and deep concrete overlay. Different overlays are applied to the damaged decks based on the extent of the deterioration. After repair, the bridge deck may experience two types of delamination damage: delamination may occur in the concrete of the bridge deck or in the repair patch. The application of the overlay and different damage pattern may affect the characteristics of the impact sound and the performance of the proposed algorithms needs to be tested under these conditions.

To perform such testing, a concrete slab was cast and repaired with different types of overlays in the laboratory. The thickness of the slab and the reinforcement layout conformed to a typical Michigan Department of Transportation (MDOT) bridge decks. The slab consisted of three different parts. The first part had a delamination at a 72.6 mm (3 inch) depth created with a plastic sheet during casting and the overlays were fully bonded to the underlying concrete. This part simulated the delamination in the concrete. The second part was solid concrete (full depth) with fully bonded repair overlays. This part was used to simulate the case of "no damage" in the bridge deck and overlay. The concrete in the third part was also solid, but plastic sheets were inserted between the repair overlays and the concrete slab. This simulated debonding between the repair patches and the slab. All patches were 203 mm by 203 mm (8 inches by 8 inches) in size.

Four different types of overlays were applied to repair the damage. Figure 6.11 and Figure 6.10 show a photograph and the plan view of the slab, respectively. The construction of the slab followed the MDOT guidelines for patching bridge decks. For convenience, the patches were labeled as follows: the first letter defines the condition of the concrete slab (D and S refer to the delaminated and solid slab, respectively); the second letter defines the condition of the overlays (U and B refers to unbonded and bonded overlay, respectively); and the last letter defines the type of the overlay (E, S, A and D refer to epoxy, shallow-concrete, asphalt with water-proofing membrane and deep concrete overlays, respectively). For example, DBA refers to a delaminated slab with a fully bonded asphalt overlay.

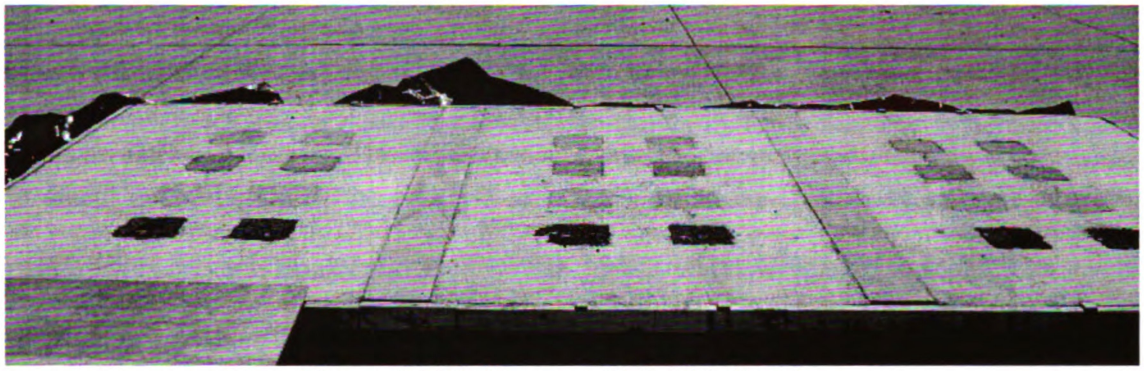
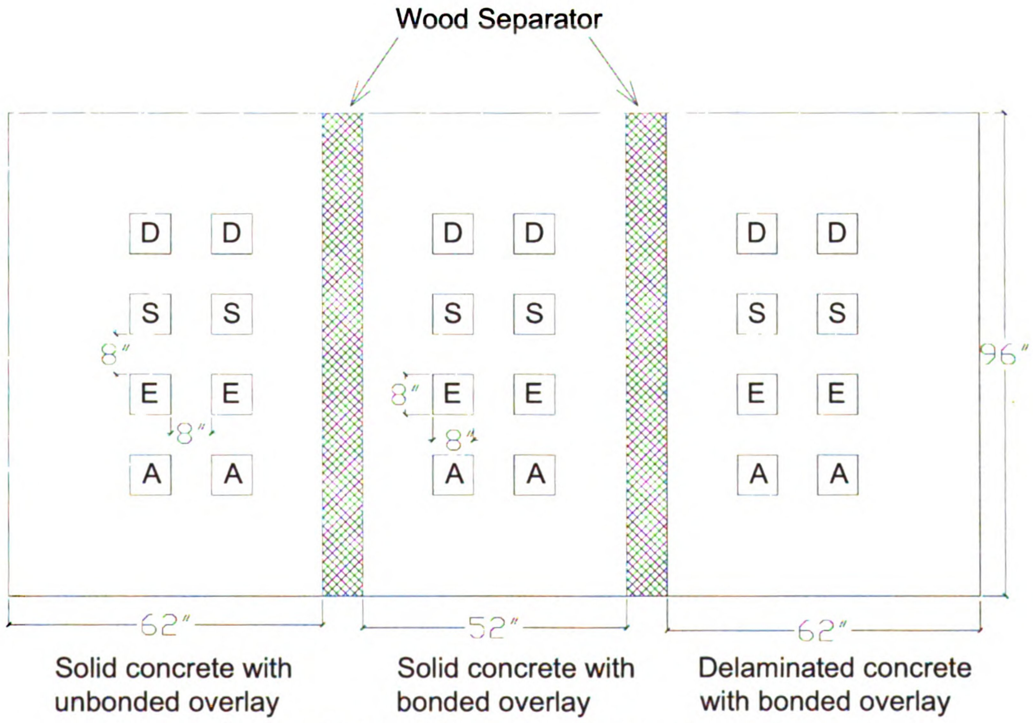


Figure 6.10 Laboratory Slab with Overlays


Wood Separator
D
S
E
A
8″
96″
62″
52″
62″
Solid concrete with unbonded overlay
Solid concrete with bonded overlay
Delaminated concrete with bonded overlay
Figure 6.11 Plan View of the Slab with Overlays

Signals obtained from Part 1 (delaminated slab with bonded patches) and Part 2 (solid slab with bonded patches) were used to check the performance of detecting the delamination of the slab with the existence of different overlays. There were two patches for each type of overlay on each part of the slab. Signals obtained on one of the patches were used as the training set and signals from the other patch were used to test the performance. Similar to previous case, the signals were collected on two different days to include variance between days. The results are shown in Table 6.2. The system was able to detect delamination in the concrete slab when an epoxy overlay or shallow concrete overlay was present. Epoxy and shallow concrete overlays were used to repair less severe or shallow damage and can be easily excited and penetrated by the impact. The performance on the asphalt concrete overlay was not as good as on the epoxy and shallow concrete overlays. This is likely due to the compliant visco-elastic properties of asphalt which change the propagation of acoustic waves inside the asphalt concrete as well as the wave reflection at the interface between the concrete and asphalt concrete. The special properties of asphalt change the sound and make it difficult for the AIDD system to detect the delamination. The error rate for the deep concrete overlay was very high. This is because the deep concrete overlay fully repairs the concrete and the signal from a deep concrete overlay on a delaminated slab was similar to the signal from patches on a solid slab.

Table 6.2 Detection of Slab Delamination with Existence of Overlays

Overlay Type	Solid	Delaminated	Error 1 (%)	Error 2 (%)	Total (%)
Epoxy Overlay	SBE	DBE	6.27	10.00	16.27
Shallow Concrete Overlay	SBS	DBS	4.54	11.27	15.81
Asphalt Overlay	SBA	DBA	11.57	14.05	25.62
Deep Concrete Overlay	SBD	DBD	28.01	27.27	56.28

After being repaired by different overlays, the bridge deck may experience debonding of the overlays after years of service. The capability of the system to detect the delamination of repair patches was evaluated by comparing the signal from Part 2 (solid slab with bonded patches) and Part 3 (solid slab with unbonded patches). The results are shown in Table 6.3. Similar to the previous comparison, the epoxy and shallow concrete overlays had good performance, indicating that the impact method can effectively detect the debonding of the repair patches for these overlays. The performance of the system on the asphalt concrete was again poor due to the same reason mentioned earlier. The system could not detect debonding of the deep concrete overlay due to its large thickness.

Table 6.3 Detection of the Delamination of Overlays

Overlay Type	Solid	Delaminated	Error 1 (%)	Error 2 (%)	Total (%)
Epoxy Overlay	SBE	SUE	3.63	1.36	4.99
Shallow Concrete Overlay	SBS	SUS	5.35	10.27	15.62
Asphalt Overlay	SBA	SUA	11.80	16.36	28.16
Deep Concrete Overlay	SBD	SUD	25.18	32.69	57.87

In summary, the system was able to effectively detect the delamination in the slab as well as the repair patches for epoxy and shallow concrete overlays. The performance on the asphalt concrete and deep concrete overlay was not satisfactory.

6.3.3 Performance under Field Conditions

To evaluate the detection algorithms under field conditions, tests were performed on two bridges near Mason, Michigan. Bridge 1 is located on Barnes Road over US 127 (shown in Figure 6.12) and Bridge 2 is on Sitts Road over US 127 (shown in Figure 6.13). Both bridges had concrete decks with delaminations. The concrete condition at several spots was first identified through traditional bar tapping (i.e., impacting the bridge deck using a steel bar and listening to the sound). The impact machine described in the Section 6.1 was then used to test these spots and the sound signals were collected using the data acquisition card.

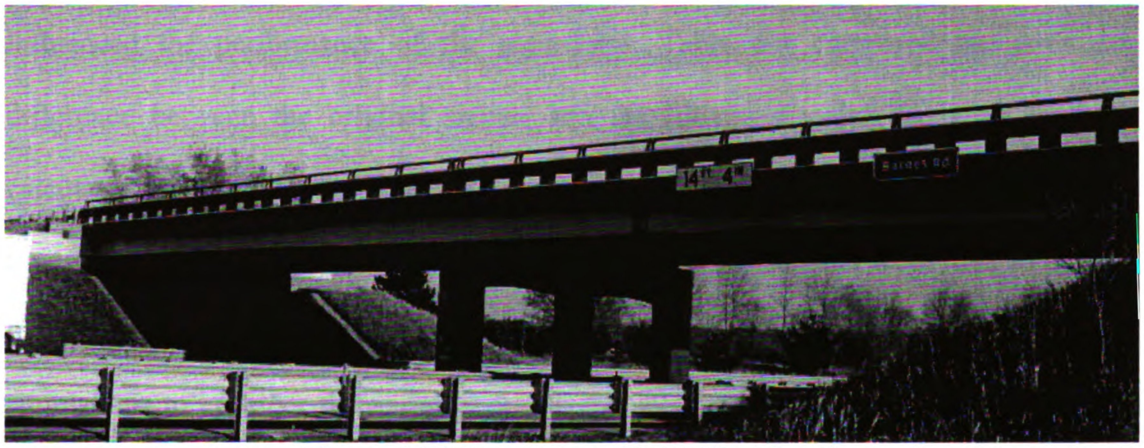


Figure 6.12 Barnes over US127 (Bridge 1)

Figure 6.13 Sitts over US127 (Bridge 2)

The analysis of the signal was performed offline to investigate the factors that influenced the performance of the algorithms. Table 6.4 compared the error rate difference between the original signals and the filtered signals. The error rate was calculated as described in Chapter 5: training signals were randomly selected from the training pool, and then the selected features and trained classifiers were used to classify the data in the testing pool. The error rate was calculated based on the average of 100 simulations to consider the variance due to the selection of the training data. Both the original signals and filtered signals gave very good results and there was no significant improvement with the filtered signals. This was because the noise level was low and filtering does not significantly enhance the signals.

Table 6.4 Error Rates of Original Signals and Filtered Signals

Signal Type		Error 1 (%)	Error 2 (%)	Total Error (%)
Bridge 1	Original Signals	0.05	0.20	0.25
	Filtered Signals	0.07	0.27	0.34
Bridge 2	Original Signals	0.48	0.21	0.69
	Filtered Signals	0.31	0.41	0.72
Bridge 1 and Bridge 2	Original Signals	0.38	0.75	1.13
	Filtered Signals	0.55	0.43	0.98

In the previous analysis, the training signals were randomly selected from the data pool. However, in real situations, the training signals can only be obtained from existing recordings. To investigate the effect of the training set, data obtained from Bridge 1 was divided into two groups: the first half (labeled as group A) and the second half (labeled as group B). Similarly, the data from Bridge 2 were divided into group C and group D. The training data and testing data were randomly selected from the recordings in the training pool and testing pool. The number of training data and testing data were 150 and 100, respectively, and were fixed for all cases. Table 6.5 shows the error rates obtained by using different training sets. Comparing Table 6.4 and Table 6.5, it can be seen that the error rates listed in Table 6.5 are higher than those in Table 6.4. This is because in Table 6.4, the training set contains all the information of the testing set and the classifier was tuned to this particular type of data. But in reality, the inspector does not have prior information about the bridge to be inspected and the training sets can only be selected from previous data which may not be fully representative, in which case the error rate will increase. It can be seen from Table 6.5 that when the number of groups in the training pool increased, the average error rate dropped for most groups (except for group

D, possibly due to variance in the data). If the training data was sufficient, the error rate can be lowed to a satisfactory level (around a 15% error rate for a single impact). As more bridges are inspected and more data becomes available, the performance of single impact detection will also improve. Multiple impacts at the same location could further reduce the error rate as shown in Figure 5.18. Even though the error rate of a single impact can be as high as 17.67%, the error rate may drop to less than 5% if 5 impacts were recorded according to Equation (5.31).

Table 6.5 Error Rates under Different Training Sets

roups in Training	Groups in Testing	Error 1 (%)	Error 2 (%)	Total Error (%)	Average Error (%)
B	A	1.66	4.12	5.78	
C	A	13.34	0.28	13.62	8.71
D	A	6.72	0.02	6.74	
BC	A	1.93	1.58	3.51	
BD	A	2.29	0.13	2.42	5.29
CD	A	9.80	0.15	9.95	
BCD	A	2.35	0.28	2.63	2.63
A	B	0.01	12.54	12.55	
C	B	14.18	5.72	19.90	14.60
D	B	5.39	5.96	11.35	
AC	B	1.45	6.60	8.05	
AD	B	9.82	5.38	15.20	10.13
CD	B	0.37	6.78	7.15	
ACD	B	1.40	6.03	7.43	7.43
A	C	0.03	24.64	24.67	
B	C	2.30	16.70	19.00	20.98
D	C	4.80	14.46	19.26	
AB	C	0.90	17.48	18.38	
AD	C	0.26	15.90	16.16	17.39
BD	C	2.38	15.26	17.64	
ABD	C	1.16	14.48	15.64	15.64
A	D	0.32	6.38	6.70	
B	D	0.57	9.54	10.11	8.77
C	D	7.65	1.84	9.49	
AB	D	0.24	8.50	8.74	
AC	D	15.86	2.26	18.12	14.33
BC	D	11.04	5.09	16.13	
ABC	D	10.29	2.96	13.25	13.25

6.4 Summary

This chapter described the development of an inspection and training system and then the performance of the system was verified using both experimental and field data.

The first section of the chapter focused on the development of the system. In the training process, a certain number of recordings were selected from the existing data files as training data. The features and the concrete condition were read directly from the file. The mutual information of each feature was calculated and compared. The features with a high value of mutual information were selected and used to train the RBF neural network classifier. The index of the selected features and the trained neural network were saved as a classifier for future use. During inspection, the data was first collected by the data acquisition system and the signal was filtered by the modified ICA algorithm. The features (MFCCs) of the filtered impact signal were calculated and the concrete deck condition was determined by the classifier obtained through the training process. The algorithms were incorporated using mixed language programming using LabVIEW, MATLAB and C++. A LabVIEW project was created to provide a graphical user interface (GUI) and perform the data acquisition.

The performance of the algorithms was verified using both experimental and field data. The results from the experimental data showed that the algorithms worked well under quiet conditions. However, the error rate increased with increasing the noise level. The introduction of the noise cancelling algorithm made the system noise robust and produced better results. Tests on a slab with different repair patches showed that the system can detect the delamination in the slab as well as the delamination of the overlay for epoxy and shallow concrete overlays. The performance on an asphalt concrete overlay was not satisfactory due to the special physical properties of asphalt concrete overlay. The system cannot detect the delamination for a deep concrete overlay because the impact was not strong enough to excite the mode that can differentiate the delaminated overlay from a solid slab. The field data indicated that the selection of the training data has an important effect on the performance. If the training sets are not representative of the test set, the error rate can be quite high. However, the error rate drops if a sufficient number of different training sets are used. Also, multiple impacts at the same location should further increase the accuracy. By recording five impacts on each location, the error rate should reduce to less than 5% even in the worst scenario shown in Table 6.5. The proposed automated system is considered to be fast and accurate enough to be used for field inspection.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

In this chapter, the research efforts to improve the sounding using an impact rod are first summarized. The conclusions obtained during the investigation are also included. The last section of the chapter provides several directions for further investigation and potential areas in which the research in this study can be beneficial.

7.1 Summary of the Study

Even though sounding methods are simple, fast and inexpensive for detecting delamination in concrete bridge decks, their performance can be undermined by traffic noise in adjacent lanes and the subjectivity of the operator. To improve the performance of the traditional sounding methods, this study addressed the two factors that reduce their performance. The sounding method used in this work was restricted to impacts by a rod because it produced cleaner signals than a chain drag. Several noise cancelling algorithms were investigated including spectrum subtraction, adaptive filtering, traditional independent component analysis (ICA) and modified ICA. The performance of the algorithms was evaluated based on the signal to distortion ratio (SDR). The results showed that the modified ICA had the best performance among all the candidate algorithms considered in this study and it was selected as the noise cancelling algorithm in this work.

After the noise signals and the impact signals were successfully separated, the features of the filtered signals were extracted. Different feature extraction algorithms were used to extract features of the signals: energy of sub-bands using the fast Fourier transform (FFT), energy of the wavelet packet tree, mel-frequency cepstral coefficients (MFCC), principal component analysis (PCA), and independent component analysis (ICA). The performance of the different algorithms was evaluated using repeatability, seperability and mutual information measures. The extracted features were further reduced based on the criterion of mutual information to select those features that best separated the sounds from solid and delaminated concrete slabs. Based on a weighted rank and the error rate, the MFCCs were selected as the best features for this study.

Delamination detection was posed as a classification problem and several candidate classifiers including linear and non-linear Bayesian classifiers, the support vector machine (SVM), the multi-layer perceptron (MLP) neural network, and the radial basis function (RBF) neural network were considered. Selected features of the signals in a training set were used to train the classifiers. The selected features and trained classifiers were used to classify the signals in the test set. The performance of the different classifiers was evaluated using the error (misclassification) rate. The results showed that the RBF network had the lowest error rate and hence it was selected as the classifier for delamination detection.

The selected noise cancelling and delamination detection algorithms were implemented in LabVIEW, MATLAB and C/C++ routines for use by general users. Performance of the system was verified using experimental data obtained in the laboratory and field data obtained from two bridges. The results showed that delaminations could be accurately detected by the proposed algorithms.

7.2 Major Conclusions

This study improved the performance of an acoustic-based non-destructive evaluation method by including noise cancellation, feature extraction and selection, and pattern recognition. The improvement in the performance was verified using data from laboratory experiments and field tests. The conclusions obtained from the investigation are summarized below.

Noise Cancelling Algorithms

Spectrum subtraction is very simple to implement, but it requires that the noise signal is short-term stationary, which is not guaranteed for traffic noise.

The recursive least square (RLS) adaptive filter can adaptively cancel the noise in the reference recording from the primary recording, but it will also cancel the source and lead to distortion in real situations because the source is also in the reference recording.

Independent component analysis (ICA) can separate linear mixtures without any prior information about the sources, but the records in real situations are convolutive mixtures and cannot be separated by traditional ICA.

A modified ICA was the only algorithm that works with real signals and was selected to cancel the traffic noise in this study. The pre-defined delay required in this method can be estimated through simple calculations.

The results also showed that SDR provides an adequate comparison among different algorithms.

Feature Extraction and Feature Selection

Five different feature extraction algorithms were evaluated against three criteria: repeatability, separability and mutual information. The values of repeatability and separability could not provide a consistence comparison due to ill-conditioning. Mutual information, however, provided a better indication of separability.

Different algorithms extract different features of the signals and features extracted by the same algorithm performed differently. The existence of features with poor separability may have a negative effect on the classification and mutual information was used as a criterion to eliminate unwanted features.

The weighted rank based on repeatability and separability, and the error rate based on a linear Bayesian classifier, was used to test the performance of features. The results from the weighted rank and the error rate agreed well and MFCCs were selected as the features to be used for delamination detection.

Pattern Recognition and Delamination Detection

Quadratic Bayesian classifiers had better performance than linear Bayesian classifiers but were sensitive to "redundant" features.

The quadratic support vector machine (SVM) had a slight better performance than the linear SVM for the data collected in this research. Both types of SVM were not sensitive to redundant features.

The performance of the multi-layer perceptron (MLP) network increased with the number of perceptrons in the hidden layers, but the increase was not significant if the number of perceptrons was greater than 4. The MLP was sensitive to the presence of redundant features.

The performance of the radial-basis-function (RBF) network increased with an increase in the number of neurons and the spread of the activation functions of each neuron. But the increase became insignificant after the number of neurons and the spread reached a certain value.

An RBF with $\sigma = 10$ and $N = 20$ had the best performance among all the classifiers and was used for delamination detection.

Performing multiple impacts on the same spot should increase the detection accuracy.

Algorithm Verification

The performance of filtered signals and original signals were both satisfactory under low noise levels, but only the filtered signals could provide good results under high noise levels.

The automated system developed in this reaserach was able to detect delamination of the slab and the overlay for epoxy and shallow concrete overlays. The system performance dropped for asphalt concrete overlays due to different acoustic properties. Delamination of deep concrete overlays cannot be detected by the method due to the depth of the delamination and limited impact energy.

Field tests indicated that the training data must be representative of the testing data to achieve good performance. In real situations, if the data available for training increases, the performance will improve.

7.3 Recommendations for Future Work

Even though the performance of the system was satisfactory for both experimental and field data, there is room for improvement.

1. The classifier selected in this reseach was not optimized. The parameters of the classifier such as the center and spread of each RBF can be optimized in a systematic way to further reduce the detection error. For example, the centers of the RBF can be optimized by an unsupervised training algorithm such as self-organizing map.

2. Detection in this research was formulated as separating the signal into two groups. The extent of the delamination may be able to be quantified by formulating the problem as the classification of multiple classes.

3. Acoustic methods that are effective with asphalt and deep concrete overlays need to be developed using different measurements such as acceleration or using different excitation methods such as hammers with controlled power.

The algorithms used in this study are general and can be used in other civil engineering areas.

1. Independent component analysis can be used in solving de-coupling or de-convolution problems. Principal component analysis can be used in solving eigen-value related problems. For example, vibration modes of structures can be used in health monitoring of structures and can be extracted from acceleration or displacement measurements using PCA or ICA.

2. Feature extraction of signals and detection algorithms described in this research can be used in the other areas of non-destructive evaluation, in which the task can be formulated as a classification problem.

Appendix A:

USER MANUAL

The delamination detection consists of two modules. The first module is for use at a bridge site where signals are recorded, analyzed and saved. The second module is for post-processing where the original signal and the filtered signal can be played back, directories can be added to or deleted from the training list, and the classifier can be trained using the data contained in the training list. This section described the installation and the use of the software.

A.1 Installation of the System

The installation of the system consists three parts: MATLAB Component runtime, inspection module and training module.

A.1.1 Installation of MATLAB Component Runtime

To run the *.dll* outside MATLAB environment, MATLAB component runtime (MCR) needs to be installed on the target computer. The installation file of MCR is named as MCRinstaller.exe. It should be noted that the version of the MCRinstaller should be same as the version of the MATLAB from which the *.dll* file is compiled. After installing the mcrinstaller.exe, add the following paths to the system variables:

```
%MATLAB_runtime\v**\bin\win32
%MATLAB_runtime\v**\runtime\win32
```

where %MATLAB_runtime is the home directory of the MCRinstaller and v** is the version of MCRinstaller. The system path can be edited by changing the PATH value in environmental variables by right clicking my computer -> properties -> advanced -> environmental variables.

A.1.2 Installation of the software

The installation of the software is simple. The user may click the setup.exe and follow the instructions to complete the installation process. The installation file will automatically install all required files to run the LabVIEW and the hardware drivers. The software consists of two modules: inspection and training, each module has its own installation package. The user may choose to install one component or both. When two components are installed on the same computer, the installation directory of the two modules must be the same to avoid potential errors.

A.2 Hardware Connection

The data acquisition card used in this project is USB 6211 from National Instuments. The connectors of the device are shown in Figure A.1. The signals are measured in the form of voltages across different connectors. In this software, three digital channels were used: Channel 1 was the voltage across connector 17 and 18, measureing the voltage across the electric magnetic; Channel 2 was the voltage between 19 and 20, measureing the signal from the primary microphone (the microphone under the base) and Channel 3 was the voltage between 21 and 22, measuring the signal from the reference microphone (the microphone that measures the ambient noise).

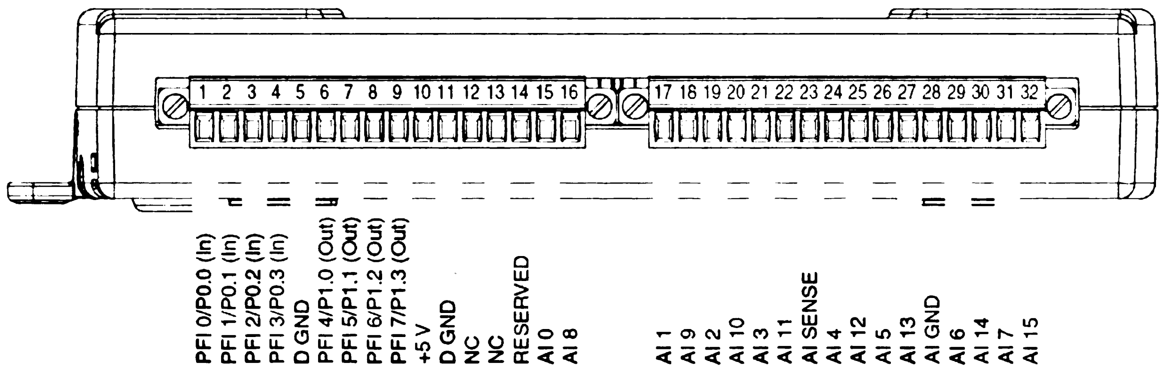
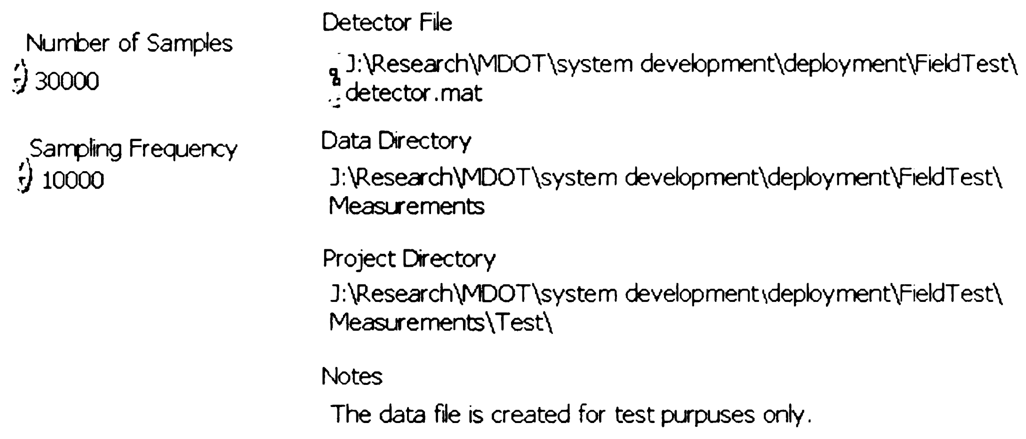


Figure A.1 Connectors of USB 6211

A.3 Bridge Inspection

The use of the bridge inspection system is quite simple. It contains three parts in the panel: a control part, an operation part and an indicator part. The control part shows the information about the path to save data, defines the number of samples for each test and the sampling frequency for the recording. The operation part has 5 buttons (DAQ & DSP, Play Original Signal, Play Filtered Signal, Save Data and Override & Save), one "DAQ ready?" light indicating whether the DAQ system is ready to use, and two text indicators showing the status of the tasks and deck condition determined by the system. Figure A.2 shows the two panels.


Number of Samples
30000

Sampling Frequency
10000

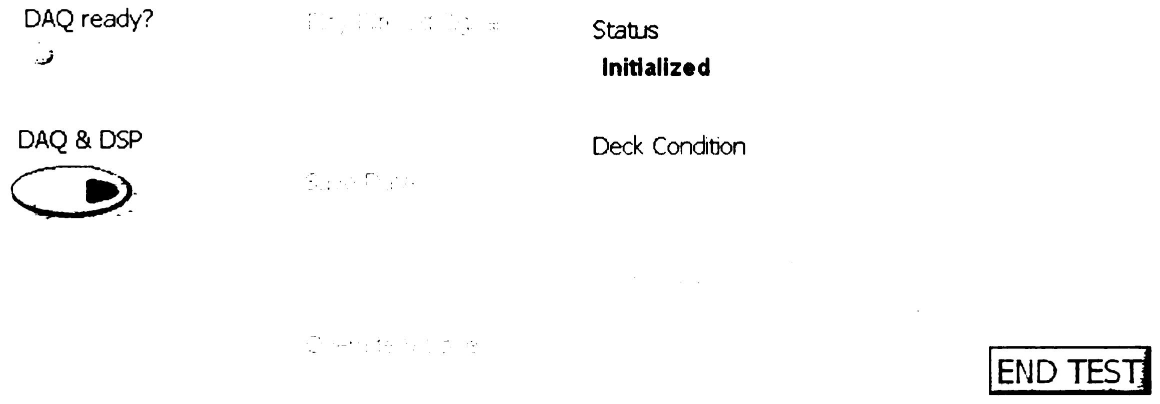
Detector File
J:\Research\MDOT\system development\deployment\FieldTest\detector.mat

Data Directory
J:\Research\MDOT\system development\deployment\FieldTest\Measurements

Project Directory
J:\Research\MDOT\system development\deployment\FieldTest\Measurements\Test\

Notes
The data file is created for test purpuses only.

Control Panel


DAQ ready?

Status
Initialized

DAQ & DSP

Deck Condition

END TEST

Operation Panel

Figure A.2 Panels for Inspection

When the program runs, the system will initialize the required *.dll* files. After the *.dll* files were initialized successfully, a window will pop-up asking for the name of the project. The data will be saved in a sub-directory with the project name. The user needs to input a new name that has never been used previously. If the project name already exists, the window will pop-up again.

After the project name is entered, the project directory is updated and the DAQ indicator turns green, indicating that the system is ready for data collection. The user may optionally input notes of up to 1000 characters if needed.

When the "DAQ & DSP" button is clicked, the system will start to record and process the signal. During this time the "DAQ ready" indicator will turn off, indicating that the DAQ is not available. The "Status" textbox will show "Recording and analyzing the signal...". After the signal has been analyzed, the status shows "Signal Analyzed" and the "DAQ & DSP" button is disabled, indicating that this operation is not available.

After the signal has been analyzed, the user has four options: (1) listen to the original signal; (2) listen to the filtered signal; (3) agree with the computer and save the data or (4) override the result provided by the computer and save the data. The user can click the corresponding button to select the desired option. The original signal and filtered signal can be played multiple times until the user is confident about the result. Both data saving buttons will be disabled after clicking either of them to prevent repeatedly saving the same data.

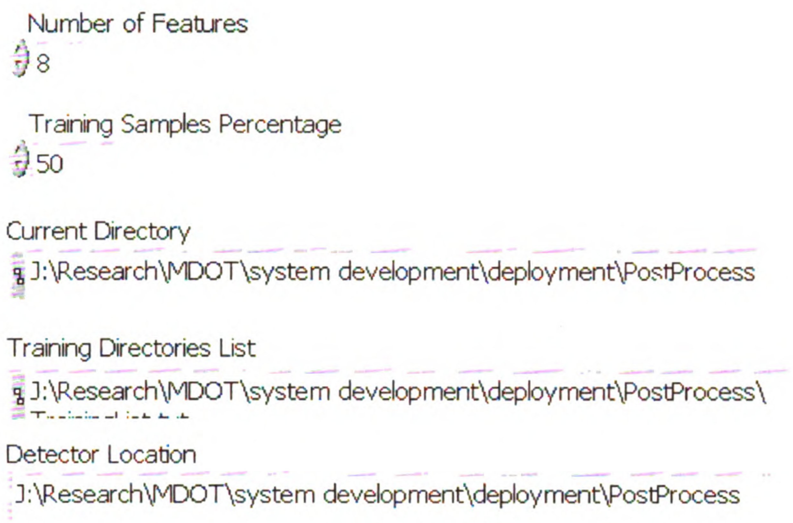
The user must click the "NEXT LOCATION" button to continue testing. After clicking this button, the "DAQ ready?" light will turn on and the system is ready for a new location.

The testing and data collection process is continued until all locations have been tested. After the last location is tested, the user will need to click "NEXT LOCATION" to turn on the "DAQ ready" light and then click "END TEST" to exit the program. If the "DAQ ready?" is not on, the system will not respond to the "END TEST" button.

After the program ends, the user will need to exit LabVIEW before a new program can be opened. This is due to the initialization of the *.dll* file created by MATLAB.

A.4 Post Processing

The panel of the post processing system consists of three major parts: the control part, the operation part and the file exploration part. In the control part, parameters such as the number of features, percentage of training samples, home directory for file explorer, location of training directory list and the location where the detector will be saved are speficified and shown. Default values are set for these parameters but the user may change them. The operation part consists of five buttons (Train Detector, Play Original Signal, Play Filtered Signal, Add Directory and Delete Directory), two lights indicating the readiness of the data and three text indicators (Deck condition for the data file, Notes in the data file and Status showing the operational status of the tasks). The file explorer consists of one multi-column list-box, an "UP" button and one text indicator showing the current path of the explorer. Directories for training are shown in a single-column list-box. These three parts are shown in Figure A.3.



Number of Features

8

Training Samples Percentage

50

Current Directory

J:\Research\MDOT\system development\deployment\PostProcess

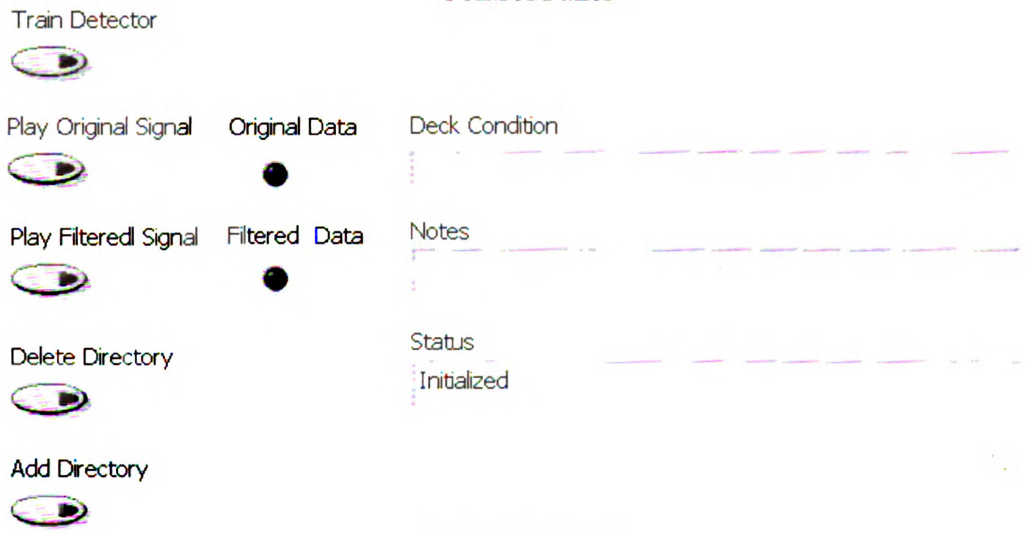
Training Directories List

J:\Research\MDOT\system development\deployment\PostProcess\

Detector Location

J:\Research\MDOT\system development\deployment\PostProcess

Control Panel



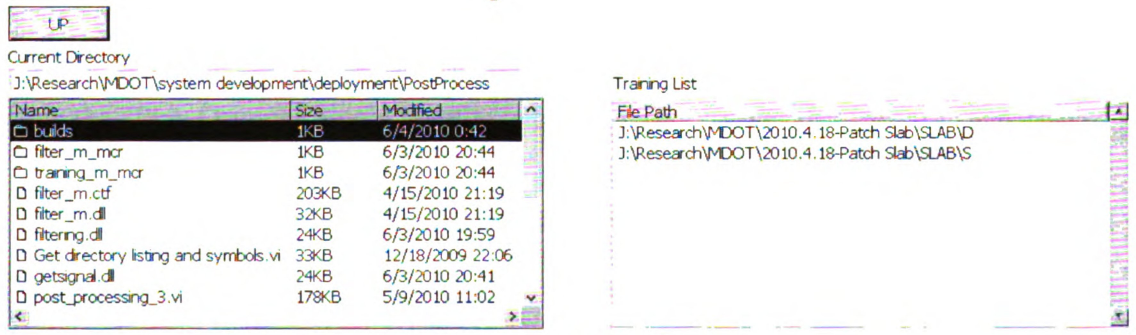
Deck Condition

Notes

Status

Initialized

Operation Panel



Current Directory

J:\Research\MDOT\system development\deployment\PostProcess

Name	Size	Modified
builds	1KB	6/4/2010 0:42
filter_m_mcr	1KB	6/3/2010 20:44
training_m_mcr	1KB	6/3/2010 20:44
filter_m.ctf	203KB	4/15/2010 21:19
filter_m.dll	32KB	4/15/2010 21:19
filtering.dll	24KB	6/3/2010 19:59
Get directory listing and symbols.vi	33KB	12/18/2009 22:06
getsignal.dll	24KB	6/3/2010 20:41
post_processing_3.vi	178KB	5/9/2010 11:02

Training List

File Path
J:\Research\MDOT\2010.4.18-Patch Slab\SLAB\D
J:\Research\MDOT\2010.4.18-Patch Slab\SLAB\S

Explorer Panel

Figure A.3 Panels for Post Processing

After the control parameters are provided, the user can start using the post-processing functions. To add a directory to the training list, the user needs to select the target directory by clicking on the file explorer. After selecting the target directory, click the "Add Directory" button and the selected folder will be added to the training list. The use of the file explorer is similar to use of "My Computer" in Windows. Double click a folder to see its contents and click "UP" to go back to the parent directory and see the contents.

To delete a directory from the training list, simply select the target directory in the "Training Directories" list-box and click "Delete Directory". The system will delete the folder after the user confirms the operation.

To listen to the original signal in a data file, the user only needs to browse with the file explorer and double click on the target data file. The system will read the data into memory. The deck condition and the notes from the data file will also be shown. The "Original Signal" light will turn on, indicating that the original signal is ready to be played. If the user double-clicks a file that is not a valid data file, the system will pop-up a dialogue box to warn the user and no operation will be performed. The user may re-click the "Play Original Signal" button to listen to the original signal again.

After reading the original data, the user can also listen to the filtered signal by clicking the "Play Filtered Signal" button. The computer will filter the signal and play the filtered signal when the processing is complete. The "Filtered Signal" will also turn on when the signal is filtered. The user may re-click the "Play Filtered Signal" button to listen to the filtered signal again. In this case, the filtered signal stored in the memory is played back.

After the the training list is finalized, the user must click the "Train Detector" button to train the classifier. A dialogue box indicating the path of the detector will pop-up once the training is completed. The system is now ready to be used in the field for further delamination detection.

The user can end the program by clicking the "END PROGRAM" button. The user will need to exit LabVIEW before a new program can be opened.

Appendix B:

SOFTWARE DOCUMENTATION

In order to implement the software in the form of an independent program with a graphical user interface (GUI), the algorithms were implemented by mixed-language programming via LabVIEW, MATLAB and C++. This appendix provides detailed information on how to program in the different environments including a sample wrapper code in C++, details of LabVIEW block diagrams for different modules, as well as MATLAB source codes.

B.1 Mixed-Languate Programming using LabVIEW, MATLAB and C++

B.1.1 Creating DLL from MATLAB

A MATLAB script can be converted into a dynamic link library file so that the script can be run from outside the MATLAB environment. This requires the optional MATLAB compiler. The command to convert a MATLAB script into a *.dll* file in the MATLAB environment is:

```
>>      mcc -W lib:filename1 -T link:lib filename2
```

filename1 is the name of the file containing MATLAB scripts, and *filename2* is the name of the *.dll* file to be created. Note that *filename1* and *filename2* cannot be the same to avoid potential conflicts.

The command in MATLAB will create several files in the current directory. Among them, *filename2.lib* and *filename2.h* will be used to create a wrapper as described later. *filename2.dll* and *filename2.ctf* will be required when running the program.

B.1.2 Creating C++ Wrapper

The data acquisition and GUI is implemented through LabVIEW. However, data cannot be communicated directly between LabVIEW and MATLAB, and therefore a wrapper is needed. In this project, another dynamic link library was created using C++ to act as a wrapper.

To create a *.dll* file in Visual Studio 6.0 (VS6), select *File* and then *New* from the VS6 menu. A new window will pop up, select *MFC AppWizard (dll)* from the *Project* panel. Input the name of the project (usually the name of the wrapper, for example, *wrappername*) and the directory in which you want to work. VS6 will create a series of files necessary to be compiled into a *.dll* file (the wrapper).

To call the *.dll* file from MATLAB in the C++ environment, several settings are needed.

1. Copy *filename2.h* and *filename2.lib* to the folder where the C++ source file is compiled.

2. Go to "Project Setting"->"C/C++". Choose category 'preprocessor', add the path of folder 'include' (present in the MATLAB root directory) under the option 'Additional include directories'. For MATLAB 2006a, this path is:

"%MATLAB\extern\include", where %MATLAB is the path where MATLAB is installed.

3. Go to "Project Setting"-> "Link" and choose category 'Input'. add *libmx.lib* and *filename2.lib* (created by MATLAB) in the 'Object library modules' options.

4. Add the path of folder 'Libraries' (present in the MATLAB root directory) in the 'Additional library paths' option. For MATLAB 2006a, the path is: "%MATLAB\extern\lib\win32\microsoft", where %MATLAB is the path where MATLAB is installed.

In the C++ source file, use the following format:

```
#include "filename2.h"
#include "mclcppclass.h"

_declspec (dllexport) int function_1(char *para1, double *para2……)
{
……
}
```

Detailed information about _declspec (dllexport) can be found online.

6. Compiling the C++ source file will create *wrappername.dll*.

B.1.3 Call Wrapper in LabVIEW

To run these DLL files in LabVIEW, copy *filename2.dll*, *filenmae2.ctf* and *wrappername.dll* to the same directory containing the LabVIEW VI files to be run. LabVIEW 8.2 was used for this project. In the LavVIEW block diagram, find "Call library function node" under "Function -> Connectivity -> Libraries and Executables" and put it on the block diagram. Double click on the node, enter the path of the *wrappername.dll* and select the name of the function. Set the parameters needed by the selected function and then click OK. Wire appropriate data in LabVIEW to the node and the LabVIEW will call *wrappername.dll* which calls *filename2.dll.*

B.1.4 Data Communication between LabVIEW, C++ and MATLAB

The data communication is shown in Figure B.1. The signal obtained from the DAQ node in LabVIEW is stored in rows, i.e., each row is a channel. Data in C++, however, is stored as a 1D array using a pointer. The data in LabVIEW needs to be transposed so that the data structure is compatible when data is transferred from LabVIEW to C++.

MATLAB reads data from C++ in rows. In order to keep the data structure, the following C++ code is used:

```
M_data=mxCreateDoubleMatrix(NC,NR,mxREAL);
memcpy(mxGetPr(M_data), C_data, NC*NR*sizeof(double));
```

The first function allocates a space in memory to save the data for MATLAB. The data in C++ is copied to the allocated memory using the second function. The documentation on these functions can be found online.

Results from MATLAB can be transferred back to LabVIEW using a similar method.

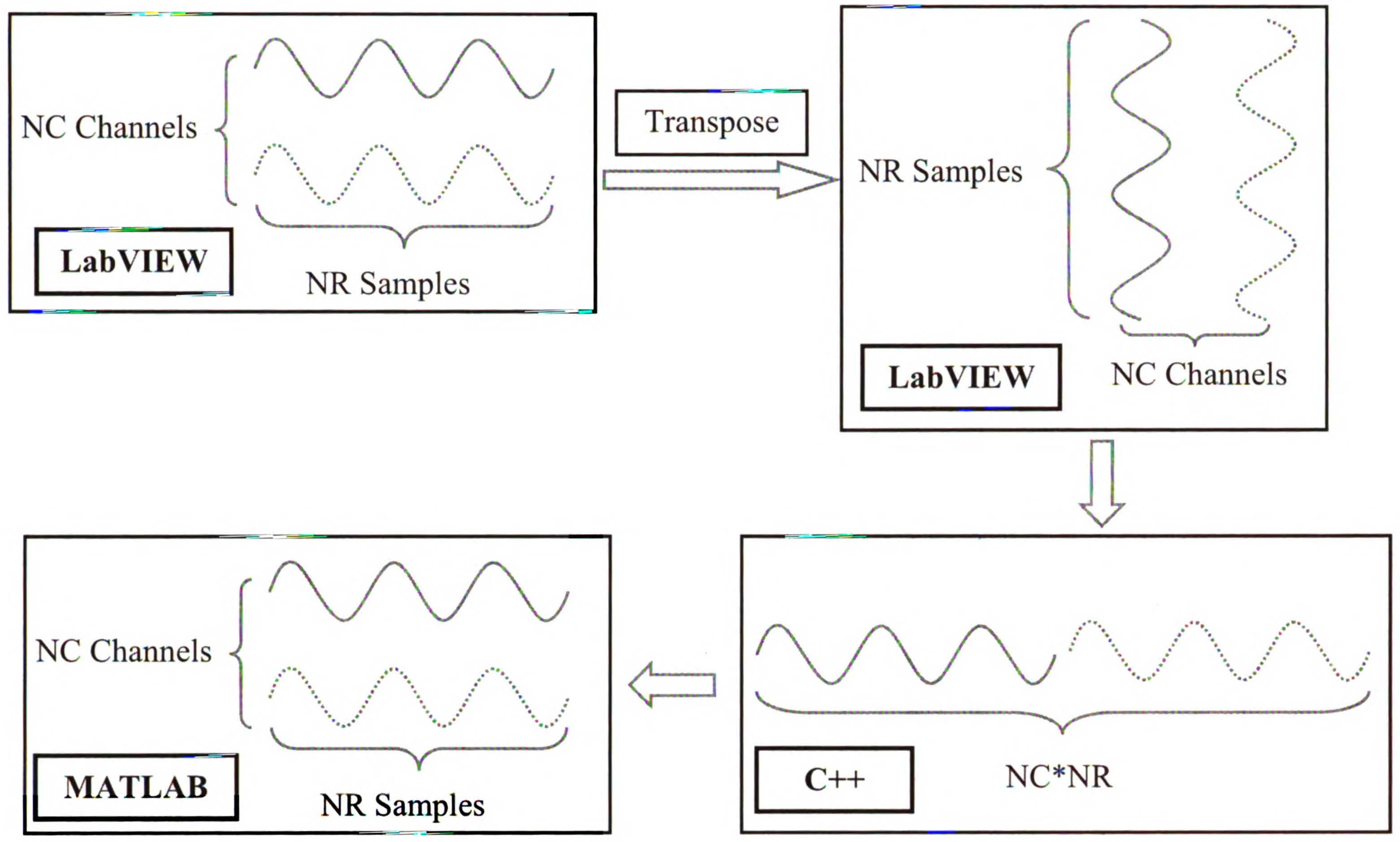


Figure B.1 Data Communication between LabVIEW, C++ and MATLAB

An example wrapper code is attached below:

```
#include <string.h>
#include "StdAfx.h"
#include "iostream.h"
#include "stdio.h"
#include "mclcppclass.h"
#include "inspection_m.h"

__declspec (dllexport) void inspection(double *data_in, double *size, double Fs, char
*crankfile, char *detectorpath, double *para_in, double *data_out, double *features,
double *para_out, double *index)
{
        mxArray *in1=NULL, *in2=NULL, *in3=NULL, *in4=NULL, *in5=NULL,
        *mout1=NULL, *mout2=NULL, *mout3=NULL,*mout4=NULL;
        double *out1=NULL, *out2=NULL,*out3=NULL, *out4=NULL;
        int *temp=NULL;

        // allocate memory for MATLAB data.
        in1=mxCreateDoubleMatrix(size[1],size[0],mxREAL);

        // copy data to MATLAB
        Memcpy(mxGetPr(in1),data_in,size[0]*size[1]*sizeof(double));
        in2=mxCreateDoubleMatrix(1,1,mxREAL);
        memcpy(mxGetPr(in2),&Fs,sizeof(double));
        in3=mxCreateString(crankfile);
        in4=mxCreateString(detectorpath);
        in5=mxCreateDoubleMatrix(1,8,mxREAL);
        memcpy(mxGetPr(in5),para_in,8*sizeof(double));

        // start calling MATLAB dll
        mlfInspection(4,&mout1,&mout2,&mout3,&mout4, in1,in2,in3,in4,in5);

        temp=(int*) malloc(2*sizeof(int));
        // get size of mout1
        memcpy(temp,mxGetDimensions(mout1),2*sizeof(int));

        // allocate memory to save mout1 from MATLAB
        out1=(double*) malloc(temp[0]*temp[1]*sizeof(double));
        // copy real part of the results from MATLAB to C++
        memcpy(temp,mxGetDimensions(mout2),2*sizeof(int));
        out2=(double*) malloc(temp[0]*temp[1]*sizeof(double));
        memcpy(out2,mxGetPr(mout2),temp[0]*temp[1]*sizeof(double));
        memcpy(features,out2,temp[0]*temp[1]*sizeof(double));

        memcpy(temp,mxGetDimensions(mout3),2*sizeof(int));
        out3=(double*) malloc(temp[0]*temp[1]*sizeof(double));
        memcpy(out3,mxGetPr(mout3),temp[0]*temp[1]*sizeof(double));
        memcpy(para_out,out3,temp[0]*temp[1]*sizeof(double));

        memcpy(temp,mxGetDimensions(mout4),2*sizeof(int));
        out4=(double*) malloc(temp[0]*temp[1]*sizeof(double));
        memcpy(out4,mxGetPr(mout4),temp[0]*temp[1]*sizeof(double));
        memcpy(index,out4,temp[0]*temp[1]*sizeof(double));

        // clear memory allocated for MATLAB variables
        mxDestroyArray(in1); in1=0;
        mxDestroyArray(in2); in2=0;
        mxDestroyArray(in3); in3=0;
        mxDestroyArray(in4); in4=0;
        mxDestroyArray(in5); in5=0;
        mxDestroyArray(mout1); out1=0;
        mxDestroyArray(mout2); out2=0;
        mxDestroyArray(mout3); out3=0;
        mxDestroyArray(mout4); out4=0;
}
```

B.2 LabVIEW Block Diagrams

As mentioned in Chapter 6, the system consists of two modules: inspection and training. This section includes detailed information about the LabVIEW block diagrams for the inspection module. The flow charts of the LabVIEW program were given in Chapter 6. This section provides detailed information about these modules.

B.2.1 Data Acquisition

The data from the microphone was acquired using the DAQ assistant module included in LabVIEW. Figure B.2 shows the LabVIEW block diagram for data acquisition. In the diagram:

Nsample is the number of samples per-channel.

Fs is the sampling rate.

Local variables are used for data transfer between modules.

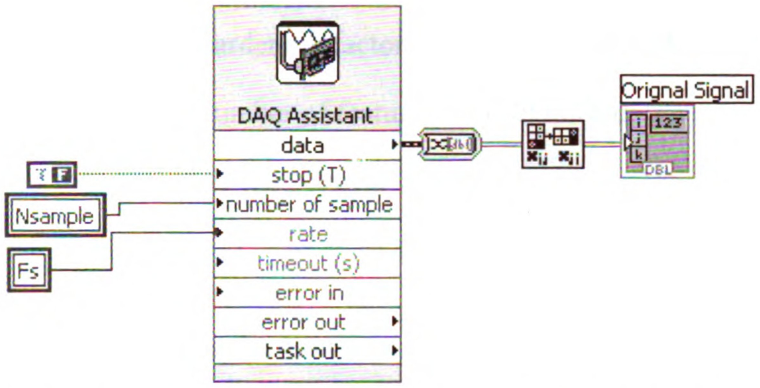


Figure B.2 Block Diagram for Data Acquisition Module

B.2.2 Signal Processing

After the signals were acquired from the microphones, modified ICA, impact identification feature extraction and delamination detection are lumped into one signal processing module as shown in Figure B.3. The kernel of this module is a *.dll* created from MATLAB. Another *.dll* created by C++ was used as the wrapper to provide a bridge between LabVIEW and MATLAB.

Inputs to the module are as follows:

Original Signal is the signal acquired by the DAQ assistant in the previous section;

Fs is the sampling rate;

detectorpath is the path of the detector;

para_in is an array containing the controlling parameters, where:

para_in (1) is the threshold for impact identification;

para_in (2) is the length of the impact (in seconds);

para_in (3) is the delay between the magnet voltage change and the impact;

para_in (4) is the maximum difference in the arrival time for microphones;

para_in (5) controls the ICA method used (see MATLAB codes);

para_in (6) is the hardening factor for the modified ICA;

para_in (7) is the number of features to be extracted;

Nsample is the number of samples per-channel.

Outputs to the module are as follows:

signal_out is the signal filtered by modified ICA;

features is the extracted features;

para_out is an array contaning the control factors in the data files, where:

para_out (1) is the deck condition;

para_in (2) is the sampling rate, same as *Fs*;

para_out (3) is the feature vectors, same as *features*;

para_out (4) is the number of impacts in the recorded signal;

para_out (5) is the number of samples in each channel, same as *Nsample*;

para_out (6) is the number of channels or microphones;

TYPE is the deck condition, same as *para_out (1)*;

index indicates the locations of the impacts.

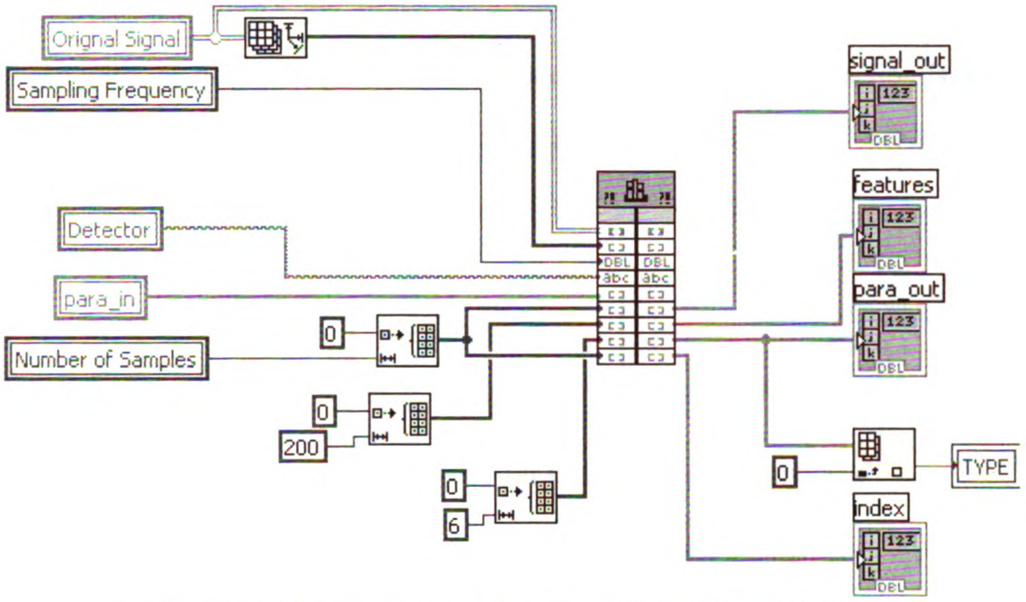


Figure B.3 Block Diagram for Signal Processing Module

B.2.3 Signal Playback and Data Save

After the signal is analyzed, a detection result will be provided by the computer, the user will have four options: (1) listen to the original signal, (2) listen to the filtered signal, (3) save the data, or (4) save the data but override the detection result provided by the computer.

Figure B.4 shows the block diagram for the original signal playback. The play waveform sub-VI that is included in LabVIEW is used to play the signals. Since there are multiple microphones, only the signal recorded by the microphone that is close to the impact point is played in this project. The original signal is stored in LabVIEW as a 2D array and needs to be separated.

Figure B.5 shows the block diagram for the filtered signal playback. The difference with the original signal playback module is that channel selection is no longer needed as the filtered signal has only one channel.

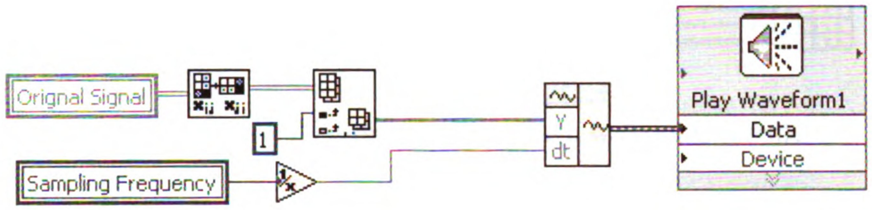


Figure B.4 Block Diagram for Original Signal Playback

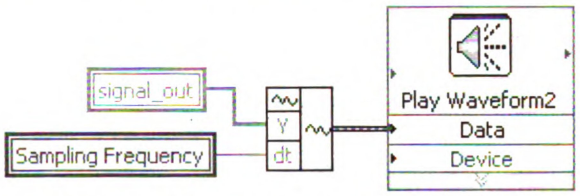


Figure B.5 Block Diagram for Filtered Signal Playback

Figure B.6 shows the block diagram for saving data without overriding the result provided by the algorithm. The data save module was implemented using a *.dll* created from C++. In this module:

projectdata controls the output directory of the data files;

notes contains any user input during the test;

para_out, features and *Original Signal* are the outputs of the signal processing module.

The name of the data file has two parts. The first part indicates the condition of the concrete deck: "S" for solid; "D" for delaminated and "U" for unknown deck condition. The second part is the index of the data file. For example: "S_5.txt" indicates that the data file was the fifth test saved for a "solid" deck condition during the inspection.

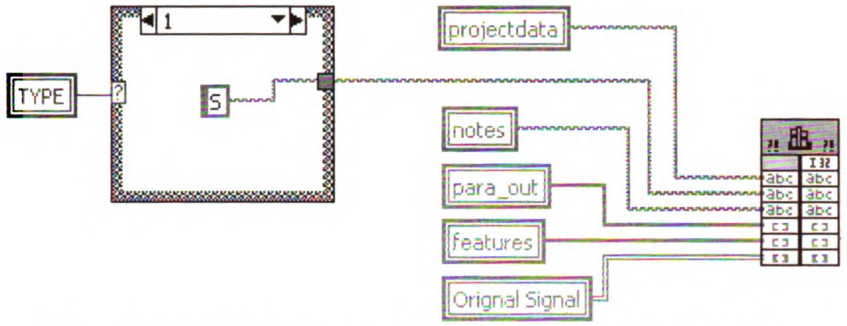


Figure B.6 Block Diagram for Data Save without Override

Similarly, Figure B.7 shows the block diagram for the data save module with override.

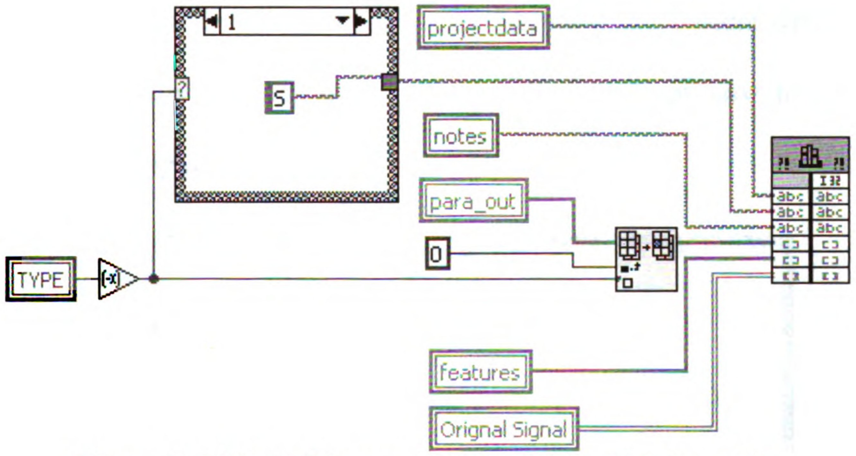


Figure B.7 Block Diagram for Data Save with Override

B.2.4 File Explorer

During post-processing the user will need to browse the computer and process the saved data. A file explorer function is provided for data selection. The content of a folder is shown by using the multi-column list-box in LabVIEW. The module that shows the contents of the current directory is shown in Figure B.8 where "My Computer" is the name of the multi-column list-box that lists the directory contents. A sub-VI "Get directory listing and symbols" from LabVIEW is used here.

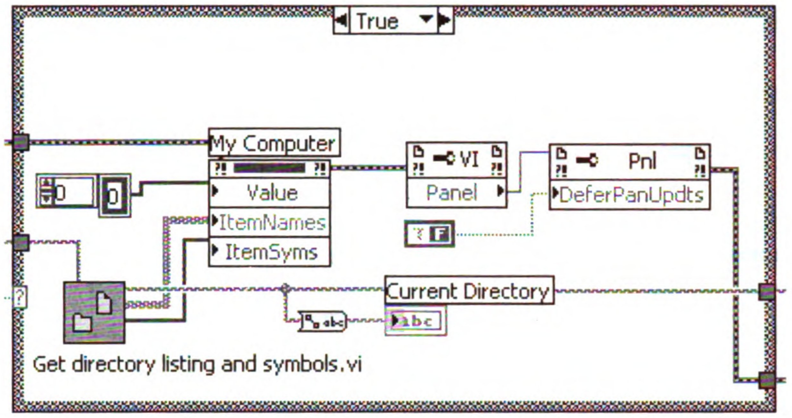


Figure B.8 List Contents of the Current Direct ory

An “UP” button is provided for the user to go back to the parent directory. When the “UP” button is clicked, the parent directory becomes the current directory and the contents are listed using the diagram in Figure B.8. Details of how to get the parent directory is shown in Figure B.9.

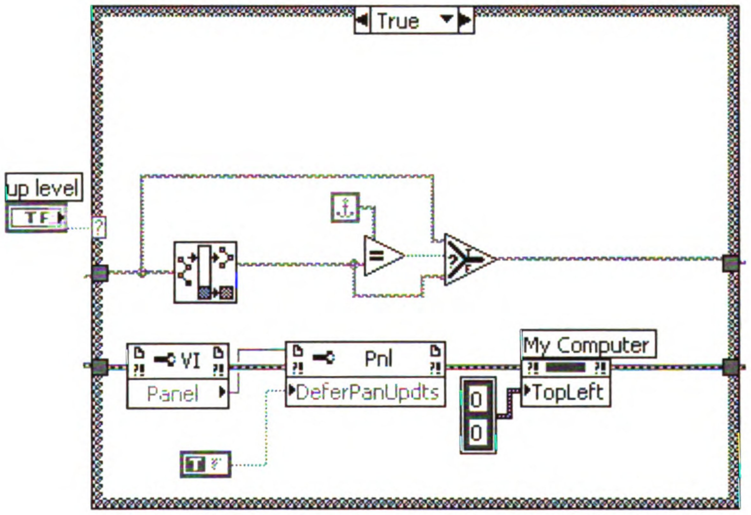
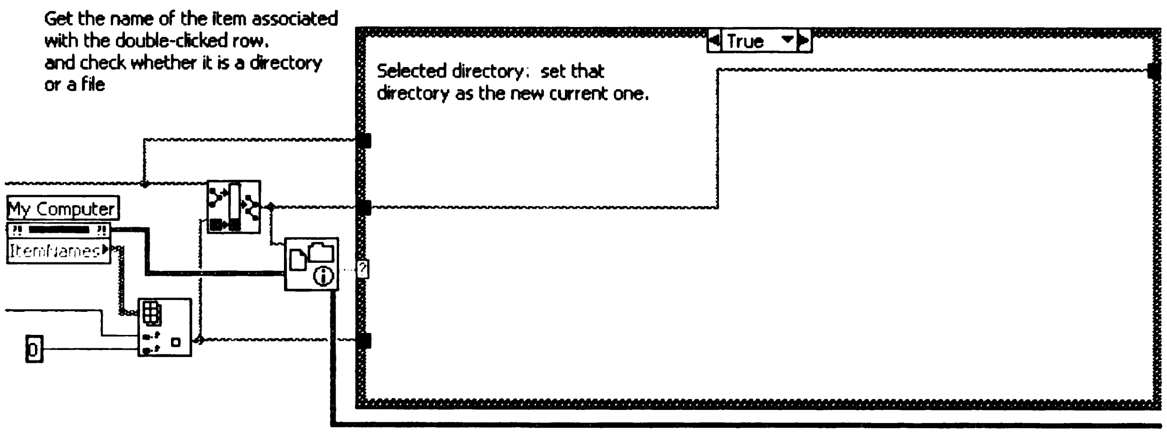
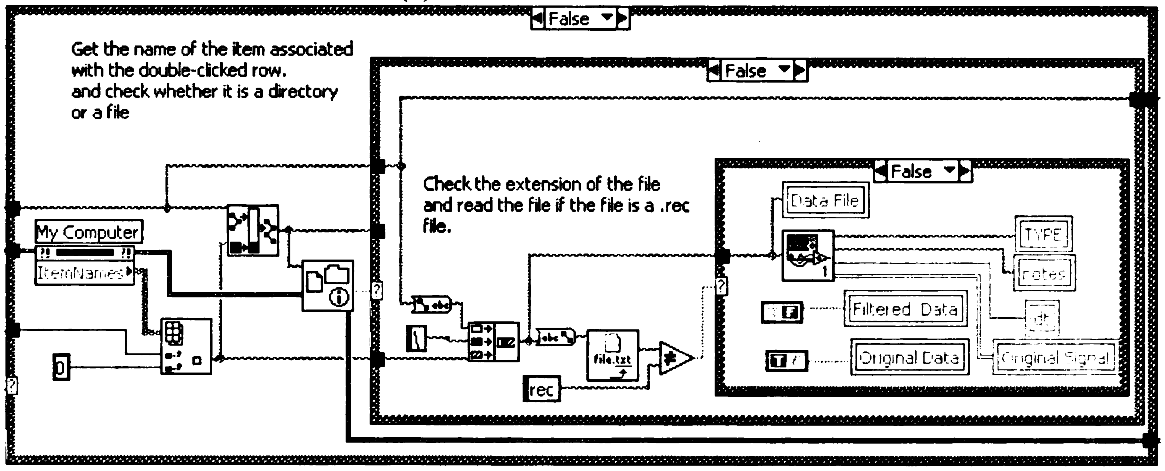


Figure B.9 Block Diagram for “UP” Button

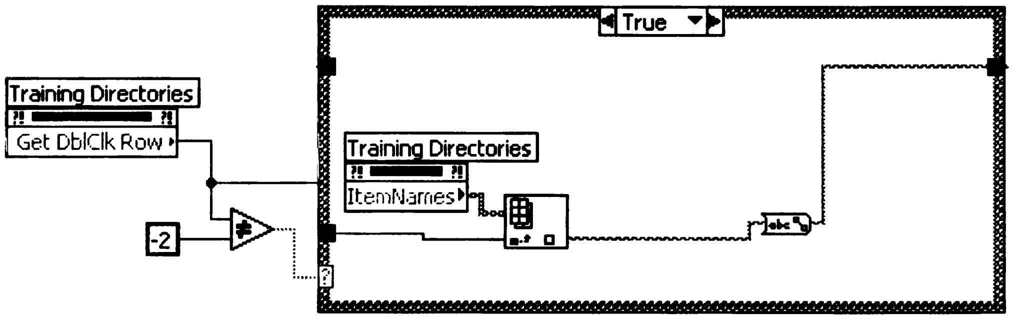
Similar to the GUI in the windows system, a directory is opened by double-clicking. This is accomplished by letting the current directory supercede the old directory as shown in the block diagram in Figure B.10 (a). When a file is double clicked, the software detects whether the file is a saved data file with the extension *.rec*. If the file is a data file, it is read as shown inFigure B.10 (c); if not, a dialoge box displaying "Invalid data format" will pop-up. An "Invoke Node" in LabVIEW is used to detect the operation of a double click.


Get the name of the item associated with the double-clicked row. and check whether it is a directory or a file
True
Selected directory: set that directory as the new current one.
My Computer
ItemNames

(a) Double Click on a Folder


False
Get the name of the item associated with the double-clicked row. and check whether it is a directory or a file
False
Check the extension of the file and read the file if the file is a .rec file.
False
My Computer
ItemNames
rec
Data File
TYPE
notes
Filtered Data
dt
Original Data
Original Signal

(b) Double Click on a File



(c) Double Click on the Training List

Figure B.10 Block Diagram for "Double Click"

B.2.5 Train Classifier

Training and saving of the classifier is performed in MATLAB. The classifier is saved as a *.mat* file which can be directly read by MATLAB or the *.dll* file created by MATLAB. A C++ wrapper is used for data communication purposes. The flow chart of the training process was shown in Figure 6.3. The LabVIEW diagram of the training process is shown in Figure B.11.

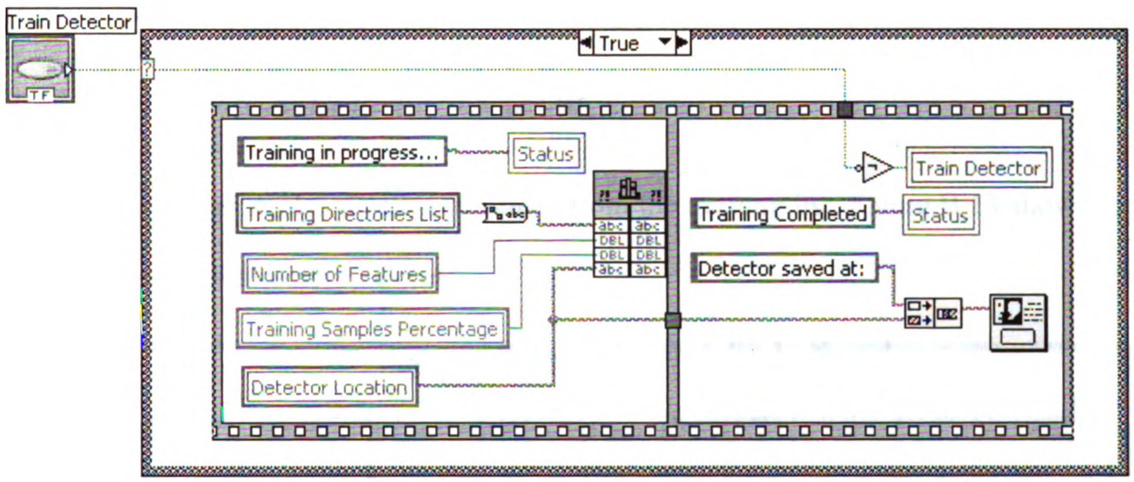
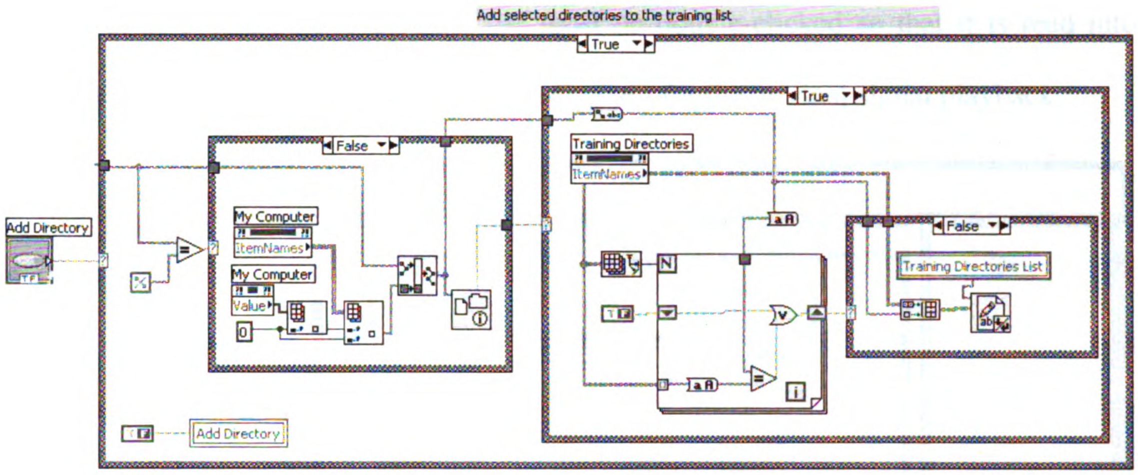


Figure B.11 Diagram of Classifier Training

B.2.6 Adding Directories to Training List

The user is provided with the option of selecting the directory for training so that appropriate training data for up-coming inspection projects can be selected. Figure B.12 shows the diagram for adding a directory to the training list. The software first obtains the name and path of the selected directory and converts them into a string. The file list is then searched to check whether the selected directory is already included in the list and if it is not, the path is written to the training list.


Add selected directories to the training list
True
False
Add Directory
My Computer
ItemNames
Value
Training Directories
Training Directories List
Figure B.12 Block Diagram for Adding Directory to the Training List

B.2.7 Deleting Directories from Training List

The user may also delete a directory from the training list. Figure B.13 shows the block diagram for directory deletion.

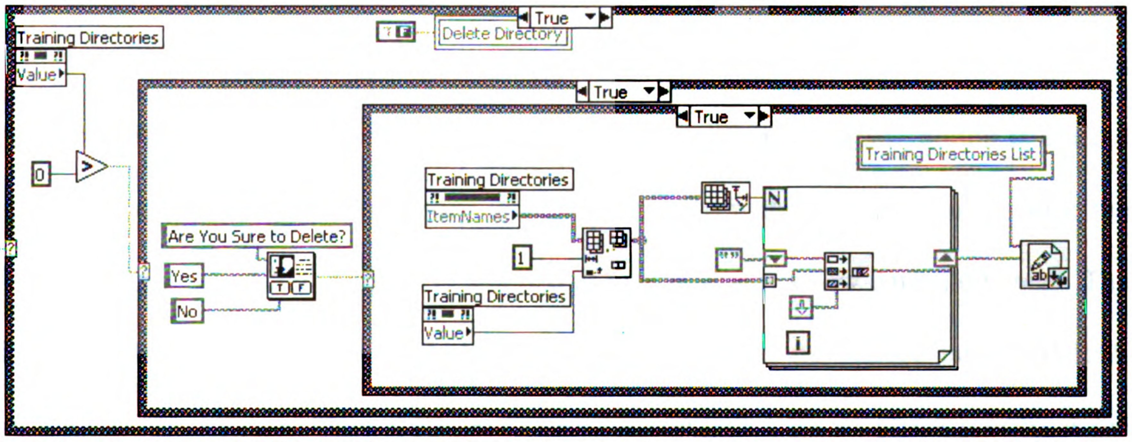


Figure B.13 Block Diagram for Deleting Directory from the Training List

B.2.8 Play Original or Filtered Signal

In order to select the appropriate training data, the user may wish to listen to the original signal recorded at the site and the filtered signal obtained by the noise cancelling algorithm described in Chapter 3. The software allows both types of signals to be played back. Before playing back, the data file must be double-clicked so that it is read into memory. Figure B.14 gives the LabVIEW diagram for the original signal playback.

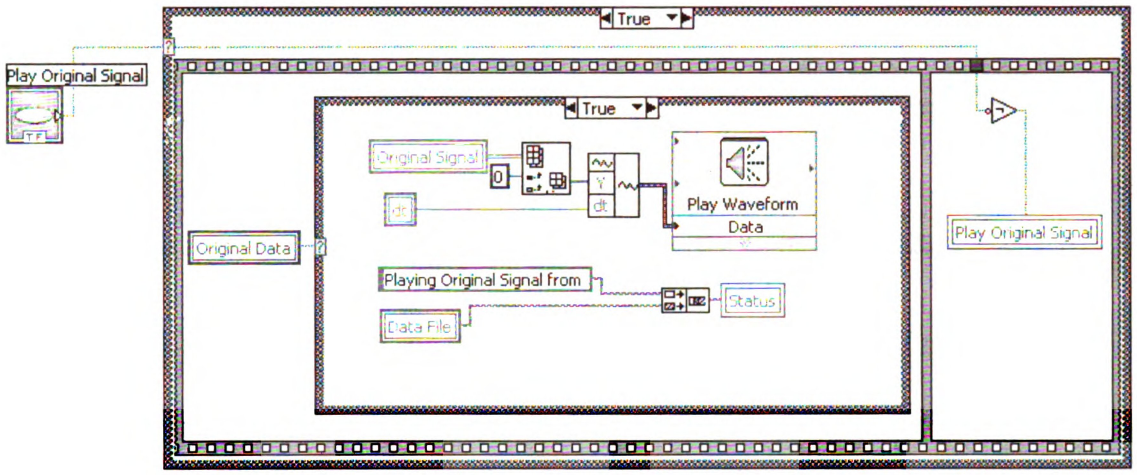


Figure B.14 Block Diagram for Original Signal Playback

To save storage space, the filtered signal is not saved in the data file but can be computed and played back during post-processing. To reduce the computational load, the signal is filtered only when the "Play Filtered Signal" is clicked. Once the signal is filtered, it is stored in memory so that when the "Play Filtered Signal" button is clicked again, the filtered signal in memory is played directly without having to perform the filtering again. The filtered signal is retained in memory until a new data file is read. The filtering process is performed in MATLAB and the result is transferred into LabVIEW through a C++ wrapper. Once the filtered signal is computed, the playback process is the same as for playback of the original signal. Figure B.15 shows the diagram for signal filtering.

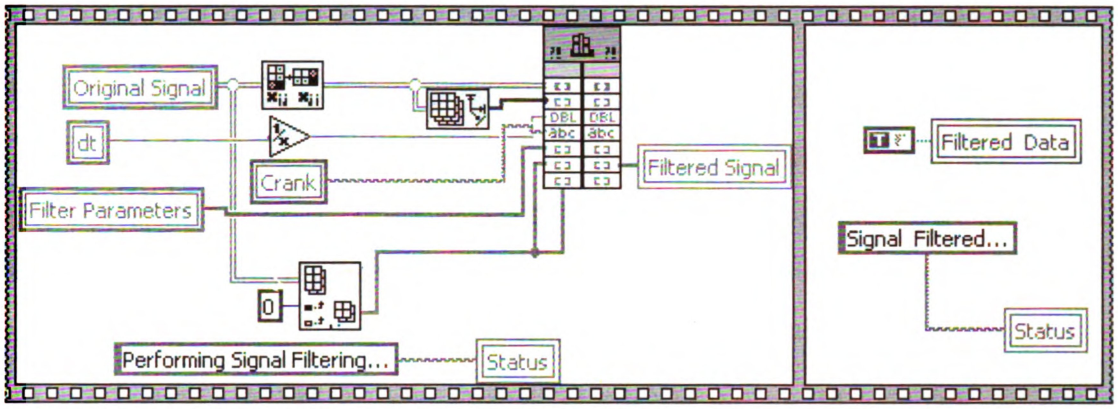


Figure B.15 Block Diagram for Signal Filtering

B.3 MATLAB Routines

Most of the signal processing and pattern recognition tasks were implemented using MATLAB. This section lists the code for the MATLAB routines used in this project.

B.3.1 Main Function for Inspection

```
function [shat features para_out
impactindex]=inspection(signal,Fs,crankpath,detectorpath,para_in)
% Assign control parameters
threshold=para_in(1);  % threshold=0.5; % recommended
Limpact=para_in(2);
delay=para_in(3);
dfilter=para_in(4)*Fs;
method=para_in(5);
Lblock=para_in(6)*Fs;
alf=para_in(7);
Nmfcc=para_in(8);

if size(signal,1)>size(signal,2)    % signal is saved in row
    signal=signal';
end

[Nchannel,Nsample]=size(signal);  % get size of the signal
[impactindex]=crankelimntv7(signal(1,:),Fs,threshold,Limpact,delay);
% impact identification
temp=find(impactindex==1);
shift=temp(1);
% signal filtering
[shat]=bssv46(signal(2:size(signal,1),:),dfilter,method,Lblock,shift,al
f);
% Feature Extraction
```

```
[features]=exfeatures(shat, impactindex,Limpact*Fs, Fs, Nmfcc);
% output parameters
Nimpact=length(features)/Nmfcc;
type=delamorno(features,Nmfcc,detectorpath);
para_out=[type,Fs,Nmfcc,Nimpact,Nsample,Nchannel];
```

B.3.2 Impact Identification

```
function [impactindex]=crankelimntv7(signal,Fs,threshold,L,d)


L=L*Fs;  % length of the impact signal
d=d*Fs;  % time difference betweeen the voltage change and the impact
signal=1/(max(signal)-min(signal))*(signal-min(signal)); % normalize
the signal to range [0,1]
index=[];
for i=1:length(signal)-1
    if (signal(i)>threshold && signal(i+1)<threshold)
        index=[index,i];
    end
end

if index(1)<d+1
    index=index(2:length(index));
end

if max(index)+L-d>length(signal)
    index=index(1:length(index)-1);
end
impactindex=zeros(1,length(signal));
if length(index)>1
    for i=1:length(index)
        impactindex(index(i)-d:index(i)-d+L-1)=1;
    end
end
```

B.3.3 Blind Source Separation

```
function [target] = bssv46(fm,L,method,Lb,shift,alf)
% Reference: Koldovsk, Z. and Tichavsk, P. Time-Domain Blind Audio
Source Separation Using Advanced Ica Methods

% Rearrange the signal such that each channel is one row
[Ls,Ns]=size(fm);
if Ls<Ns
    fm=fm';
    [Ls,Ns]=size(fm);
end
% Get part of the signal for training ICA
fmica=fm(1+shift:Lb+2*(L-1)+shift,:);
% Rearrange the training signal
for i=1:Ns
```

```
    for j=1:L
        M((i-1)*L+j,:)=fm(L-j+1:Ls-j+1,i)';
        Mica((i-1)*L+j,:)=fmica(L-j+1:Lb+2*(L-1)-j+1,i)';
    end
end
% Calculating ICA: codes of these functions can be found at:
% http://itakura.kes.tul.cz/zbynek/tddeconv.htm
switch method
    case 1
        w = efica(Mica);
    case 2
        w=bgsep(Mica,10);
    case 3
        w=bsep(Mica,10);
    case 4
        w=befica(Mica);
end
% Calculating similarity
d=similarity(w*Mica,L);
% Cluster the ICs using average linkage
T=clusterdata(d,'linkage','average','maxclust',Ns);
IC=w*M;
Lx=size(IC,2);
shat=zeros(Ns, Lx-L+1);

% Calculate weighting parameters for signal reconstruction
for i=1:length(T)
    d(i,i)=0;
    for j=1:Ns
        index=find(T==j);
        temp=sum(d(:,i));
        num=sum(d(index,i));
        den=temp-num;
        nmd(i,j)=(num/den)^alf;
    end
end
% Reconstruct the sources
for i=1:Ns
    x=(inv(w)*diag(nmd(:,i)))*IC;  % effect of source i
    for j=1:L
        shat(i,:)=shat(i,:)+x((i-1)*Ns+j,j:Lx-L+j);
    end
    shat(i,:)=shat(i,:)/max(abs(shat(i,:)));
end
shat=[shat,zeros(Ns,2*(L-1))]
sumshat=sum(shat.^2,2);
% Find the channel of impact signals
target=shat(find(sumshat==min(sumshat)),:);

function d=similarity(IC,L)
% computing similairy matrix between extracted ICs.
N=size(IC,2);
d=zeros(size(IC,1),size(IC,1));
for i=1:size(IC,1)
    for k=1:2*L+1
        Ci(:,k)=IC(i,k:N-2*L+k-1)';
    end
    temp2=Ci*inv(Ci'*Ci);
    for j=i+1:size(IC,1)
        cj=IC(j,L+1:N-L)';
        temp1=Ci'*cj;
        temp3=temp2*temp1;
        Dij=cj-temp3;
        d(i,j)=norm(Dij)^2;
    end
end
d=d+d';
```

B.3.4 Feature Extraction

```
function [features]=exfeatures(signal, index,Limpact, Fs, Nmfcc)
% Calculate features for each impact

% collect all impacts
impact=signal(find(index>0.5));
% calculate MFCC, detailed codes can be found at:
% http://labrosa.ee.columbia.edu/matlab/rastamat/
features=melfcc(impact,Fs, Limpact,Nmfcc);
% reshape feature vector
features=reshape(features,1,[]);
```

B.3.5 Delamination Detection

```
function r=delamorno(features,Nimpact,detector_path)
% The signal is classified as delaminated if the number of impact
% classified as delamination is great than that of solid
% r=1 means solid; r=-1 means delaminated

% load classifier (neural network) and feature index
load(detector_path);
rbf=detector.network; % RBF Classifier  .
ifeature=detector.feature; % Feature index

F=reshape(features,Nfeature,[]);
Nimpact=size(F,2);

if Nimpact==1      % If only one impact is recorded
    in_rbf=F(ifeature)';
    out_rbf=sim(rbf,in_rbf);
    if out_rbf<0.5 % if out_rbf is less than 0.5, the impact is delam
        r=-1;
    else
        r=1;
    end
else  % If more than one impact is recorded
    in_rbf=F(ifeature,:);
    out_rbf=sim(rbf,in_rbf);
    Ndelam=length(find(out_rbf<0));
    if Ndelam > floor(Nimpact/2)  % if Ndelam is less than half, it is
solid, r=1
        r=-1;
    else
        r=1;
    end
end
```

B.3.6 Main Training Program

```
function training(traininglist,Nfeature,TrainPercent,detector_path)
% Train and save the classifier
% Nfeature must less than Nfeautre=16
filename=traininglist;
fid=fopen(filename,'r');
filelist=[];

% Read directories from training list and obtain data file under each
% directory
while ~feof(fid)
    length1=size(filelist,2);
    folder=fgetl(fid);
    temp=ls([folder,'\*.rec']);
    length2=length(folder)+size(temp,2);
    maxleng=max(length1,length2);
    for i=1:size(temp,1)
        filelist=strvcat(filelist,[folder,'\',temp(i,:)]);
    end
end
fclose(fid);

% randomly select data files from file list
Nfile=size(filelist,1);
Ntraining=floor(Nfile*TrainPercent/100);
I=randperm(Nfile);
filelist=filelist(I,:);
TrainList=filelist(1:Ntraining,:);

traindata=[];
result=[];
for i=1:Ntraining
    tline=TrainList(i,:);
    [features, Nmfcc, type]=getfeature(tline);   % read features
    temp1=reshape(features,Nmfcc,[]);
    traindata=[traindata,temp1];
    result=[result,type*ones(1,size(temp1,2))];
end
[featureindex]=FE_MI(Nfeature,traindata,result); % feature selection
% get effective features of training data
input_train=traindata(featureindex,:);

% Training RBF using training data
RBF=newrb(input_train,result,0.01,10,20);
detector.network=RBF;
detector.feature=featureindex;
save([detector_path,'\detector.mat'],'detector');
```

```
function [features,Nfeature,type]=getfeature(filename)
% Read features from file

fid=fopen(filename,'r');
while ~feof(fid)
    tline=fgetl(fid);
    switch (upper(tline))
        case ('NUMBER OF FEATURES PER IMPACT:')
            tline=fgetl(fid);
            Nfeature=sscanf(tline,'%f');
        case ('TYPE:')
            type=str2num(fgetl(fid));
        case ('FEATURES:')
            tline=fgetl(fid);
            features=sscanf(tline,'%f');
            break;
        otherwise
            continue;
    end
end
fclose(fid);


function [B,I]=randomize(A,rowcol)
% randomize(A,0) randomize row vectors
% randomize(A,1) randomize colum vectors

if nargin == 1,
   rowcol=0;
end
if rowcol==0,
   [m,n]=size(A);
   p=rand(m,1);
   [p1,I]=sort(p);
   B=A(I,:);
elseif rowcol==1,
   Ap=A';
   [m,n]=size(Ap);
   p=rand(m,1);
   [p1,I]=sort(p);
   B=Ap(I,:)';
end
```

B.3.7 Feature Selection

```
function tf = FE_MI(N_feature,DataTrain,Result)
% Number of columns of DataTrain is Number of Samples
% Number of rows of DataTrain is the Number of Features
% Length of Result is the number of Samples
if size(Result,1)~=1
    Result=Result';  % Force Result to be a row vector
end
% Force the number of columns of DataTrain to be the same as the length
% of Result
if size(DataTrain,2)~=length(Result)
```

```
    DataTrain=DataTrain';
end

[DataTrain, I]=randomize(DataTrain,1); % randomize the column vectors
Result=Result(I);
for i=1:size(DataTrain,1)
    temp=DataTrain(i,:);
    % calculate mutual information, detailed codes for mutual
    % information can be found at:
    % http://www.cs.rug.nl/~rudy/matlab/
    MI(i)=information(temp,Result);
end
[p1,I]=sort(MI,'descend');
tf=I(1:N_feature); % feature selection based on mutual information
```

BIBLIOGRAPHY

[1] http://www.nationalconcretebridge.org/advantage.html; Accessed on Feb. 7., 2010.

[2] http://www.ascc.org/reportcard/2009/grades.cfm; Accessed on Feb. 7, 2010.

[3] Rens, K.L. and Transue, D.J., *Recent Trends in Nondestructive Inspections in State Highway Agencies*. Journal of Performance of Constructed Facilities, 1998. **12**(2): pp. 94-97.

[4] Rhazi, J., *NDT in Civil Engineering: The Case of Concrete Bridge Decks*. CSNDT Journal, 2000. **21**(5): pp. 18-22.

[5] Yehia, S., et al., *Detection of Common Defects in Concrete Bridge Decks using Nondestructive Evaluation Techniques*. Journal of Bridge Engineering, 2007. **12**(2): pp. 215-224.

[6] Carino, N.J. *The Impact-Echo Method: An Overview*. Proceedings of *the 2001 Structures Congress & Exposition*. 2001. Washington, DC.

[7] Mindess, S., *Acoustic Emission and Ultrasonic Pulse Velocity of Concrete*. International Journal of Cement Composites and Lightweight Concrete, 1982. **4**(3): pp. 173-179.

[8] Sansalone, M. and Carino, N.J., *Impact-Echo Method: Detecting Honeycombing, the Depth of Surface-Opening Cracks, and Ungrouted Ducts*. Concrete International, 1988. **10**(4): pp. 38-46.

[9] Bungey, J.H., *Sub-surface Radar Testing of Concrete: A Review*. Construction and Building Materials, 2004. **18**(1): pp. 1-8.

[10] Li, C.Q., et al., *Concrete Delamination Caused by Steel Reinforcement Corrosion*. Journal of Materials in Civil Engineering, 2007. **19**(7): pp. 591-600.

[11] ASTM-C4580, *Standard Practice for Measuring Delaminations in Concrete Bridge Decks by Sounding*. 2003, West Conshohocker, PA: ASTM Internatinal.

[12] Sansalone, M. and Carino, N.J., *Detecting Delaminations in Concrete Slabs with and without Overlays using the Impact-Echo Method.* ACI Materials Journal, 1989. **86**(2): pp. 175-184.

[13] Warhus, J.P., Mast, J.E., and Nelson, S.D. *Imaging Radar for Bridge Deck Inspection*. Proceedings of *SPIE*. 1995.

[14] Sansalone, M., *Impact-Echo: The Complete Story.* ACI Structural Journal, 1997. **94**(6): pp. 777-786.

[15] Ata, N., Mihara, S., and Ohtsu, M., *Imaging of Ungrouted Tendon Ducts in Prestressed Concrete by Improved SIBIE.* NDT and E International, 2007. **40**(3): pp. 258-264.

[16] McCann, D.M. and Forde, M.C., *Review of NDT Methods in the Assessment of Concrete and Masonry Structures.* NDT & E International, 2001. **34**(2): pp. 71-84.

[17] Zhu, J.,*Non-contact NDT of Concrete Structures using Air-Coupled Sensors,* Ph.D Dissertation, University of Illinois at Urbana-Champaign, 2006

[18] Ottosen, N.S., Ristinmaa, M., and Davis, A.G., *Theoretical Interpretation of Impulse Response Tests of Embedded Concrete Structures.* Journal of Engineering Mechanics, 2004. **130**(9): pp. 1062-1071.

[19] Popovics, J.S., *NDE Techniques for Concrete and Masonry Structures.* Progress in Structural Engineering and Materials, 2003. **5**(2): pp. 49-59.

[20] Martin, J., et al., *Ultrasonic Tomography of Grouted Duct Post-Tensioned Reinforced Concrete Bridge Beams.* NDT & E International, 2001. **34**(2): pp. 107-113.

[21] Malhotra, V.M. and Carino, N.J., *Handbook on Nondestructive Testing of Concrete*. 2nd ed. 1991, Boca Raton, FL: CRC Press.

[22] Bungey, J.H. and Millard, S.G. *Radar Inspection of Structures*. Proceedings of *the Institution of Civil Engineers: Structures and Buildings*. 1993.

[23] Clemena, G.G. and McKeel, W.T., *Detection of Delamination in Bridge Decks with Infrared Thermography*. Transportation Research Record, 1978(664): pp. 180-182.

[24] Brink, A., et al. *Application of Quantitative Impulse Thermography for Structural Evaluation in Civil Engineering - Comparison of Experimental Results and Numerical Simulations*. Proceedings of *Quantitative Infrared Thermography*. 2002. Croatia.

[25] Clark, M.R., McCann, D.M., and Forde, M.C., *Application of Infrared Thermography to the Non-Destructive Testing of Concrete and Masonry Bridges*. NDT and E International, 2003. **36**(4): pp. 265-275.

[26] Masad, E., et al., *Computations of Particle Surface Characteristics using Optical and X-Ray CT Images*. Computational Materials Science, 2005. **34**: pp. 406–424.

[27] Jandhyala, V.K. and Dasgupta, N., *Characterization of Air Void Distribution in Asphalt Mixes using X-Ray Computed Tomography*. Journal of Materials in Civil Engineering, 2002. **14**: pp. 122.

[28] Zelelew, H.M., Papagiannakis, A.T., and Masad, E. *Application of Digital Image Processing Techniques for Asphalt Concrete Mixture Images*. Proceedings of *the 12th International Association for Computer Methods and Advances in Geomechanics*. 2008. Goa, India.

[29] MDOT, *Instruction Manual for Bridge Deck Delamination Detector*. 1977, Michigan Department of Transportation: Lansing, MI.

[30] Henderson, M.E., Dion, G.N., and Costley, R.D. *Acoustic Inspection of Concrete Bridge Decks*. Proceedings of *SPIE*. 1999. Newport Beach, CA.

[31] Vincent, E., Gribonval, R., and Fevotte, C., *Performance Measurement in Blind Audio Source Separation*. IEEE Transactions on Audio, Speech, and Language Processing, 2006. **14**(4): pp. 1462-1469.

[32] MATLAB 7.2.0, The MathWorks Inc. , Natick, MA, 2009

[33] Boll, S., *Suppression of Acoustic Noise in Speech using Spectral Subtraction.* IEEE Transactions on Acoustics, Speech and Signal Processing, 1979. **27**(2): pp. 113-120.

[34] Widrow, B. and Hoff, M., *Adaptive Switching Circuits.* IRE WESCON Convention Record, 1960: pp. 96-104.

[35] Comon, P., *Independent Component Analysis: A New Concept?* Signal Processing, 1994. **36**: pp. 287-314.

[36] Haykin, S., *Neural Networks: A Comprehensive Foundation.* 2nd ed. 1999, Englewood Cliffs, NJ: Prentice Hall.

[37] Amari, S., Cichocki, A., and Yang, H.H., *A New Learning Algorithm for Blind Signal Separation.* Advances in Neural Information Processing Systems, 1996.

[38] Koldovsky, Z.T., P.; Oja, E.;, *Efficient Variant of Algorithm FastICA for Independent Component Analysis Attaining the Cramér-Rao Lower Bound.* IEEE Transactions on Neural Networks, 2006. **17**(5): pp. 1265-1277.

[39] Kay, S.M., *Fundamentals of Statistical Signal Processing, Volume I: Estimation Theory.* 1993, Englewood Cliffs, NJ: Prentice Hall.

[40] Koldovsk, Z. and Tichavsk, P. *Time-Domain Blind Audio Source Separation using Advanced ICA Methods*. Proceedings of *the 8th Annual Conference of the International Speech Communication Association*. 2007.

[41] Thomas, J.D., Y.; Hosseini, S.;, *Time-Domain Fast Fixed-Point Algorithms for Convolutive ICA.* IEEE Signal Processing Letters, 2006. **13**(4): pp. 228 - 231

[42] Cherkassky;, V. and Mulier, F.M., *Learning from Data: Concepts, Theory and Methods*. 1998, Hoboken, NJ: John Wiley & Sons Inc.

[43] John, G.H., Kohavi, R., and Pfleger, K. *Irrelevant Features and the Subset Selection Problem*. Proceedings of *the 11th International Confrence on Machine Learning*. 1994.

[44] Debnath, L., *Wavelet Transforms and Their Applications*. 2002, Boston, MA: Birkhäuser.

[45] http://en.wikipedia.org/wiki/Wavelet_packet_decomposition; Accessed on Jan. 20, 2010.

[46] Wicker, E.Z. and Fastl, H., *Psychoacoustic: Facts and Models*. 1990, Berlin, Germany: Springer

[47] Zheng, F., Zhang, G., and Song, Z., *Comparison of Different Implementations of MFCC*. Journal of Computer Science and Technology, 2001. **16**(6): pp. 582-589.

[48] Lee, S.M., et al., *Improved MFCC Feature Extraction by PCA-Optimized Filter Bank for Speech Recognition*. IEEE workshop on Automatic Speech Recognition and Understanding, 2001: pp. 49-52.

[49] Lee, J.H., et al. *Speech Feature Extraction using Independent Component Analysis*. Proceedings of *the IEEE International Conference on Acoustics, Speech and Signal Processing*. 2000.

[50] Torkkola, K., *Feature Extraction by Non Parametric Mutual Information Maximization*. The Journal of Machine Learning Research, 2003. **3**(7): pp. 1415-1438.

[51] Kay, S.M., *Fundamentals of Statistical Signal Processing, Volumn II: Detection theory*. 1998, Englewood Cliffs, NJ: Prentice-Hall.

[52] Rice, J.A., *Mathematical Statistics and Data Analysis*. 2007, Belmont, CA: Duxbury Press.

[53] Moon, T.K., *The Expectation-Maximization Algorithm*. IEEE Signal Processing Magazine, 1996. **13**(6): pp. 47-60.

[54] Theodoridis, S. and Koutroumbas, K., *Pattern Recognition*. 2nd ed. 2003, San Diego, CA: Academic Press.

[55] Bertsekas, D.P., et al., *Nonlinear Programming*. 1995, Belmont, MA: Athena Scientific.

[56] Tikhonov, A.N., *Solution of Incorrectly Formulated Problems and the Regularization Method.* Soviet Math., 1963. **4**: pp. 1035-1038.

[57] LabVIEW 8.2, National Instruments Corporation, Austin, TX 2006