Data Fusion for Detection of Faults in Slab-on-Girder Bridges

by

Marc Andrew Soiferman

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfilment of the requirements of the degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

University of Manitoba

Winnipeg

Data Fusion for Detection of Faults in Slab-on-Girder Bridges

BY

Marc Andrew Soiferman

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

Of

Master of Science

Abstract

This thesis reports the development of algorithms required to implement an automated fault detection system for structural health monitoring (SHM) applications for slab-on-girder bridges. The system is designed to detect girder cracking failures by modeling damage as a reduction in moment of inertia. To properly implement fault detection, three algorithms needed to be designed.

This thesis demonstrates how unwanted effects corrupt measurement data, which complicates extracting useful information. A preprocessing algorithm is proposed that is shown to remove two explicitly defined negative effects, while keeping the useful information untouched. Once these effects are removed, the measurement data is processed to extract vehicle events. Special care is taken to implement cascading of input data, to allow a system to process the large amount of data generated by an SHM system.

To determine structure failure, a classification algorithm was developed which was shown, through simulation, to be able to detect girder cracking failures based on features extracted from the event information. The classification algorithm is based on anomaly detection and uses an error count to guarantee a reported failure is a persistent failure.

Acknowledgements

First and foremost I would like to thank my advisor, Dr. Dean McNeill, for his input and dedication to this project. Without his direction and aid, this thesis would not have been completed. I would also like to thank the examining committee, Dr. Ehab El-Salakawy and Dr. Pradeepa Yahampath, for their time and contributions to this thesis.

Acknowledgement must be given to Dr. Aftab Mufi, Dr. Darshan Sidhu, and Dr. Ehab El-Salakawy for lending their expertise in the Civil Engineering aspects of this thesis, explaining concepts that were beyond my knowledge.

The assistance and input provided by my friends is greatly appreciated. Further acknowledgements are owed to Nico Danell for proof reading this thesis.

Special thanks are owed to my girlfriend, Mercedes Rich, for her support, understanding and encouragement throughout the entire process of completing this thesis. Special thanks are also owed to my family: my parents, Karen and Jacob, and my sister, Heather. They provided an environment and lifestyle that allowed for completion of this thesis. Without the support shown by them throughout my entire life it is impossible to predict if I would have been able to reach this point.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Structural health monitoring (SHM) is a developing field based on the idea of autonomously monitoring the health and stability of structures for the purposes of identifying the state of the structure. One use for this monitoring is failure detection and prevention. This project expands a pre-existing SHM system, composed only of a monitoring station, with the goal of developing algorithms for continuous monitoring and automated fault detection that can eventually be employed on-site to provide continuous, autonomous monitoring of the structure.

1.1 Motivation

The overriding motivation for this project is that there are no systems commercially available that can automatically and reliably interpret measurements from an existing SHM system to determine the health of a bridge. The current system for fault detection of a bridge involves sending an inspection team to inspect the bridge visually. This inspection typically occurs every two years. As a result, any faults that can grow from visually unnoticeable to complete failure within a two year period may be missed by the current inspection process.

Because of these shortcomings, having a system that can continuously monitor a bridge and determine the state without requiring human input is a vast improvement over standard methods. However, an automatic SHM system does not necessarily need to remove all human factors from monitoring and prevention to be considered a success. With artificial

intelligence (AI) at the level it is today, humans are still much more adept at recognizing where failures are in a bridge and how to repair them. If a monitoring system can simply alert a monitoring station that something is not right with the bridge, an expert can be sent to inspect the bridge to determine what caused the system to conclude there was a fault.

Another benefit of having an automated SHM system exists for bridges located in remote, rural areas. These areas can be extremely far from an urban centre, yet the highways still require bridges. As a result, it can be expensive to deploy an inspection team on a regular basis when the bridge is functioning properly. By installing an automated system, it can remotely monitor the bridge and report the status to a facility in an urban centre, preventing expensive deployment unless a problem is detected.

To provide further motivation for this project, there was very little research found that was concerned with modelling structure failure and implementing a classification method for determining if the modelled failure is occurring. This thesis can be used as a basis for methods on modelling structure failure as well as classifying this failure.

1.2 Scope

The scope of the project is to develop measurement processing algorithms that can be implemented in an SHM instrument. However, implementing a physical instrument that can be deployed is not part of the project. While there may be a number of potential failure modes for a structure, this project focuses only on identifying and determining failures that are detectable through the bridge girders. As well, the algorithm developed for the project should ideally differentiate between measurement system failure and structure failure to ensure robustness to imperfections in the existing system. The project should also be able to get information about different components of the system.

The entirety of the project is designed using the Red River-North Perimeter Bridge, discussed in Section 2.1. This bridge is used for all aspects of the project that require a physical structure. Empirical data is taken from the measurements gathered from the system installed on this bridge and simulations are done using computational models of this bridge. As a result, processing is designed to accommodate application-specific issues that arise from the bridge, and may not apply generally to all solutions. Ensuring that the solution is general and can apply to every bridge was not a priority; however, by examining the

base concepts and methods it is very possible that the solution can indeed be applied to a general structure with certain application-specific modifications.

1.3 Proposed Solution

The proposed solution to the problem is to develop an intelligent monitoring system that can continuously monitor a structure with the goal of identifying problems as they occur, rather than periodically monitoring the structure at fixed intervals. The system is composed of algorithms that monitor the strain channels supplied by the existing measurement system for anomalies. For the purposes of monitoring, the response to vehicle loading events was chosen as the main feature that would provide the required information about the state of the bridge. In order to use the events, they must first be identified and extracted, and then classified. Unfortunately, when observing the strain channels directly from the measurement system, some problems were discovered that made it difficult to extract events. These problems require using preliminary processing to clean the signal. As a result, there are three subsystems that make up the system: the preliminary processing subsystem, the event detection subsystem, and the event classification subsystem.

The preliminary processing subsystem is charged with removing unwanted and negative effects from the signal to facilitate event detection and classification. To do this, the unwanted effects must first be identified so that a method can be implemented to remove them. Chapter 3 provides an in-depth discussion of the algorithms and theory necessary to understand this subsystem.

The goal of the event detection algorithm is to separate the events from the rest of the signal so that they can be properly analyzed. For this to be possible, the algorithm must be able to identify automatically both where events are located in the channel, and which channels contain the same event for comparison purposes. Chapter 4 covers the event detection algorithm.

The final subsystem, the event classification system, is the main component of the project. The classifier is based on anomaly detection: that is, if something is found that does not fit a given model, it is considered anomalous, or failed. To properly design a classification system, it is first required to select descriptive features, and then select a failure model. Once the model is selected, simulations are required to determine how the model will show

a failure. It is only after this is done that a classifier can be developed and implemented. The classification subsystem is discussed in Chapter 5.

Figure 1.1 shows a block diagram of the proposed system.

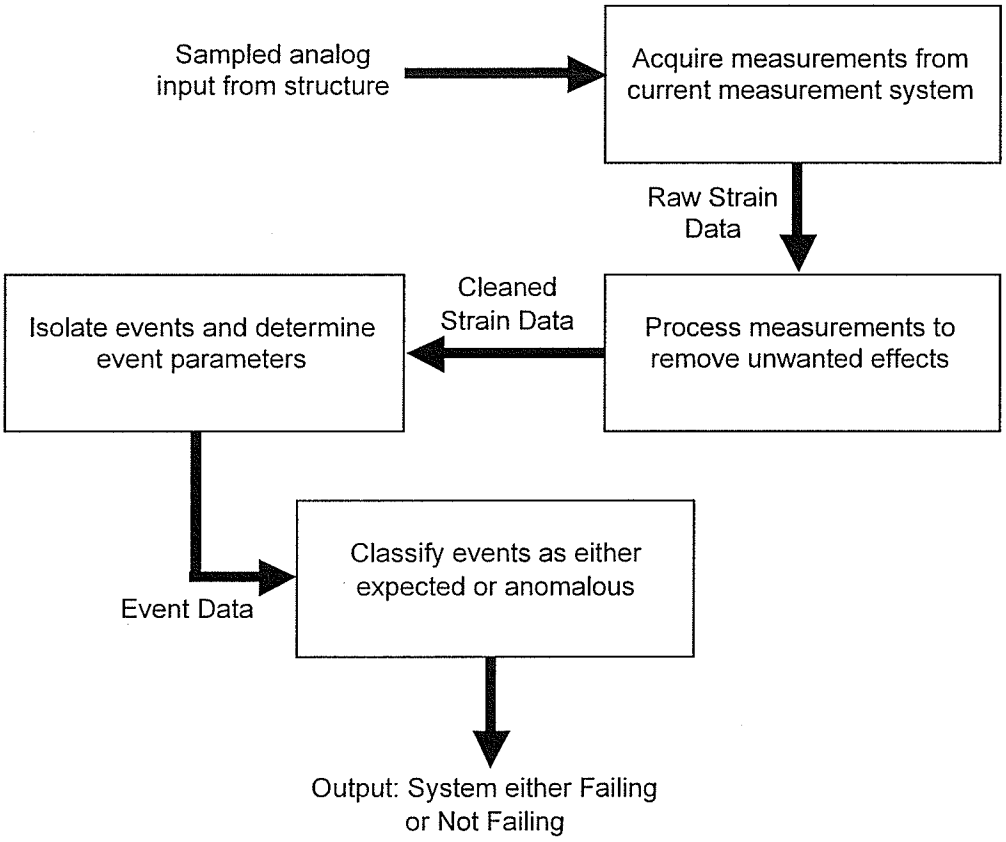


Figure 1.1: Block Diagram of Proposed System.

1.4 System Requirements

For the system to be completely successful, several requirements have to be satisfied. These are listed below in no particular order.

1. The system must be able to process x seconds of data in less than x seconds.
2. The system must be automated and reliable.
3. The system must not cause an alarm from sensor faults.
4. The system must not identify false positives or false negatives.
5. The system must have a way to isolate an evaluation component.
6. The system must be robust towards non-structural induced fluctuations.

Of these requirements, all were successfully implemented with the exception of Requirement 3. Development of the project revealed that the existing SHM measurement system provides no redundancy for individual sensors. Because of this, fatal sensor errors, where the sensor ceases to function and is unrecoverable, are unable to be separated from structure failures. The system does not, however, raise an alarm for non-fatal sensor failures provided the sensor recovers within a reasonable amount of time and begins to operate properly once again.

Chapter 2

Existing System

In order to completely understand the project and this report, some background information on the existing system that was the source for the project is required. The existing system is composed of three different components: the bridge, the measurement system, and the data acquisition unit. The required information will be provided for each of these components throughout this chapter.

2.1 The Bridge

The bridge used for this project, whenever a physical representation was required, is the Red River-North Perimeter Bridge. This bridge spans the Red River north of Winnipeg, Manitoba, Canada on Provincial Truck Highway 101 (PTH 101). A sensing system is installed on the portion of the bridge next to the abutment on the eastbound side, having a span of 25m and an average width of 12m. The bridge consists of a concrete deck supported by five steel girders with metal straps connecting them.

Figure 2.1 shows a basic drawing of a cross-section of the bridge that is being monitored, showing the girders and their positions relative to each other. It also provides numbering for the girders and indicates from which direction the bridge is being viewed. Since the left edge of the image is the edge that faces the centre of the bridge, this is the side closest to the westbound side of the bridge. This means that, viewing this cross-section, the direction of traffic is into the page. This girder numbering is very important, as all future references to girders are done using these numbers. As an example, the girder closest to the westbound

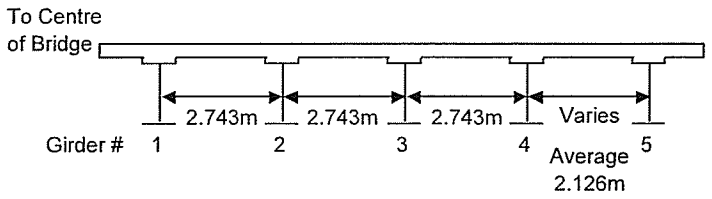


Figure 2.1: Cross Section of Red River-North Perimeter Bridge Showing Girder Information [1].
(Not To Scale)

side of the bridge is named Girder 1. The distance between Girders 4 and 5 is shown as being variable; this is because Girder 5 is angled. For the purposes of the project, the girder is assumed to be parallel to Girder 4 with a separation equivalent to the average separation given in Figure 2.1.

In addition to knowing the naming convention and the locations of the girders, knowledge of the location of the traffic lanes is also important. Figure 2.2 shows a cross-section of the bridge with the position of the three lanes marked relative to the edge of the bridge, and provides key distances from the girders to the lanes. Throughout this report, the three lanes are referred to as the *Passing Lane*, the *Normal Lane*, and the *Merging Lane*, naming conventions taken from highway driving conventions. It can be seen that the Passing Lane sits on Girders 1 and 2, the Normal Lane sits mainly on Girder 3, and the Merging Lane sits on Girder 4.

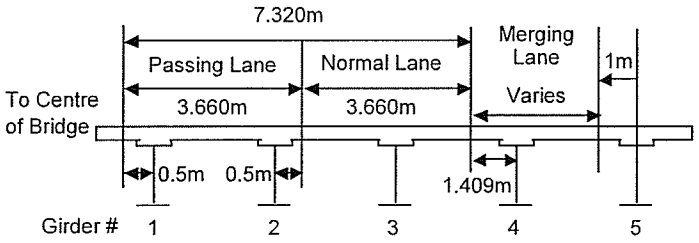


Figure 2.2: Cross Section of Red River-North Perimeter Bridge Showing Lane Information [1].
(Not To Scale)

2.2 Measurement System

The measurement system installed on the bridge is a combination of strain gauges and thermocouples. The strain gauges used are metal foil strain gauges (Vishay CEA-06-W250A-350) [2][3], and are sampled electrically by the data acquisition unit. Metal foil gauges have a variable resistance that is dependent on the area of the gauge. If a surface to which the gauge is attached bends, this area will stretch, which will then change the resistance. This change in area can be then converted into a strain value by using an appropriate factor, specific to the device.

There are thirty-six metal foil strain gauges installed on the Red River-North Perimeter Bridge. Of these, twenty are on the connecting straps between girders, denoted as *Sxx*, and sixteen are on the girders, denoted as *Gxx*. The locations of these sensors are shown in Figure 2.3. Girder gauges marked with an asterisk are mounted at the top of the girder.

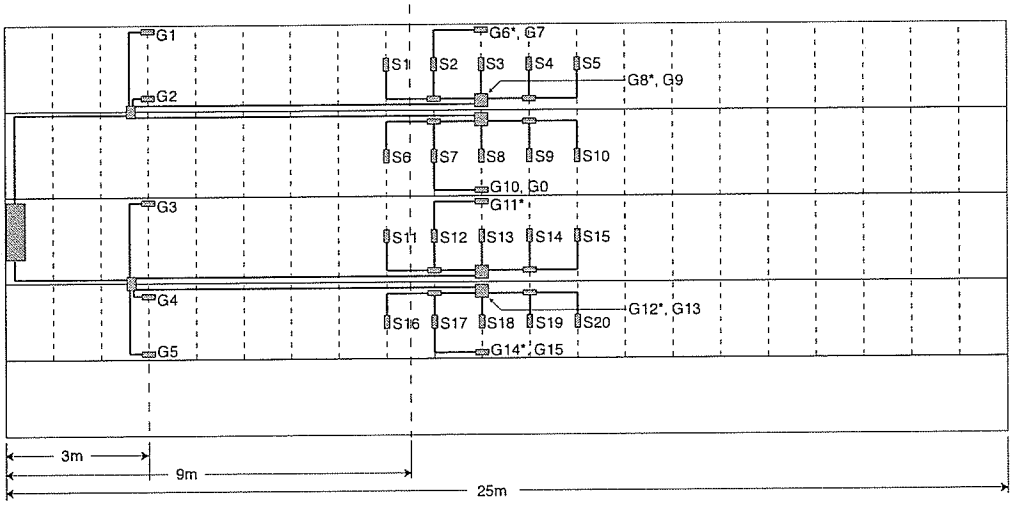
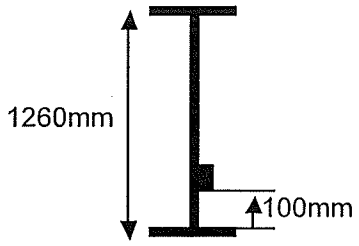


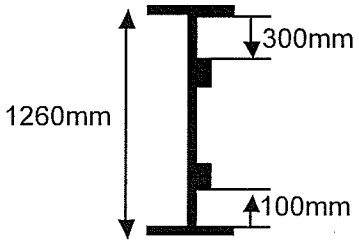
Figure 2.3: Location of Sensors on Red River-North Perimeter Bridge.
(Image Credit: Dean K. McNeill)

According to Figure 2.3, Girder 1 contains Sensors G1, G6 and G7, Girder 2 contains Sensors G2, G8 and G9, Girder 3 contains Sensors G3, G10 and G11, Girder 4 contains Sensors G4, G12 and G13, and finally Girder 5 contains Sensors G5, G14 and G15. Sensor G0 is a reference (also called a *dummy*) gauge; this means that is it theoretically unloaded

and was installed to provide a reference for interpreting the loaded sensor values. For this project, the sensors on the straps are ignored, and will not be discussed here. There are also sensors installed within the deck itself, however, like the sensors on the straps, these sensors are not used in the project. Each sensor generates its own data stream; this is referred to as the *data channel* for the sensor. This is used in the report to refer to the data stream the sensor generates, rather than the sensor itself. As an example, Channel 1 is the data stream generated by Sensor G1. Figure 2.4 shows a cross section of a girder at the abutment and the midspan, with the sensor locations shown for each location.



(a) Abutment Girder.



(b) Midspan Girder.

Figure 2.4: Cross Section of Girders, Showing Sensor Locations.
(Not to Scale)

In addition to the sensors, there are a number of thermocouples (Omega 5TC-TT-20-36) installed on the bridge to measure ambient temperature [2]. To work properly, thermocouples require a reference junction that is kept at a constant temperature. For this installation, the reference is kept with the data acquisition unit.

Figure 2.5 shows some of the sensors installed on the bridge; specifically four strap sensors (circled in red) and one girder sensor (circled in green).

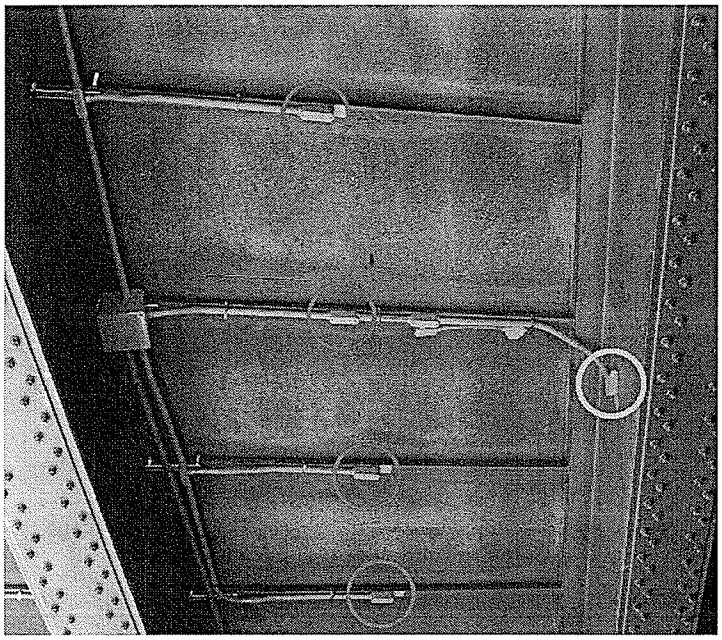


Figure 2.5: Installed Sensors on Red River-North Perimeter Bridge.
(Image Credit: Dean K. McNeill)

2.3 Data Acquisition Unit

The data acquisition system is that which interfaces between the measurement system and the outside world. In order to poll the measurement system, the data acquisition unit (DAQ) implements a Wheatstone Bridge in the quarter-bridge setup [4]. This resistor bridge uses matched resistors and determines the change in resistance of one resistor as a function of the output voltage [4]. Figure 2.6 shows a traditional Wheatstone Bridge. In this figure, R_G is the gauge resistance, modeled as a matched resistor with an error term ΔR.

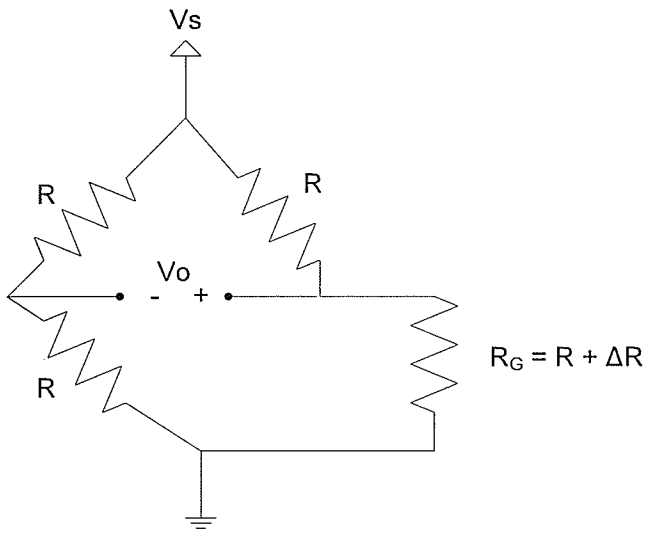


Figure 2.6: Wheatstone Bridge [4].

The output voltage is given in Equation 2.1 [4].

$$V_o = \left(\frac{R+\Delta R}{2R+\Delta R} - \frac{1}{2}\right) V_s = \left(\frac{2R+2\Delta R-(2R+\Delta R)}{4R+2\Delta R}\right) V_s = \left(\frac{\Delta R}{4R+2\Delta R}\right) V_s \quad (2.1)$$

The change in resistance, ΔR, is expressed as a function of the output voltage in Equation 2.2.

$$\Delta R = \frac{4RV_o}{V_s - 2V_o} \quad (2.2)$$

Using the Wheatstone Bridge, the strain gauges are sampled at a rate of 100Hz. The DAQ outputs a measurement file every 30 seconds, which consists of the prior 3000 samples of data. This file is then transferred as a binary file via an Ethernet connection to a computer hosting a database server. Once the measurement file is on the database server, it is converted to the database format and stored in the database in an easily accessible form. Figure 2.7 shows one of the nodes of the DAQ, responsible for monitoring a subset of the gauges.

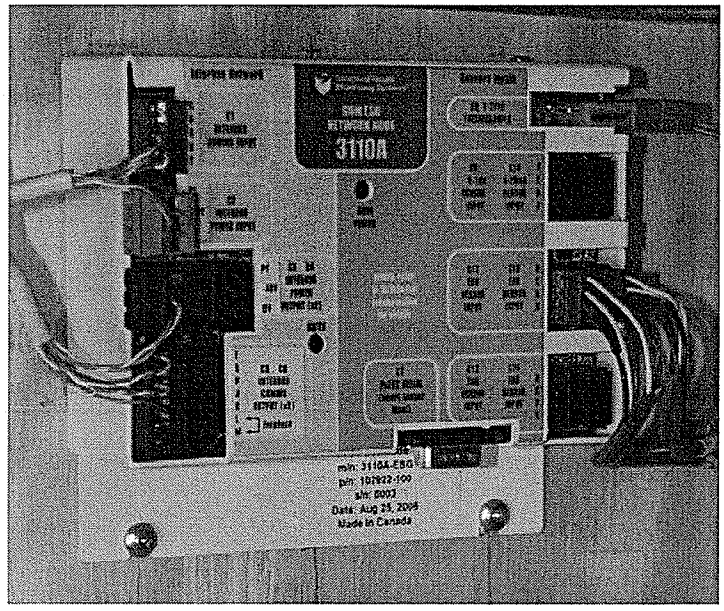


Figure 2.7: Sensor Node for Red River-North Perimeter Bridge.
(Image Credit: Dean K. McNeill)

The data acquisition system is located in a cabinet under the bridge, attached to the abutment. This cabinet contains all the computing instruments needed to monitor the Red River-North Perimeter Bridge. Some of these instruments cannot be exposed to freezing temperatures. As a result, a thermostatically controlled heater is installed in the cabinet, with the purpose of keeping the temperature above 0°C.

2.4 Problems With the System

All realized systems operate in a way that is different from the optimal, laboratory-based predictions. The existing system is no exception to this and some problems exist with the measurement system and the DAQ. These are that the sampling rate can cause problems, the thermocouple reference is not held at a constant temperature, and the strain output is extremely influenced by temperature.

The sampling rate causes problems because it may be under-sampling the strain signature generated as a vehicle traverses the bridge. This is described in detail in [5]. In short, by sampling at only 100Hz, it is possible to miss the true peak of the strain signature, giving a poor digital representation of the curve and potentially leading to false conclusions. This is in reality the only problem with the sampling rate. In this particular case, aliasing is not a concern since the sampling rate is high enough to avoid aliasing all but the peak component of an event, which allows for a good representation of the event shape.

A more serious problem is that the thermocouple reference is not held at a constant temperature. As a result, the measured temperature from the thermocouples is different from the true ambient temperature: it is affected by both changes in the cabinet temperature, as well as changes in the ambient temperature. This leads to serious oscillatory problems in the thermocouple output when the heater activates during the colder periods of the year.

The most serious problem with the current system is in the implementation of the Wheatstone Bridge. By using a quarter bridge, the only part of the resistor bridge that is exposed to the environment is the strain gauge. Since the Wheatstone Bridge attempts to use three matched, high precision resistors to determine the variation in gauge resistance, having only the gauge resistor exposed to the ambient temperature leads to temperature induced fluctuations in the strain similar to that described in [6][7][8]. While this may not be a serious concern, it is compounded by the presence of the heater. In the winter months, the heater causes the cabinet temperature to oscillate, resulting in a severe temperature mismatch in the Wheatstone Bridge. These rapid fluctuations pose a serious problem to any attempted investigation of the signals. Figures 2.8 and 2.9 demonstrate the affect the heater has on recorded strains. Figure 2.8 is a plot of 24 hours of data taken from the Red River-North Perimeter Bridge on July 4^{th}, 2007 (*the summer*). Figure 2.9 is a similar plot of data taken from the same bridge on January 4^{th}, 2008 (*the winter*). Both Figures 2.8 and 2.9 show the unloaded dummy channel and how it compares to the temperature.

In the summer, the strain data tends to track the temperature in a slow, almost linear fashion. However, in the winter, the strain varies wildly due to the temperature oscillations caused by the heater turning on and off. The variations in the winter are so severe that they can be noticed even on a small time scale. This causes issues, described in more detail in Section 3.2, when attempting to examine events.

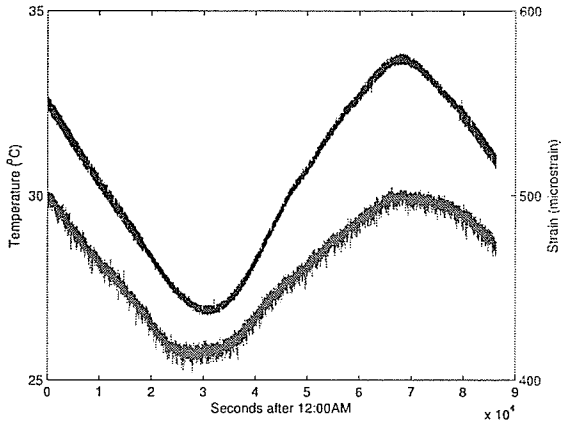


Figure 2.8: Strain and Temperature Comparison, July 4th, 2007.

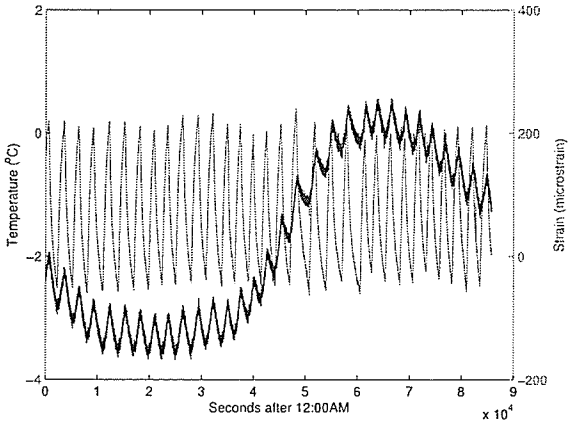


Figure 2.9: Strain and Temperature Comparison, January 4th, 2008.

Chapter 3

Data Collection And Preprocessing

For a measurement system to be useful, there needs to be a way to remove data from the system. This data must be removed from the system in a suitable form. For the purposes of structural health monitoring, data is generated in a stream. Therefore, the system must have a way to process and handle the continuous data without unnecessarily losing samples or destroying useful information.

When the physical data is first removed from the measurement system, it is not in an acceptable form to be used with any other portions of the project. There is the potential for unwanted effects to be present in the recorded signals. Two effects considered are environmental effects, related mainly to temperature variations, and a zero-offset error, related to the measurement system not being properly initialized at installation.

Both of these effects must be removed before the data can proceed to the event detection algorithm. However, removing them was a non-trivial task and many methods failed before a suitable method was found.

3.1 Data Collection

Special consideration is necessary when dealing with large amounts of data. Large amounts of data must be processed in pieces, because of physical limitations imposed on computing

systems such as available RAM and hard disk space. Since structural health monitoring generates huge amounts of data, and is continuously generating more data, there is no way to examine the entirety of the data in one block. As a result, some way to cascade sensor data must be implemented. For the purposes of this project, data was read directly from an online database. There was no interaction done with the measurement system, as an interface is already in place that takes the physical measurements and places them in the database.

3.1.1 Data Storage

The database stores its data on a sample-by-sample basis. Each sample has a corresponding timestamp with a resolution of one second, and a sample index which ranges from 1 to 100. As well, each sample has data stored channel-by-channel; for this project, the only data elements of interest were the girder strain measurements and the external thermocouple output. The data in each sample is composed of fifteen loaded strain channels and one unloaded strain channel, as well as the average of seven thermocouples.

3.1.2 Reading Data into the System

Since the data is located on a MySQL database server, it can be extracted in arbitrarily sized blocks. Each block is then processed in arbitrarily sized windows. For this project, the window size used is 15,000 samples, translating to 2.5 minutes of readings, assuming no samples are missing. This assumption is not always accurate, as there are many gaps in the recorded measurements. However, there is nothing that can be done about these gaps, since they result from issues with a communication link, with the corresponding data being irrecoverable. By using windows that are based on the amount of samples and not on timestamps, these gaps can be ignored for the purpose of inputting data into the system. While an event may become corrupted by these gaps, only one event can be corrupted per gap, and the gaps are not very common.

To ensure continuity of the data, the input windows are cascaded during processing. Once half the window, or 7,500 samples, has been examined, the first 7,500 samples are removed and the next 7,500 are then read. This requires only the current 15,000 samples to be stored in the working memory of the processing program. It should be noted that the 7,500 sample cutoff is a loose limit; the event detector may require reading more than 7,500 samples to

extract an entire event, this is allowed by the system and will be discussed further in Section 4.2. If the algorithm detects that the current window is the final window (that is, there is no more data to read), then the entire 15,000 samples are examined without cascading.

Blocks of data can be taken off the database server in almost any length. If the data is stored on a hard disk, more data can be taken in one piece than if the data was stored only in the RAM allocated to the software. A minimum length of 15,000 samples is required to satisfy the requirements of the window, and the maximum length depends on the storage of the machine. For perspective on the typical file size that can be expected, a file with two full days of data, or 17,280,000 data points x 17 data channels, has an average size of 3GB. Input file cascading can be done if required, however, if files are not cascaded, the loss of information at the boundary between files will be minimal and not a serious concern due to the sheer amount of information available to be processed. Depending on concerns about the speed of degradation of the structure, the block size can range from being long enough for bulk processing (for example, processing the data from the previous day at 12:00AM each day), or smaller for a more real time approach (for example, processing the data in 20 minute pieces).

3.2 Data Preprocessing

The main goal of processing the strain data is to isolate vehicle events, since these events contain the information required to draw conclusions about the state of the structure. Before the events can be extracted from the measurements, however, several problems exist with the raw data.

Two unwanted effects are present and easily visible when examining the raw strain data from the gauges on the bridge: environmental fluctuations and a zero-offset value. The main cause of the environmental fluctuations is temperature, and as a result, temperature effects and environmental effects are used interchangeably. Because of the two problematic effects, processing must be performed on the data before the event detection algorithm can be applied. This is considered the preprocessing step, since it is done to prepare the data so that it can be processed by the event detection algorithm.

For the purpose of this discussion, the signal is assumed to be noiseless. While in reality the signal is fairly noisy, the preprocessing does not try to remove the noise and, as a result,

providing consideration for the noise has no impact on the process. It is therefore left out to avoid unnecessary complications to formulas by introducing noise-based terms.

3.2.1 Temperature Effect

The raw data obtained directly from the measurement system may contain fluctuations due to temperature, even when considering the relatively small size of the data window compared to temperature cycles [6][7][8][9]. This temperature effect must be removed; ignoring it creates inconsistencies in measuring events and complicates any event detection algorithm that is implemented. The temperature effect can be modeled as an additive effect, added to the original signal such that Equation 3.1 represents the strain output with the temperature effect included.

$$S_o(t) = S_s(t) + f(T(t)) \tag{3.1}$$

Where $S_o(t)$ is the output strain from the system at time t, $S_s(t)$ is the strain from the structure at time t, $T(t)$ is the temperature at time t and $f(T)$ is an unknown transformative function that is applied to the temperature.

Some difficulties arise in trying to remove the temperature effect shown in Equation 3.1. First is that it cannot be assumed that the effect is *significant* in every data window. In other words, the portion of a temperature effect present may not necessarily impact the output of the system in any way. In fact, in most data windows, the temperature effect is not significant. Another issue is that, even if a temperature effect is found to be significant during a window, it may not be significant on all channels in that same window. For the effect to be considered present and significant, it is required that $f(T) \neq C$, C constant, for the duration of the window. To compound the difficulty in finding an algorithm to remove temperature fluctuations, $f(T)$ is not identical for all channels, leading to individual channels showing different variations for the same temperature. Since the effect may not be present in all channels, the algorithm should be designed to preserve channels that do not have a temperature fluctuation present, so that it can be applied indiscriminately to all channels for all windows.

3.2.2 Zero-Offset

The other unwanted effect is that the zero-offset value of the signal is not constant across all channels. For this project, the zero-offset is defined as the strain value found in the channel when there is no vehicle on the bridge. That is, it is the offset found when the strain value should be zero; in the ideal case, this would obviously be zero. Having an offset that is not zero causes problems similar to the temperature effect; it makes both event detection and event comparison difficult.

3.2.3 Combining Both Effects

Unfortunately, neither the zero-offset nor the temperature effect exists individually and both must be dealt with at the same time. An example of both of these effects can be seen in Figure 3.1. The strain curve shown in Figure 3.1 is from Channel 2; the strain value

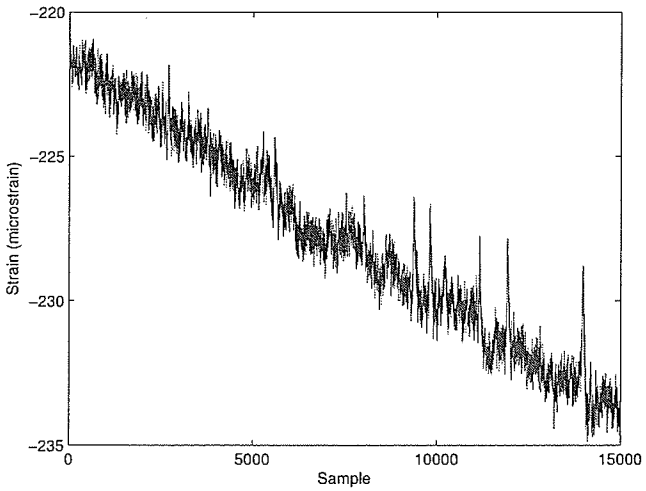


Figure 3.1: Strain Curve of a Single Window Showing Undesirable Effects.

here should be zero, with noise, except during events. This is clearly not the case; there is a noticable linear decrease, the zero-offset value is ambiguous but is likely ~ -222, and the events (larger peaks) seen from sample number 9,000 onwards are barely visible. From this example, it should be clear as to why preprocessing is required: the events are hidden in the trend, and no baseline strain value exists that allows accurate peak measurements.

As a general model, the strain output with both a temperature effect and non-zero offset is shown in Equation 3.2.

$$S_o(t) = S_s(t) + f(T(t)) + ZO \tag{3.2}$$

Where $S_o(t)$, $S_s(t)$, $T(t)$ and $f(T)$ are identical to Equation 3.1 and ZO is the zero-offset value. The parameters $S_s(t)$, $f(T)$ and ZO are unique to individual channels. Several strategies were attempted before a suitable method for correction was found.

3.2.4 Unsuccessful Strategies

Several unsuccessful strategies were employed at correcting both of these problems. The first attempt was to try temperature compensation using the unloaded dummy gauge [6][10]. The concept behind this idea was simple; if the dummy gauge has no structurally introduced strain, then any variations in its value should be caused by temperature effects. As well, since all sensors are in the same environment, it was assumed that all sensors would show the same effect from the temperature. Initial attempts involved simply subtracting the dummy channel from the other channels. This did not work, since it appeared that the temperature effect was time-shifted in different channels.

As a result, the next attempt was to find an approximation of $f(T)$ using the dummy channel and the temperature [6][11]. The concept was that once $f(T)$ was found, it could be used to subtract the effect from any channel. This also did not work, and is what led to the conclusions that the temperature effect is different for all channels and that it needed to be dealt with on a case by case basis.

The next method was an envelope detection algorithm typically used in amplitude modulation problems [12]. While this method showed initial success at removing the minor temperature fluctuations that were seen, it also had a tendency to destroy the event information. As well, in cases where envelope detection did work, it proved to be very slow, taking approximately 30 seconds per channel to clean a 2.5 minute window.

The most successful of the failed attempts was highpass frequency domain filtering [9]. This technique provided excellent results; it succeeded in removing both the low frequency temperature effects as well as removing the zero-offset value. The problem with filtering arose when trying to find a cut-off frequency that kept all events intact while removing the unwanted effects. This frequency seemed to be slightly different for each window that

was filtered. As well, filtering was slow, though not nearly as slow as envelope detection, because of the need to perform the Fourier transform, apply a filter and then perform the inverse Fourier transform.

3.2.5 Method Used

The selected method is a modified envelope detection algorithm. It is a two-step process that first removes events, leaving the background trending of data. It then removes the trend that is left over from the original signal, leaving the events. The concept for this method was found in *Using Morphological Filters to Extract Spiky Transients in EEG* [13]. This thesis presents a method which removes spiky transients from an EEG heart signal, leaving behind the background information which represents what is important in an EEG. For the purpose of this project, the algorithm is backwards; the spiky transients can be considered the events, and the background information is the undesirable effect. However, since the method presented in [13] results in the background information with the events removed, this result can simply be subtracted from the original signal to obtain only the events. The algorithm employs mathematical morphology to extract geometric properties from the signal. Two specific geometric shapes of interest exist in the signal. The events, which have a short profile consisting of a defined start, defined end and high amplitude peak, and the background fluctuations, which typically do not have an easily defined end, are small in amplitude and have long profiles.

3.2.6 Mathematical Morphology

Mathematical morphology is a technique that is commonly applied to image processing, and because of this, it was initially difficult to predict how the techniques would apply to a one-dimensional strain signal. In the most basic form, morphological operations are defined for binary images, are based on set theory, and consist of taking a geometric shape (the *structuring element*) and performing a set operation, such as the union, of the structuring element with the data in an attempt to extract geometric artifacts that are present [14]. By expanding how morphology works with binary images, it can also be applied to non-binary images, and as a result, non-binary signals as well.

For the purposes of the project, two morphological operations were chosen: opening and closing. From an image processing perspective, opening can be seen as a technique that

smoothes the contour by breaking small connections and erasing small spikes [14]. Opening can be used to identify geometric areas which can fit the entirety of the structuring element. In a complementary fashion, closing can be seen as a technique that eliminates gaps smaller than the structuring element [14]. While these concepts are more easily visualized with binary images, a description of how they apply to one-dimension signals is given later in this section. Opening and closing both rely on two fundamental morphological operations, dilation and erosion. In brief terms, dilation runs along the border of an image and increases it, while erosion runs along the border and reduces it [14]. Equations 3.3 to 3.6 show the general formulas for erosion, dilation, opening and closing, of two sets, A and B, in Euclidean space Z, expressed in terms of set operations [14]. In the equations, B_z indicates set B centered at point z.

$$\text{Erosion: } A \ominus B = \{z|(B_z \subseteq A)\} \tag{3.3}$$

$$\text{Dilation: } A \oplus B = \{z|(B_z \cap A \neq \emptyset)\} \tag{3.4}$$

$$\text{Opening: } A \circ B = (A \ominus B) \oplus B \tag{3.5}$$

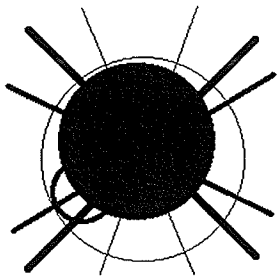
$$\text{Closing: } A \bullet B = (A \oplus B) \ominus B \tag{3.6}$$

Figure 3.2 shows an example of the dilation, erosion, opening and closing operations applied to an image, with a disk as the structuring element. These concepts are easily visualized when applied to binary two-dimensional images, but the concepts apply in a similar fashion to one-dimensional signals as well. Equations 3.7 and 3.8 show a specific form of erosion and dilation used for one-dimensional signals [13][15], in which f is the function to be eroded/dilated, with length N, and b is the structuring element, with length M.

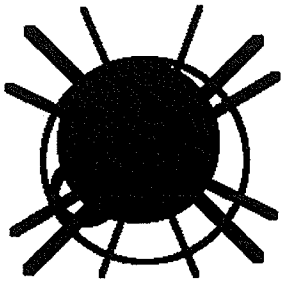
$$(f \ominus b)(x) = \min\left(f(x+n) - b(n)\right) \quad n = -\frac{M}{2}, \ldots \frac{M}{2}, 0 < x+n \leq N \tag{3.7}$$

$$(f \oplus b)(x) = \max\left(f(x+n) + b(n)\right) \quad n = -\frac{M}{2}, \ldots \frac{M}{2}, 0 < x+n \leq N \tag{3.8}$$

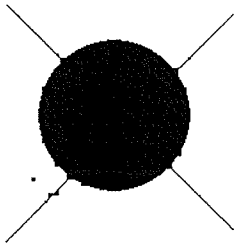
For cases where the structuring element is equal at all indices ($b(n) = \vec{C}$, C constant), the $b(n)$ term can be ignored from both Equations 3.7 and 3.8. For the purposes of this project, a constant vector is used as the structuring element.



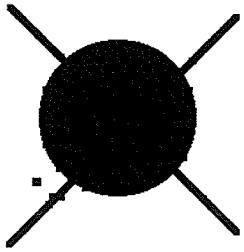
(a) Original Image.



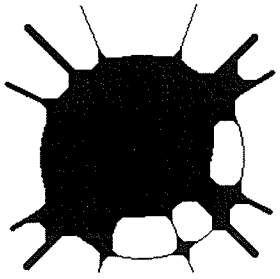
(b) Dilation, *Structuring Element: Disk* $r = 3$.



(c) Erosion, *Structuring Element: Disk* $r = 3$.



(d) Opening, *Structuring Element: Disk* $r = 3$.



(e) Closing, *Structuring Element: Disk* $r = 10$.

Figure 3.2: Examples of Morphological Operations.

By looking at Equations 3.7 and 3.8, the effects of dilation and erosion of a one-dimensional signal can be predicted. Using one of these techniques behaves like convolution. For each point in the signal, dilation will replace the point being examined with the maximum value found within the range of the structuring element, centered on the point being examined. The structuring element is then moved to the next point and truncated if it exceeds the boundaries of the signal. Erosion is similar, taking the minimum value instead of the maximum.

As an example, assuming a structuring element of size 3, and a signal consisting of [1 2 1 3 4 4 1] the result of dilation will be [max(1, 2) max(1, 2, 1) max(2, 1, 3) max(1, 3, 4) max(3, 4, 4) max(4, 4, 1) max (4, 1)] or [2 2 3 4 4 4 4]. This creates a very rough upper envelope, with plateaus potentially the length of the structuring element. Using erosion creates a lower envelope, with a similar process to dilation. The result of the erosion of the example signal would be the minimum of each data set, or [1 1 1 1 3 1 1]. As shown by these examples, dilation and erosion are useful for quickly extracting information about the upper and lower bounds of one-dimensional signals. When opening and closing combine erosion and dilation, a more favourable result is created than when just one of the two is used. Both opening and closing tighten the envelope created, allowing for better removal of the events and representation of the shape of the background curve.

3.2.7 Preprocessing Algorithm

The goal of the preprocessing algorithm is to remove the unwanted effects, and keep the events. By considering the results of the morphological operations from the example above, it is possible to predict how these operations will interact with a strain signal. Closing, by dilating then eroding, will provide a lower envelope of the upper envelope; opening similarly provides the upper envelope of the lower envelope. These envelopes will include events unless the structuring element is too wide to fit inside the event. Therefore, the structuring element should be wider than the widest event expected, or it will be removed along with the unwanted effect. To guarantee removal of unwanted effects, the structuring element must be thinner than the thinnest fluctuation in the effect. Despite these seemingly simple restrictions, meeting them may not be possible. Slow moving vehicles can generate large events, on the order of thousands of samples, requiring an extremely long structuring element. Conversely, the smaller environmental effects have a typical duration of 100 to 500 samples; using a structuring element larger than this will not remove any of them. A

compromise must then be made, with the emphasis on keeping the events. The majority of event spikes are at most 300 samples long, and so this value was chosen to be the size of the structuring element.

The actual preprocessing algorithm involves performing both opening and closing on the signal. The two operations are done in pairs, once with closing the original signal first, then opening the result, then by opening the original signal, following by closing the result. This technique of doing closing-opening and opening-closing was based on [13][16]. The two operations are shown in Equations 3.9 and 3.10

$$Co(t) = (S_o(t) \bullet b) \circ b \tag{3.9}$$

$$Oc(t) = (S_o(t) \circ b) \bullet b \tag{3.10}$$

Where $Co(t)$ is the closed-opened signal, $Oc(t)$ is the opened-closed signal, $S_o(t)$ is the output from the measurement system and b is the structuring element.

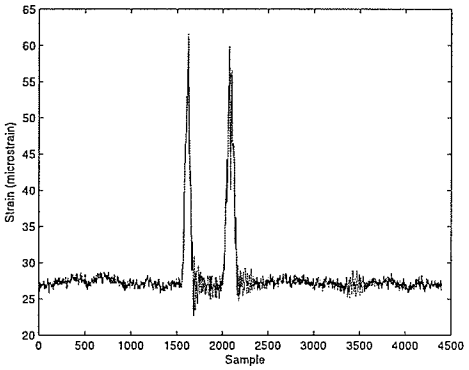
It is required to perform both operations: closing followed by opening. Simply performing closing will remove events by tracing the top of the event; it is only by performing opening on the result that the events can be skipped. To illustrate this point, Figure 3.3 shows the results of closing the original signal, then opening the closed signal. Figure 3.3c shows that closing-opening yields an approximation of the upper envelope for the base trend in the data.

Since there are no events on the bottom of the signal, performing closing following opening is not a neccessity. The closing pass is useful because of the presence of noise, and by closing the result of opening, it is possible to smooth out some of the variations that noise spikes place in the envelope. Performing opening-closing yields a lower envelope approximation.

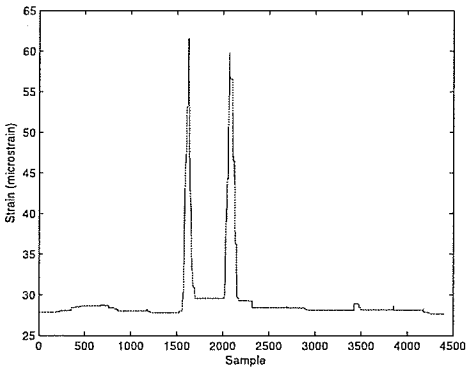
The process of closing-opening and opening-closing is performed on a channel by channel basis; once both are done, they are averaged point-by-point and the result is said to be the collection of unwanted effects. This is shown in Equation 3.11.

$$\frac{Co(t) + Oc(t)}{2} = f(T(t)) + ZO \tag{3.11}$$

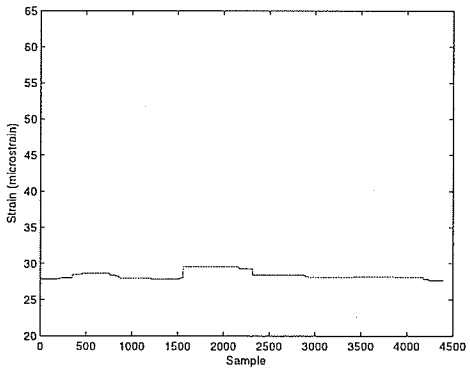
Where $Co(t)$ is the closed-opened signal, and $Oc(t)$ is the opened-closed signal.



(a) Original Signal.



(b) Closed.



(c) Closed Then Opened.

Figure 3.3: Example of Closing-Opening a Signal.

A *cleaned signal*, a signal that has the unwanted effects removed (but is still noisy), is therefore the original output signal minus this result, as shown in Equation 3.12.

$$S_c(t) = S_o(t) - \frac{Co(t) + Oc(t)}{2} \tag{3.12}$$

With $Sc(t)$ being the cleaned signal.

An additional step is performed to guarantee the removal of the zero-offset value, and that is to subtract the median of $S_c(t)$ from $S_c(t)$. Equation 3.13 therefore gives the final result of the preprocessing algorithm.

$$S_c(t) = S_c(t) - med(S_c(t)) \tag{3.13}$$

Figure 3.4 shows the data from Figure 3.1 after it has been cleaned by the preprocessing algorithm. Notice that the algorithm has succeeded in removing the linear effect and then resolving the zero-offset ambiguity and removing it. As well, it has enhanced the five events present near the end of the signal. In other words, the strain values for the start and end points of each event are almost identical, which allows for more useful interpretations to be made.

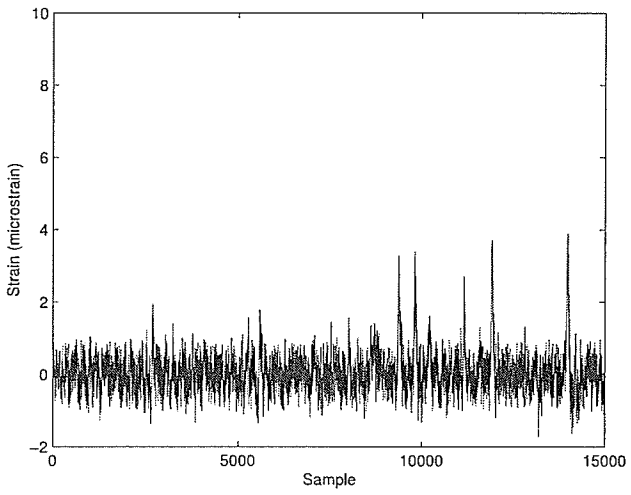


Figure 3.4: Cleaned Strain Channel.

The preprocessing algorithm is of fundamental importance to the project. Even though the temperature effects are rarely present, and may be missing for months at a time, when they are present it could prove disastrous for detection and classification. Through sparse observations, since it is infeasible to examine every segment of an entire year of data, the effects are rarely present in the summer months, and much more likely to be present in the winter months. This could be explained by the much larger and faster temperature fluctuations seen in the winter, as shown in Section 2.4.

Chapter 4

Event Detection

The events caused by vehicles traversing a bridge contain valuable information that can be used to determine if a bridge is working properly. Unfortunately, the events are entrenched in the data streams, essentially hidden in a vast sea of noise. Without some method for detecting when events are occurring, it is impossible to extract the necessary information to make automated decisions about the health of a bridge.

Event detection is a three-step process. When the algorithm is started, no events are known. In step one, on a channel-by-channel basis, the next event in a channel is found. In step two, the locations of the events in the individual channels are compared against each other to determine if the identified events are the same. In step three, a single window in time is selected to represent the event. Once all three steps are done, the process is repeated until all the data has been processed.

For the purposes of determining the health of a bridge, only one value is needed per event; this is the maximum, or peak, value of individual strain channels during the event. The location of the event peak is required for the event detector to operate; the starting location and the end of the event are required to extract peak values from channels that were missed by the detector. In all, this requires four measurements to be extracted for each event.

4.1 Preliminary Values and Algorithms to Aid Detection

Before the event detection algorithm can begin, three values must be determined and a separate pre-processing algorithm must prepare the data. The required values are two threshold values used to isolate events, as well as an estimate for the amount of noise in the system. The preprocessing algorithm that is applied is a smoothing algorithm, with the goal of making the signal more amenable to the requirements of automatic event detection and, eventually, classification.

4.1.1 Threshold Values

The first values to determine are two threshold values. The first of the thresholds, referred to as *Threshold A*, is used to determine that an event is occurring; that is, when a value is above Threshold A, it indicates that the value is part of an event, and not just part of the background noise. This does not mean that this point is the start of an event; it simply means that it is definitely in an event. The other threshold, *Threshold B*, is used to determine when the signal has decayed back into noise levels, which indicates the start and the end point of an event. Since these thresholds are used to separate events from noise, they must be able to do this independently of the noise level. Since the noise level is not constant throughout the course of a year, the thresholds must be determined dynamically, based on the noise level in the data currently being examined.

4.1.1.1 Estimating System Noise

To base the threshold values on noise levels in the system, a method must be developed that allows for estimation of the system noise into a quantifiable value. This noise level is obtained by examining the dummy signal taken from the preprocessing stage described in Section 3.2 and further processing it in order to remove large noise spikes. The reasoning behind the need to remove the large noise spikes and not simply using the cleaned dummy signal to represent the background noise is that, despite being theoretically unloaded, the dummy signal records some artifacts of the events on the bridge. The method used to remove noise spikes is a multi-step process.

The first step is to identify the samples in the original signal that are above the upper envelope and below the lower envelope. These envelopes are the same that were found

during the preprocessing stage by using the opening-closing and closing-opening techniques described in Section 3.2.7. Recall that the envelopes skip over the larger fluctuations in a signal; identifying the points above the envelopes therefore identifies the points that would be considered as large noise spikes.

The next step is to replace the identified points with the equivalently indexed sample from the proper envelopes. That is, a point above the upper envelope is replaced by its time-domain equivalent point from the upper envelope and similar for points below the lower envelope. Points within the envelopes are not changed. Following this step, it is a simple matter to subtract the resulting signal from the original signal to obtain a vector that contains only the noise spikes. After this, this new result is subtracted from the cleaned signal, that is, the result of the preprocessing step. Figure 4.1 shows an example of a cleaned signal compared to the signal used to represent the noise level.

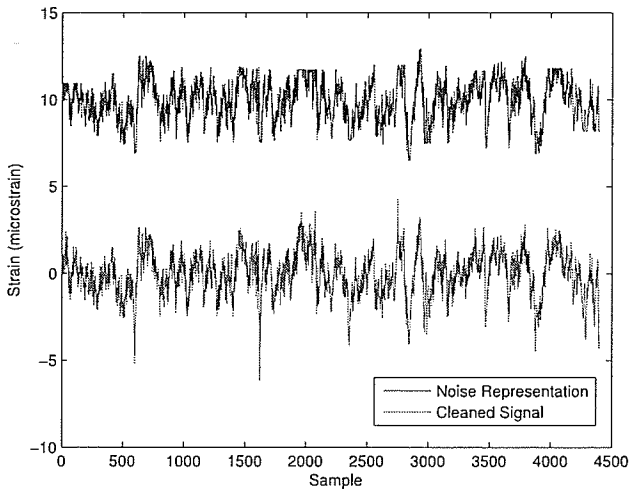


Figure 4.1: Comparison of Noise Estimate to Cleaned Signal.

The resulting vector is deemed to consist only of background noise. Based on this vector, the noise is then estimated to be zero mean, Gaussian noise with a standard deviation equal to the standard deviation of the vector. This standard deviation is representative of the noise level, and is referred to as σ_n.

4.1.1.2 Threshold A

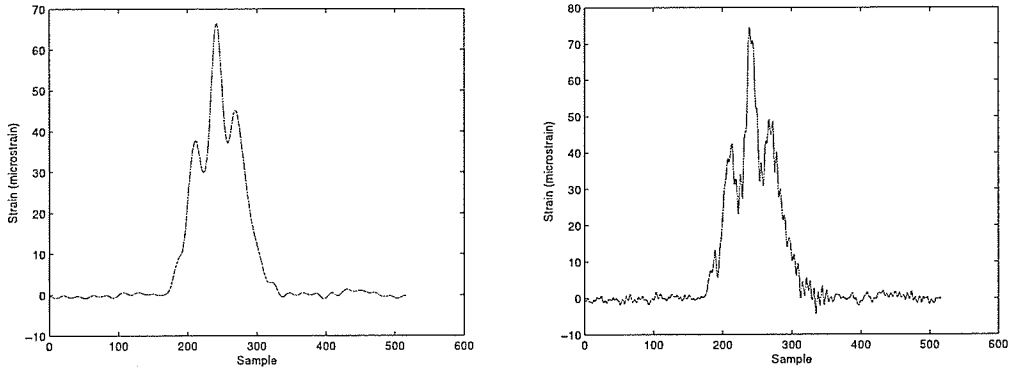
Once this approximation of the background noise is obtained, Threshold A can be determined based on σ_n. The value of Threshold A is $4\sigma_n$; the factor of four is used because, with zero-mean noise, the probability that a value is within four standard deviations is $P(|X| < 4\sigma_n) = 0.9999$. That is, 99.99% of values in a normal distribution fall within four standard deviations of the mean. Therefore, if a strain value is found to be outside of this range, it can be concluded with large certainty that it does not belong to the noise distribution, and is therefore part of an event.

4.1.1.3 Threshold B

Threshold B is determined in a similar method to that of Threshold A. Threshold B is equal to one quarter of Threshold A, or simply σ_n. This value was chosen to indicate that the measured strain value is in fact back in the noise distribution. The reasoning behind using a spread between thresholds is to add a hysteresis effect to the system, preventing the system from oscillating between the finding an event and claiming the event has ended. This would lead to an unreasonably large amount of false events, as well as compromising the integrity of the legitimate events, making classification virtually impossible.

4.1.2 Signal Preparation

Once both thresholds have been determined, the strain curves need to be further processed to increase the visibility of events to an automated detection algorithm. In order to do this, a Gaussian smoothing algorithm is applied to all channels [14]. Gaussian smoothing is an algorithm typically applied to blur images, but the concepts apply equally to a one-dimensional signal. The algorithm eliminates a significant amount of high frequency noise. Removing this noise helps in eliminating pathological cases in determining the start and end of the event. One such case is where a random noise spike exceeds Threshold A and then almost immediately drops below Threshold B. The other advantage of the smoothing algorithm is that it removes noise that corrupts peak values and leaves a smooth event. An example of a smoothed event versus an unsmoothed event is given in Figure 4.2.



(a) Smoothed Event.

(b) Unsmoothed Event.

Figure 4.2: Effect of Smoothing on an Event.

4.1.2.1 Gaussian Smoothing

A Gaussian smoothing filter is similar to a windowed averaging filter, except that it computes a weighted average for a data point based on the values of a Gaussian distribution (these values are referred to as the *coefficient vector*). A new signal is then constructed by piecing together the results of the weighted averages. A Gaussian smoothing window is centered on the data point being examined; this is done to give it the largest weighting. Using non-uniform coefficients instead of a standard uniform window average better preserves the shape of the original signal. The following approach for one-dimensional Gaussian smoothing is based on the smoothing filter discussed in [14].

Three parameters are required for Gaussian smoothing to generate an appropriate coefficient vector. These are the length of the window, M, the value for the standard deviation, σ, and how far into the tails of the distribution the end points will be, R. The mean should be set to 0 since the distribution is only being used to generate coefficients. Also, M should be made odd to balance the number of terms on each side of the centre point.

As an example, specifying $M = 5$, $R = 3$ and $\sigma = 2$ would make a window of size 5, with the coefficients for the weighted average equal to [0.0648, 0.1506, 0.1995, 0.1506, 0.0648]. These values are directly from the probability density function of the Gaussian distribution since, for example, $0.0648 = \left(2\sqrt{2\pi}\right)^{-1} e^{3^2/8}$. This allows for averaging to take into account

the values around the data point, but gives them less emphasis than the data point being examined. The values around the centre point are referred to as the *signal window.*

Since the sum of the coefficient vector is not equal to 1, a normalizing factor is introduced. This factor, which is the total of the coefficient vector, is divided from the weighted average; the reasoning for this is similar to why the calculation for the arithmetic mean is divided by the number of elements (windowed averaging, which uses the arithmetic mean, can be viewed as Gaussian smoothing with a coefficient vector of 1s). In the example above, this factor would be $2(0.0648 + 0.1506) + 0.1995 = 0.6303$. By dividing by this, it guarantees that the result of the smoothing filter will always be equal to or less than the maximum value in the signal window without grossly attenuating the signal.

The general equation for a Gaussian smoothing filter is specified in Equation 4.1. Two special cases are Equation 4.2, used when $n \leq \lfloor \frac{M}{2} \rfloor$ (the start of the filter is before the signal), and Equation 4.3, used when $N - n < \lfloor \frac{M}{2} \rfloor$ (the end of the filter is after the signal), where N is the length of the signal. These two equations are used so that zero padding is not required. For the equations, $f_o(x)$ is the output of the smoothing, while $f(x)$ is the input and $1 \leq n \leq N$.

Let $g[t]$ be the coefficient vector, with t valid from $-\lfloor \frac{M}{2} \rfloor$ to $\lfloor \frac{M}{2} \rfloor$, assuming M is odd.

$$f_o(x_n) = \sum_{i=-\lfloor \frac{M}{2} \rfloor}^{\lfloor \frac{M}{2} \rfloor} f(x_{n+i}) g[i] \tag{4.1}$$

$$f_o(x_n) = \sum_{i=1-n}^{\lfloor \frac{M}{2} \rfloor} f(x_{n+i}) g[i] \tag{4.2}$$

$$f_o(x_n) = \sum_{i=-\lfloor \frac{M}{2} \rfloor}^{N-n} f(x_{n+i}) g[i] \tag{4.3}$$

The smoothing process tends to reduce peak values; however, through observation it was found that this reduction is percentage based, with a similar factor applying to all events. As a result, the attenuation is not a significant issue and the smoothed signal is used for all subsequent aspects of the project: determining event components and event analysis.

4.2 Event Detection Algorithm

Once all channels have been smoothed, there are two ways to go about detecting events. All events can be found for one channel, then another channel, repeated sequentially throughout all channels. After this, the channels can be compared to determine which events are from the same global event, and a series of all events constructed. The other method is to find events one by one through all channels, determine which of the individual channel events are from the same global event, and then after an event is processed, determine the next one.

The second method was chosen to be implemented, since it allowed for easier testing and seemed more straightforward for termination conditions. In hindsight, both methods are equally suited to solving the detection problem and could be implemented with a similar amount of work.

Recall that event detection is a three step process. First, the next event in a channel is found. Next, it is determined which channel events are part of the same global event. Finally, a window is selected to represent the event.

For the algorithm, three states must be defined. The first is that an event has been found on a channel (*Event Found*), the second is that there are no more events left on the channel (*No Events*), and the final state is when an event has yet to be found, but it has not been proven that there are no more events (*Searching for Events*).

4.2.1 Finding the Next Event

This step is executed when a channel is in the state *Searching for Events*. Rather than monitoring all fifteen channels, the only channels that are monitored for events are Channels 1 through 5. Recall that these channels obtain measurements from each of the five girders near the abutment; if any events are going to occur on the bridge, they will need to occur here before they occur at the middle of the span. A channel is flagged as containing an event the first time a value that exceeds Threshold A is encountered. Once this happens, the channel is monitored for two values.

The maximum value is tracked so that the location and value of the peak can be stored. While this tracking is being done, the strain values are monitored to see if they drop below Threshold B, which would indicate the end of the event. Once the end of the event

is determined, it is possible to move backwards through the event, monitoring for a drop below Threshold B to determine the start. The start must be determined in reverse because Threshold B, which identifies the end points, is smaller than Threshold A, which identifies when to start looking at an event. It is determined last because it simplifies the calculations, allowing linear progression through an event until the end is detected. Once both end points have been identified, the channel is switched into the state *Event Found.*

It is possible that no data point is found that exceeds Threshold A. In this case, it can be concluded that there are no more detectable events left on the channel, and the channel is placed in the state *No Events.* If a channel is put in *No Events*, its data is set to nonsense and ignored for the remainder of the algorithm.

4.2.2 Determining Global Events

In order for the event detection algorithm to be useful for the purposes of this project, events found on different channels need to be labelled as part of the same global event, so that the characteristics between channels can be observed properly. Once all channels are in the states of either *Event Found* or *No Events*, and at least one channel is in *Event Found*, then a process begins to identify which channels have found the same global event. This is done by comparing the peak locations found by the different channels.

If any peaks are within a certain distance from each other in the time domain, then all peaks are said to belong to the same event. The distance used is 200 samples, centered on the first examined peak, or one second in both positive and negative directions. This was chosen because it is unlikely that two separate events will occur within a one second window and it is equally unlikely that the propagation of the load across the girders will take longer than one second. To make computations easier, all peaks are compared to the earliest peak found in the data (the *reference peak*).

For example, if Channel 1 has a peak at sample 1150, Channel 2 at 1152, Channel 3 at 1145, Channel 4 at 1562 and Channel 5 is in *No Events*, the reference peak will be 1145, from Channel 3. The algorithm will conclude that Channels 1 (reference+5), 2 (reference+7) and 3 (reference+0) contain the event, while in Channels 4 (reference+417) and 5 (*No Events*) the event is missing, likely masked by noise.

Using the mean of the peaks of any channels that match the reference peak was considered, but the advantages were not there to justify changing the algorithm. The starting and

ending points from each of the channels determined to be part of the event are stored, and then the channels are put into *Searching for Events*. The process is repeated until all channels are in the *No Events* state.

4.2.3 Determining a Bounding Window for an Event

Once an event has been detected, a suitable window needs to be selected to represent the event in its entirety. This window is selected from the starting and ending points of each channel found to contain the event, and in order to accommodate the entire event, the largest window, within reason, is chosen. This means that a small test is put in place to attempt to remove outliers and nonsensical values. This happens when one channel is exceedingly noisy when compared to the others; its noise level is much higher than the predicted value and as a result, the event may appear to start early and end late in this channel. These values, however, should not impact how the window of the event is determined.

Because the data set is small, containing anywhere from one to five elements, using traditional measures such as the deviation from the mean will not provide a robust identification of an outlier as both the mean and standard deviation are easily skewed by an outlier in small data sets. The measure chosen instead is the *Median Absolute Deviation* (*MAD*) [17]. The formula for the *MAD* is given in Equation 4.4.

$$MAD = \text{median}(|x - x_m|) \tag{4.4}$$

Where x_m is the median of the data set x.

Any value that is found to be greater than 3.5 times the *MAD* is removed from consideration. After this check is performed on both the set of starting values and the set of ending values, the event window is constructed by taking the minimum of the remaining starting values and the maximum of the remaining ending values.

When the system was implemented, the choice was made to calculate all windows after all events were found. This choice was made to facilitate debugging, to allow all the windows to be calculated at once, so that they could more easily be plotted and compared to the original signal to determine the accuracy of the detector.

Once this process is complete, it can be concluded that all the events are found and classification can begin on the events.

4.2.4 Cascading the Detector

Recall from Section 3.1 that the data collection needs to be cascaded so that the entirety of the data can be read. This poses difficulties to the event detection algorithm, as the preliminary algorithm relies on having the entire signal present. Some minor changes to the algorithm are required to accomodate this. The noise estimate and thresholds must be recomputed for each window. The condition for the transition to *No Events* changes from no event being detected in the channel, to no more events being detected before the midpoint of the window. While this midpoint seems like the new end of the channel, an event already detected is allowed to cross the midpoint boundary to keep continuity.

When all channels are placed in *No Events*, the detector is cascaded by moving back the tracking value the algorithm uses, the end of the last event found in a given channel. This is done to ensure that the same event is not detected twice when it occurs at the edge of a window. The value is lowered by 7,500 each time the reset happens, with a floor value of 1, and all channels are placed back into *Searching For Events*. If the data collection system determines that the end of the data set has been reached, the algorithm behaves as it would without cascading. Figure 4.3 shows a flowchart of the algorithm.

4.3 Determining Effectiveness of the Detector

In order to test how well the detector isolates events, data from *Model-Free Bridge-Based Vehicle Classification* [9] was used. The author, Grant Rutherford, manually counted vehicles on August 22nd, 2007. The results of this count were that 582 large vehicles traversed the bridge. In comparison, the event detection algorithm described above determined 1,576 events for the same time period.

While it is more promising that more events were detected, rather than less being detected, it still causes concern. One possible explanation for this large discrepancy is the type of events detected. While only large vehicles were counted during the experiment, the event detector may be capable of detecting smaller events. Testing this theory, the event detector determined there was 582 events above a threshold of 15.2 microstrain.

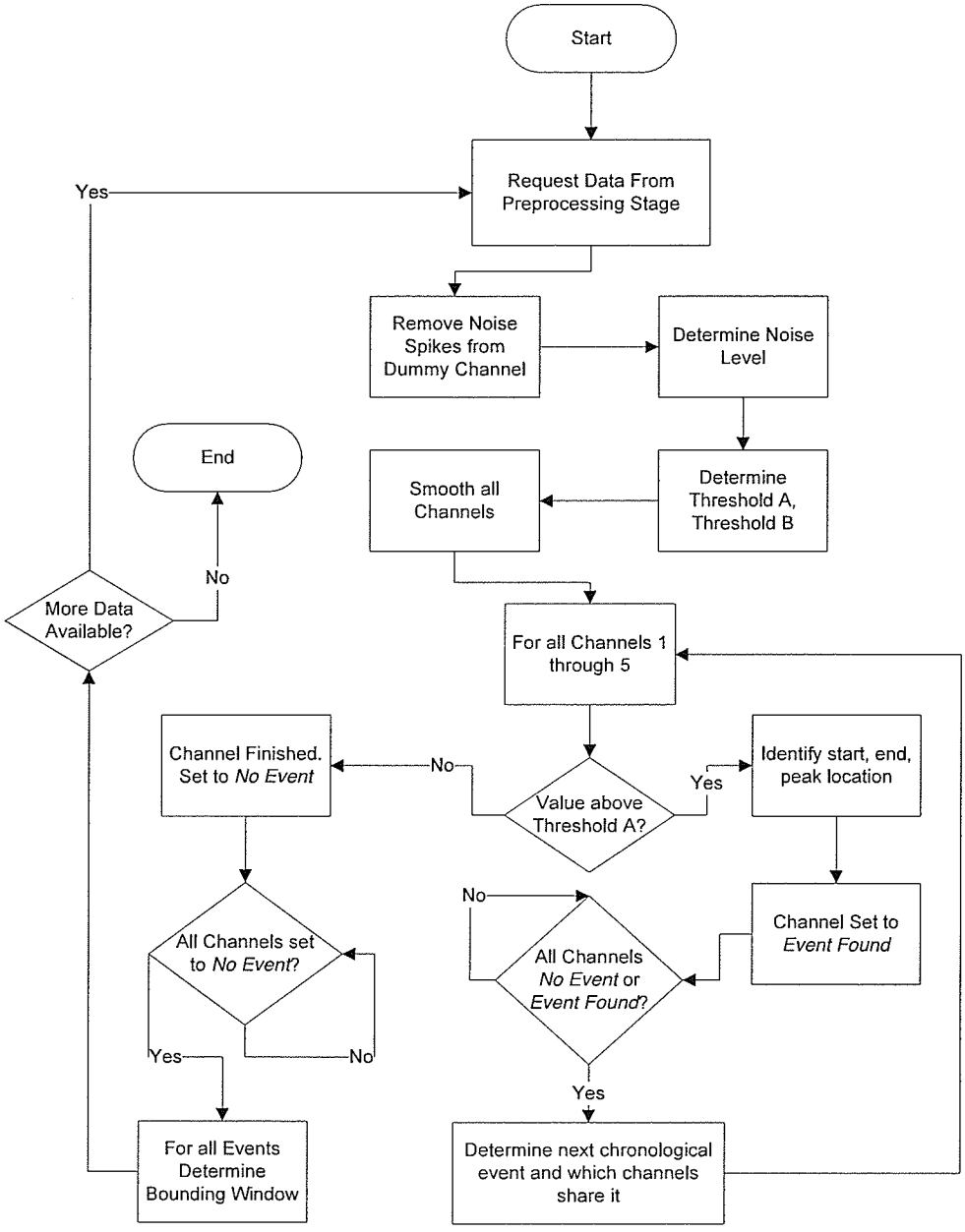


Figure 4.3: Flowchart of the Event Detection Algorithm.

Chapter 5

Event Classification

Event classification proved to be a difficult problem to generalize for every potential load case that can appear in the sensor measurements. It is based on anomaly detection; that is, events are classified as either an expected or an unexpected event. Because no general form was found, the classification problem has been broken up into several sub-problems. Each sub-problem represents a different classification case; these are split up by traffic lane and by weight class. In order to understand and test the classification algorithm, a simulation needed to be used to generate controllable failure conditions. Data produced by this simulation led to the development of a theory for the classifier; this theory can eventually be applied to a physical system.

5.1 Usable Data for Classification

Recall from Section 2.2 that the measurement system for the Red River-North Perimeter Bridge consists of fifteen active, loaded strain gauges for monitoring. However, not all of the data channels are usable for analyzing the behaviour of the bridge. There are three sensors per girder: one at the abutment and two at the midspan, where one is at the top of the girder and another at the bottom. Of these sensors, the strain readings from sensors at the top of the girders have small event amplitudes and, as a result, events are indecipherable from background noise. Figure 5.1 shows two vehicle events as recorded by both the bottom midspan sensor and the top midspan sensor. Examining this plot clearly shows how the top midspan channel contains little useable information. This eliminates five

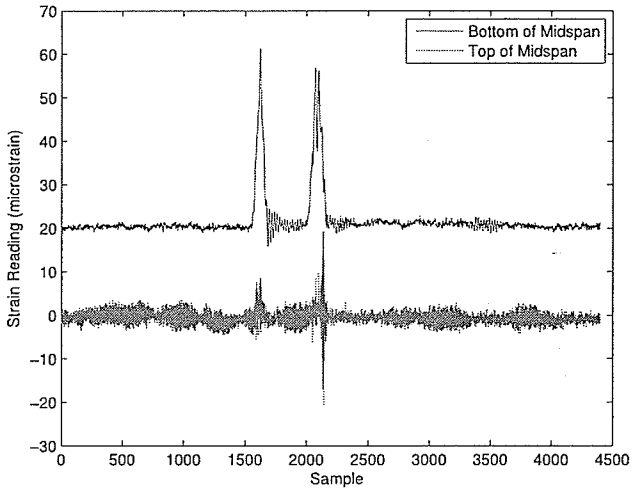


Figure 5.1: Usefulness of Bottom Midspan Sensor vs Top Midspan Sensor.
(Offsets Modified for Visibility)

sensors from the system, leaving only the abutment sensors and the sensors at the bottom of the midspan (future references to *midspan sensors* mean the sensors at the bottom of the midspan).

In addition to removing an entire group of sensors from consideration because their signal is too weak to be useful, the same problem applies to an entire girder, Girder 5. Girder 5 is the girder that is at the edge of the merging lane; it gets very little traffic and carries very little load. As a result, measurements from this girder contain no useful information about how the bridge is behaving. It is also unlikely to fail due to the much smaller loads it carries when compared to the other girders. Because of these characteristics, measurements from Girder 5 are ignored for the purposes of event classification. This leaves the system with eight possible sensor locations from which to measure. These are the sensors on Girders 1 through 4, located at the abutment and at the bottom of the midspan.

Of the remaining eight sensors, either set (abutment or midspan) can be used for classification, but different sets should not be mixed due to the fact that the strain signature is slightly different at the midspan and the abutment and the two cannot be compared directly. The midspan sensors typically record higher strain values, and if sensor groups were to be mixed, a scaling factor can be employed to try and fix this problem. Figure 5.2

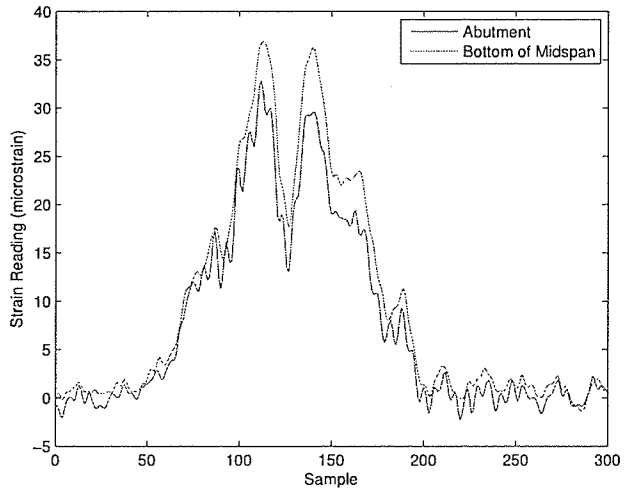


Figure 5.2: Amplitude Difference between Midspan and Abutment Sensors.

shows a close-up view of the second event in Figure 5.1, for the purpose of comparing the readings of an abutment sensor and a bottom midspan sensor. As indicated by the overlaying curves in Figure 5.2, the event amplitudes are different for the two sensor locations. As a result of this amplitude discrepancy, for this project, event classification is done using only the abutment channels. Despite this choice, the process outlined in this chapter will apply to midspan sensors as well, and can be employed to monitor both groups of sensors simultaneously if additional coverage is deemed necessary.

5.2 Event Features

The initial data extracted to describe an event is the peak strain value from the different channels being monitored. The four channels monitored for this project are from sensors located on Girders 1 through 4, at the abutment (G1, G2, G3 and G4 from Figure 2.3) and are referred to respectively as Channels 1 through 4. Therefore, each event is initially described by a 4 element vector, $V_m = [\max(C_1)\max(C_2)\max(C_3)\max(C_4)]$, consisting of the peaks of each of the four channels. Unfortunately, issues arise when trying to use V_m for classification, due to the large variation in peak strains recorded for different vehicles; these are discussed in Section 5.2.2. These issues cause V_m to not be very useful for extracting

quality information about the response of the bridge to the event. To solve this problem, some transformations are performed on the four quantities to establish new features: the source lane of the event, the girder ratios, and the weight class of the event.

5.2.1 Source Lane

One feature that can be derived from V_m is the source lane of the event. Determining the source lane is fairly simple; it is found based on the channel with the largest strain peak and is identically the index of the maximum value in V_m. If an event originates from the Normal Lane, then Girder 3, the *Main Girder* for the event, will show the most strain. Similarly, if an event originates in the Passing Lane, then Girder 1 will be the main girder. One more case is defined, Unknown Lane, when the peak strain channel is either Channel 2 or Channel 4. The source channel is an important feature since the distribution of load is different based on which channel is the source channel. Figures 5.3 and 5.4 show curves generated by the sensing system, with Figure 5.3 indicating the vehicle crossing in the Passing Lane and Figure 5.4 showing the same vehicle crossing in the Normal Lane. By examining Figure 5.3 it can be seen that the strongest response is in Channel 1, followed by Channel 2, then 3 and finally a weak response in Channel 4. In contrast to this, Figure 5.4

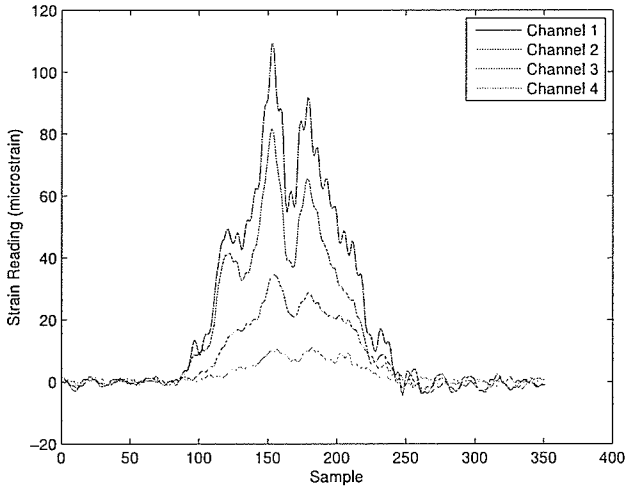


Figure 5.3: Sensor Curves of an Event in the Passing Lane.

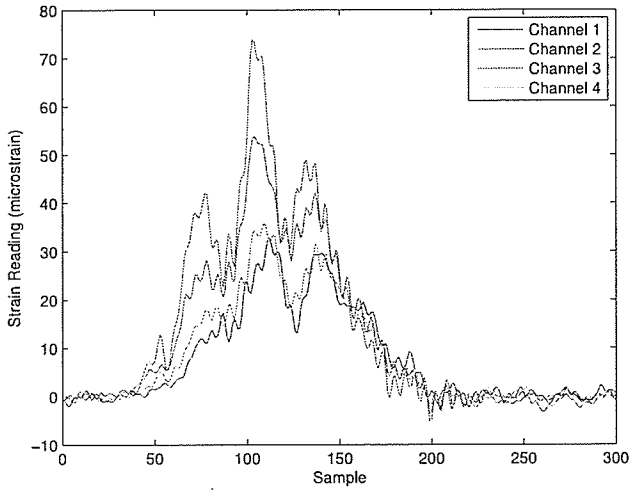


Figure 5.4: Sensor Curves of an Event in the Normal Lane.

shows a different response, with the strongest channel being Channel 3, followed by Channel 2, then 4 and 1. There is no excessively weak signal in this case.

Events labelled as originating in an Unknown Lane are thrown out, since they are unpredictable in their response, unlike Normal Lane or Passing Lane events, and are a small fraction of the events recorded. Equation 5.1 provides a formal definition of the source lane.

$$\text{Source Lane} = \begin{cases} \text{Passing Lane} & \max_i(V_m) = 1 \\ \text{Normal Lane} & \max_i(V_m) = 3 \\ \text{Unknown Lane} & \max_i(V_m) = 2 \text{ or } 4 \end{cases} \tag{5.1}$$

Where $\max_i(V_m)$ indicates the index of the maximum value of V_m, starting from 1.

5.2.2 Girder Ratios

The peak values located in V_m, while quite telling about how a bridge is operating, do not lead to the formulation of a general model that can apply to an unknown case. This is because the values in V_m have a large potential range based on the type and size of the

vehicle that is causing an event. This means that a general model would need to incorporate this large range of acceptable values, and that anomaly detection would in reality only be possible at the two extremes, when values are higher than expected for the largest event, or lower than expected for the smallest event. For an idea of how large this range is, data from the Red River-North Perimeter Bridge for August 22nd was passed through the Event Detection algorithm. For events deemed to be originating in the Normal Lane, the minimum peak in Channel 3 was 2.4 microstrain, while the maximum was 49.5 microstrain. For the Passing Lane, Channel 1 had a minimum peak of 2.4 microstrain, while the maximum was 71.6 microstrain. If a vehicle has an expected response of 20 microstrain, and failure in the bridge causes a reading of 30 microstrain, there would be no way to determine that the event should have given a reading of 20 microstrain, and thus no way to qualify it as a failure case.

The next set of features that are derived from V_m is the set of girder strain ratios. The ratios are more useful than the raw strain values found in V_m because they behave consistently no matter what the actual values in V_m are. This solves the problem described above where peak amplitudes can fluctuate over a wide range; the ratios between the girders can be shown to be constant over all amplitude ranges, provided noise is absent. This was demonstrated through simulation, and is discussed in Section 5.4.5. Using these relative measurements to compare how the strain is split between the different girders in the bridge is a meaningful description of the operating conditions of the bridge; this description of how strain is distributed across the bridge is referred to in this document as *Load Sharing*. Recall Figures 5.3 and 5.4; it can be seen that the value of these ratios will depend on the source girder. Since Girder 1 is the main girder when an event is from the Passing Lane, the ratios taken for a vehicle in the Passing Lane should all be taken with respect to Girder 1. When an event is from the Normal Lane, the ratios should be taken with respect to Girder 3. Additional ratios can be taken provided they are taken in a decreasing direction of amplitude; that is, all ratios taken should be greater than 1. The ratios that were selected to be examined for this project are summarized in Table 5.1.

Table 5.1 shows four ratios that can be used for each lane case. Each one contains an additional entry that does not depend on the main girder. This additional ratio can provide further information to supplement the ratios dependant on the main girder, to hopefully provide an expanded view on the state of the structure and the sensing system.

Table 5.1: Girder Ratios to be Used for Classification.

Passing Lane *Main Girder: Girder 1*	Normal Lane *Main Girder: Girder 3*
$\frac{G1}{G2}$	$\frac{G3}{G1}$
$\frac{G1}{G3}$	$\frac{G3}{G2}$
$\frac{G1}{G4}$	$\frac{G3}{G4}$
$\frac{G2}{G3}$	$\frac{G2}{G1}$

5.2.3 Vehicle Weight Classes

In addition to identifying the source lane and girder ratios, each event is assigned into a different weight class. The need for different weight classes is due to the presence of (additive) noise corrupting a measured strain channel. Recall that the ratios defined in the previous section are independent of the absolute amplitudes of the different peaks, but only in the absence of noise. Additive noise introduces into these ratios a dependency on the absolute amplitudes. This dependency is introduced because adding a value on both parts of a ratio has a different influence on the result of the ratio depending on what the initial values are. This is explained in Equation 5.2.

$$\frac{A+N}{B+N} \text{ approaches the ideal } \frac{A}{B} \text{ as } A \text{ and } B \text{ get larger than N.} \tag{5.2}$$

Equation 5.2 also applies in the inverse situation. If A and B get smaller, the ratio drifts away from the ideal value of $\frac{A}{B}$. If N is the noise level, Equation 5.2 means that the average ratio will drift depending on how A and B relate to the level of noise.

Due to the noise level being directly responsible for the magnitude of the disparity between the base ratio and the calculated ratio, the different weight classes are defined based on the magnitude of the peak in the main girder with respect to the noise level, N. For the actual system, three weight classes are used; the boundaries of these classes are based on the amount of noise. The boundaries are also dependent on the frequency of different

event amplitudes; this helps in preventing all events from being put in the same weight class.

5.3 Classification Implementation

Event classification is performed here using a single distribution. Events are classified as either being a member of the distribution or not, and this is done by comparing events against a decision boundary. This boundary is a threshold value, which divides the problem into members and non-members; that is, the probability of an event being a member of the class would be compared to this threshold and either included or excluded.

A single decision boundary that is usable in all cases is impractical to consider; it cannot be expected that the bridge will respond identically to all potential stimuli. This is where the different features described in Section 5.2 factor in. The problem is split into many sub-problems, with each sub-problem performing its own classification of events. These sub-problems are defined by three parameters: the vehicle weight class, the source lane, and the ratio used. Events can then be classified based on the probability that a calculated ratio is above the chosen threshold. Classification dealing only with individual events poses a problem, however. Due to the presence of noise and other potential measurement errors, simply detecting a single value outside of the selected range and using it to conclude that the bridge has failed is not a robust system and will generate far too many false positives to be useful. As a result, adding memory to the system will allow it to detect and monitor trends and consistencies in the data. This leads to a concept of *binning*.

5.3.1 Grouping Events

In order to solve the memory problem, each sub-problem is given a bin, and an associated event error counter. The reasoning for using an error count is similar to why the CAN bus protocol [18] uses an error count. That is, simply because an anomaly is seen does not necessarily mean something has failed; it is only after a number of anomalies are detected in a given interval that a conclusion can accurately be made about the state of failure. This introduces the idea for an error count. In the CAN protocol [18] a device can be in three states based on its error count, where each determines a different level of failure. A similar case can be made for classifying events. An isolated anomaly will likely be a sensor malfunction, a large noise spike, or some other one-time problem with the measurement

system. These isolated anomalies should be ignored, since they do not indicate anything about the state of the structure. Similarly, assuming a structure is failing, an isolated *proper event* (an event that fits within the expected error variation) should not remove an alert that the structure is failing. An error count allows tracking of the number of anomalies compared to the number of proper events, provided the count is increased when an anomaly is detected and decreased if an event is proper.

In the general case, the error count does not need to change in the same way in every situation. For example, anomalies could increase the error count proportionally to how far they are from the threshold. Despite providing additional freedom, in reality this is not especially useful. All anomalies should be considered equally serious, and should be valued equally. The choice was made for each anomaly to increase the error count by 1, with no ceiling on the maximum count, while each proper event decremented the error count by 1, with a floor value of zero. If these conditions are in place, then an error count which is steadily increasing will indicate a failure, while an error count that remains around zero will indicate that no failure has happened.

Because the Red River-North Perimeter Bridge has yet to fail, there is no real data available to indicate a failed structure. This led to the need to perform simulations in order to understand what happens to strain values with different amounts of noise, different vehicle weighting, and different failure cases. Knowledge of the simulation is required to discuss specifics on how the classifier is implemented and how decision making is performed. As a result, these specifics will be discussed in both Sections 5.4.9 and 5.5.

5.4 Classification Simulation

For the purposes of modelling failure, simulations needed to be run to extract information about how the bridge behaves under different conditions. These simulations were run using a program supplied by Dr. A. Mufti, SECAN4 (referred to simply as SECAN) [19]. This program allows for complete parameterization of a bridge structure as well as a load vehicle. Using this information, it performs a static loading calculation based on load position. The simulation allows the load to be positioned at any point across the surface of the bridge. Combining the results of several simulations allows for the generation of a curve similar to what is captutred by the physical system when a vehicle traverses the bridge. The material constants required for the simulation were provided by Dr. D. Sidhu [20] and are included

in Appendix A. To ensure the accuracy of the simulation, several important elements first needed to be examined.

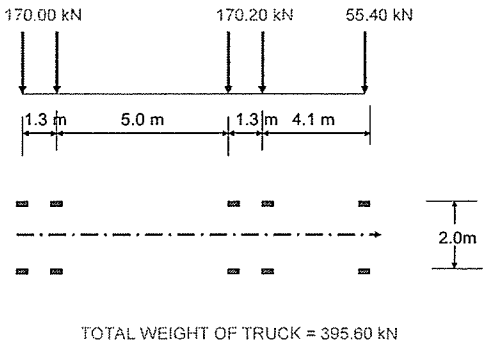
5.4.1 Vehicles

For the simulation, two different trucks were simulated to represent real vehicles. These trucks, named Truck A and Truck B, are representative of trucks loaded to the maximum legal axle weight for the Province of Manitoba; the difference between the two is the number of axles on the trucks (62,500kg [9]). See Figure 5.5 for images of the two trucks that were the basis for the simulation. Figure 5.6 shows these two trucks converted into a sequence of point loads [21]. This conversion is necessary so that the trucks could be used by the SECAN simulation.

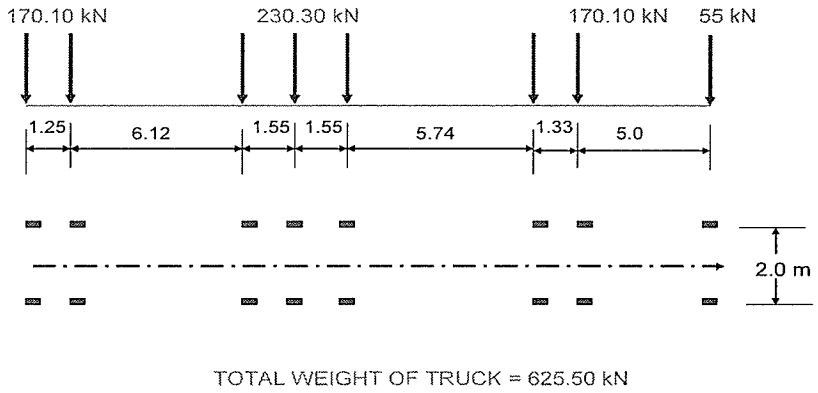
(a) Truck A.

(b) Truck B.

Figure 5.5: Images of Real Trucks used in SECAN Simulation.
(Image Credit: Dean K. McNeill)



(a) Truck A.



(b) Truck B.

Figure 5.6: Trucks as Point Loads for use in SECAN Simulation [21].

In order to simulate the physical system as closely as possible, the trucks were moved in a fashion similar to how they would be expected to traverse the bridge. The actual sensing system samples the sensors at 100Hz; for a vehicle moving at 100km/h, which is standard highway speed, Equation 5.3 shows the distance travelled per sample.

$$velocity = 100km/h * \frac{1000m/km}{3600s/h} = 27m/s = \frac{0.27m}{0.01s} = \frac{27cm}{sample} \tag{5.3}$$

This result shows that a truck travelling at 100km/h will move 27cm between each sample.

5.4.2 Lanes

The source lane of the vehicle is controlled by a parameter in SECAN. In order to properly simulate all cases of the bridge, both lanes needed to be simulated. The lanes are defined as horizontal position from the left-most girder. For this bridge, the left-most girder is in fact Girder 5; the girder under the merging lane. Figure 5.7 shows the locations of the lanes, from Figure 2.2, re-labelled for how the lanes were defined for SECAN. The vehicles were centered in the lane, with axles placed 1m on each side, at 4.5m and 6.5m (Normal Lane) and 8.2m and 10.2m (Passing Lane).

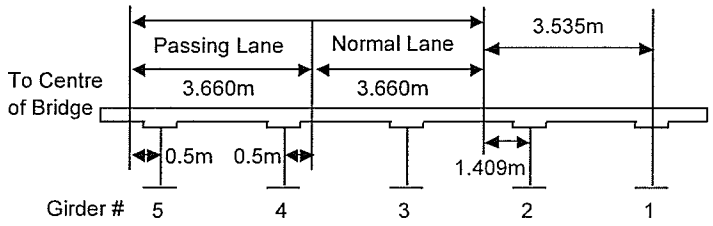


Figure 5.7: Lane Positions and Girder Numbering, Redefined for SECAN [1].
(Not to Scale)

5.4.3 Sensor Locations

Different measurement locations can be defined in the input for SECAN, which allows for multiple sensing locations to be defined. The locations of the sensors are defined only as a distance from the abutment; no positioning on the girder is able to be defined. For

robustness of the simulation, measurements are taken from both near the abutment and at the midspan, although the values from the midspan are not used in any way for this project.

5.4.4 SECAN Output

The output provided by SECAN, unlike the measurement system, is not strain values. SECAN outputs the moment about the geometric centroid. This value can be converted directly into a strain value by using Equation 5.4 [19].

$$\text{strain} = \frac{My}{IE} \tag{5.4}$$

Where M is the moment described above, y is the distance of the sensor to the centroid, I is the moment of inertia about the neutral axis and E is the modulus of elasticity of the girder material.

This moment is output for each girder in the simulation at each measurement location; in this case, two values for each of girders 1 through 5. The output is given in reverse order; that is, Girder 5 is labelled as Girder 1, Girder 2 as Girder 4, and vice-versa. While Equation 5.4 is convenient for converting the output of SECAN into strain to match it up with the output of the real system, in reality it is unnecessary. Calculating y accurately is a difficult endeavour, since the physical dimensions of the girders are hard to determine based on available documentation. However, it can be assumed that y will be identical for all girders, and will cancel when the ratios are taken; the same can be assumed for E. This leads to Equation 5.5, a more simplified form of the strain equation given earlier. The concept is labelled as the *M/I ratio*, since it is not a strain value, but can be used as one for the purposes of classifying events based on girder ratios. The M/I ratio will be treated as unit-less when used in discussions to match the unit-less strain measurements from the physical system.

$$\text{M/I Ratio} = \frac{M}{I} \tag{5.5}$$

5.4.5 Weight Simulation

SECAN was used to simulate the concept of different vehicle weight classes. This was done by providing a scalar multiplier on all components of the vehicle weights. The multipliers used were 1.0, 0.75, 0.50 and 0.25. The purpose of the weight simulation was to determine how the moment varies with respect to weight. Figure 5.8 shows the values of the M/I ratio for Sensor 1 for a vehicle in the passing lane in a case with no noise, for all four of the different weight levels.

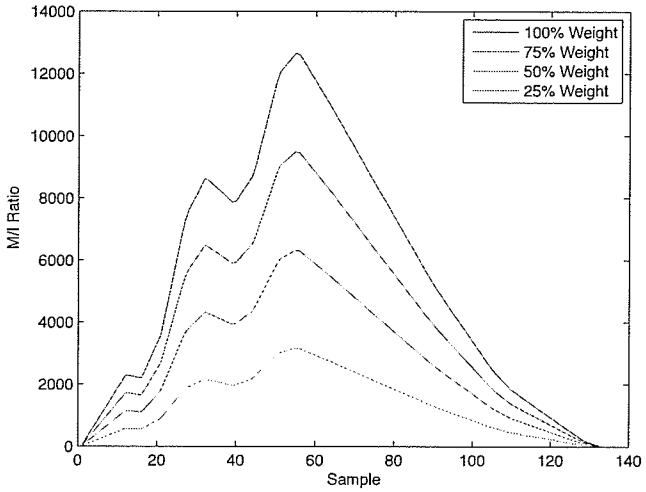


Figure 5.8: Weight Comparison for Truck A, Passing Lane.

Investigating these curves shows that the shape remains the same, independent of the weight class. This leads to the conclusion that the same type of vehicle will generate the same event curve independent of the weight, simply scaled by some factor. Numerically comparing the curves in Figure 5.8 shows that this is indeed the case, and that the scale factor is identically the multiplier used on the weight. It follows directly from this result that girder ratios will be identical for different weights of the same type of vehicle, since the multiplying constants will apply equally to the terms on both the top and bottom of the ratio.

5.4.6 Determining Noise Level

Once the idealized output was generated, it became apparent that to properly simulate a real system, noise would need to be added. Recall that the noise model used is that of zero-mean Gaussian noise. The variance of the noise was determined directly from data taken from the Red River-North Perimeter Bridge. On July 12th, 2007, ten tests were run on the Red River-North Perimeter Bridge. These tests were controlled for different trucks, speeds and source lanes and are summarized in Table 5.2. In Tests 1 through 8, Truck A traversed the bridge first, followed by Truck B. In Test 9, only Truck B traversed the bridge, and in Test 10, the two trucks travelled side-by-side.

These tests were convenient for the purposes of noise simulation since they provided a direct comparison between the results of the physical system and the results of SECAN. Since the two trucks used in the controlled tests are the same as the two trucks shown in Figures 5.5 and 5.6, the controlled tests can be used to determine a proportional constant that relates the noise level to the amplitude of the event response.

Table 5.2: Controlled Tests.

Test	Speed	Lane
1	25km/h	Normal
2	50km/h	Normal
3	75km/h	Normal
4	100km/h	Normal
5	100km/h	Passing
6	100km/h	Normal
7	100km/h	Passing
8	100km/h	Passing
9	100km/h	Passing
10	100km/h	Both

To determine this proportional constant, the event detection algorithm was run on Controlled Test 4. Recalling Section 4.1.2, one result of this detection is the standard deviation of the noise. This result was compared to the unsmoothed peak value for Channel 3 of the event and used to determine a ratio for the amount of noise in the measurement system that can be applied directly to the SECAN results. The result of this test was that the standard deviation of the noise was 1.17 microstrain while the peak had a value of 50 microstrain. This leads to the ratio expressed in Equation 5.6, with σ_n being the standard deviation of the noise.

$$\frac{\sigma_n}{\max(C_3)} = \frac{1.17}{50} = \frac{1}{42.73} \tag{5.6}$$

Or, expressed in a more useful form,

$$\sigma_n = \frac{\max(C_3)}{42.73} \tag{5.7}$$

The peak value of the M/I ratio provided by SECAN for a similar test, that is, Truck A in the Normal Lane, was 9264. By using Equation 5.7, this leads to a noise standard deviation of 216.80. This value is used for all simulations that include noise, independent of factors such as the size and weight of the vehicle, the source lane, or any smoothing performed on SECAN curves.

5.4.7 Failure Model

The model for structure failure used in this project is a decrease in the moment of inertia of a specific girder [22]. Modifying the moment of inertia of a girder has two effects on the M/I ratio for that girder. The first is that the calculated moment, M, decreases because the girder will take less load. The second is, of course, that I itself decreases. This leads to a potentially ambiguous state, where a decrease in I may result in either an increase in the M/I ratio, or a decrease in the M/I ratio, depending on which quantity has more of a change. In a simpler case, for the girders that are not failing, M will increase due to these girders needing to support a larger portion of the vehicle load. This increase in M, coupled with the knowledge that I is unchanged, leads to an obvious increase in the M/I ratio and consequently the strain.

In order to determine how the M/I ratio for the failing girder changes as I is changed, SECAN was run with different I values. Figures 5.9 and 5.10 show two different cases: when the failing girder is the main girder and when it is not. Both cases are with Girder 3 failing, and show curves of Girder 3; Figure 5.9 with the vehicle in the Normal Lane and Figure 5.10 with the vehicle in the Passing Lane. Examining both these figures shows a

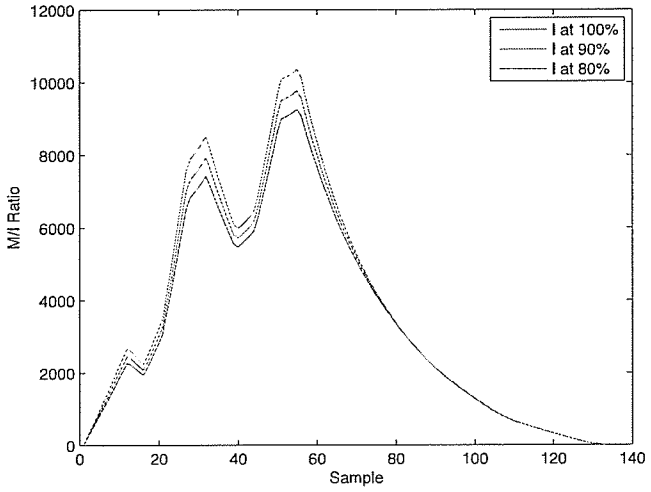


Figure 5.9: Curves showing M/I ratio when Main Girder is Failing.

distinct trend in the M/I ratio; as a girder fails, the M/I ratio will increase. Since, for the purposes of classification using girder ratios, the M/I ratio is directly comparable to the strain values from the measurement system, the strain values recorded by a sensor on a girder will increase as the girder fails.

Despite this promising result, recall from Section 5.2.2 that the absolute value is useless for classification and these findings cannot be used directly since the expected response for an unknown vehicle is itself unknown. In other words, if the baseline value is not known, a deviation from the baseline cannot be measured. However, these findings can be applied to the girder ratio measurements, since a baseline for the ratios does exist that can be applied to any unknown vehicle. Figure 5.11 shows the girder ratios presented in Table 5.1 for a truck in the Normal Lane, failing Girder 3; this plot is of C_i vs Girder Ratio, where C_i is used according to Equation 5.8. These ratios are calculated without noise and can therefore be considered an ideal situation and impractical, but they represent the idea of a baseline

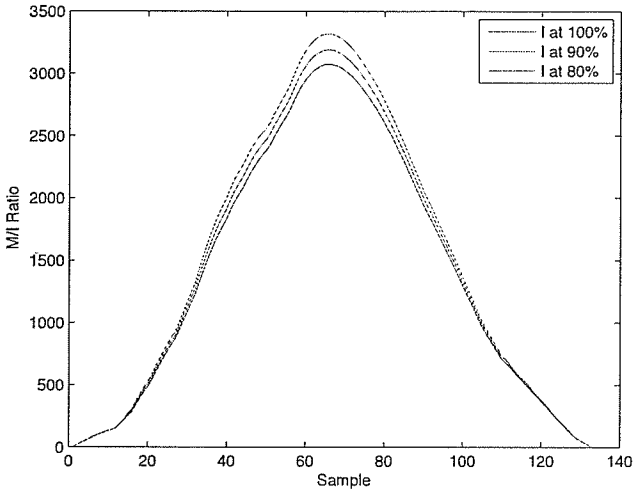


Figure 5.10: Curves showing M/I ratio when Other Girder is Failing.

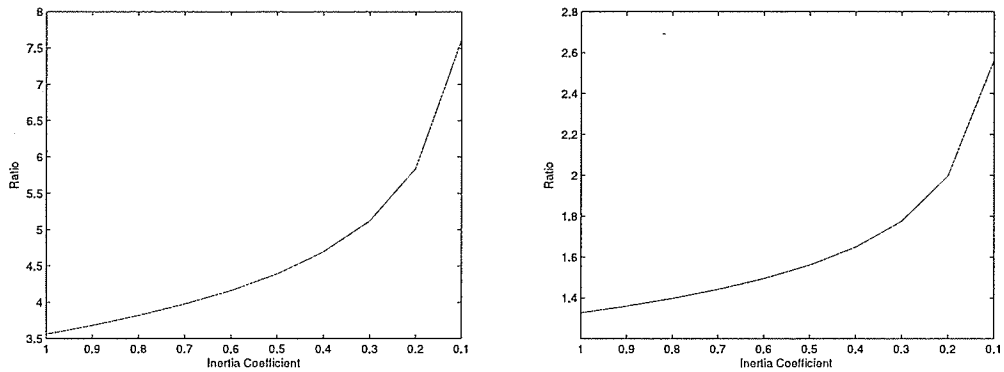
measurement that can be applied to any and all events.

$$I_{new} = C_i I_o, C_i = 1, 0.9, 0.8, \ldots, 0.1 \tag{5.8}$$

Figure 5.11 shows that there is a considerable change in the ratios for large changes in I. An equation exists which relates changes in I to a cracking failure. This is given in Equation 5.9 [23], which shows the percentage of change in I to a given percentage of the girder that is cracked.

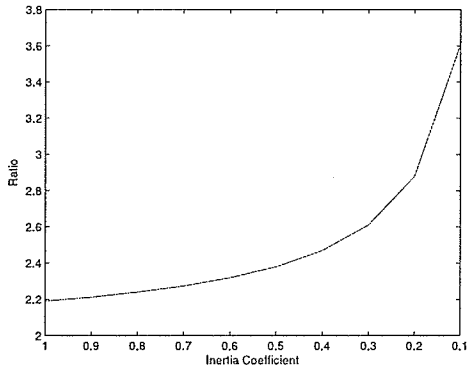
$$I_c \propto \left(1 - \frac{\text{crack height}}{\text{girder height}}\right)^3 I_o \tag{5.9}$$

Where I_c is the I for a cracked girder, $\frac{\text{crack height}}{\text{girder height}}$ represents the percentage of the girder that the crack occupies, and I_o is the moment of inertia for the uncracked girder. Using Equation 5.9 shows that a 2% crack causes a 5% decrease in I, while a 5% crack will cause a 15% decrease in I.

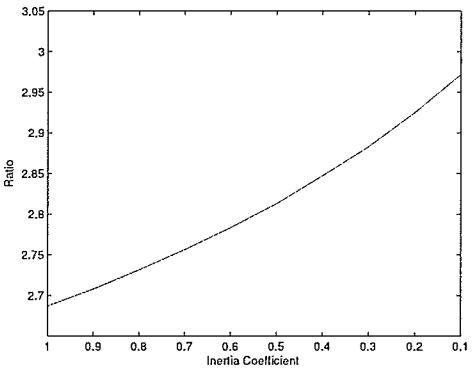


(a) $\frac{G3}{G1}$

(b) $\frac{G3}{G2}$



(c) $\frac{G3}{G4}$



(d) $\frac{G2}{G1}$

Figure 5.11: Girder Ratios for Varying Degrees of Failure.
Girder 3 Failing, Truck in Normal Lane.

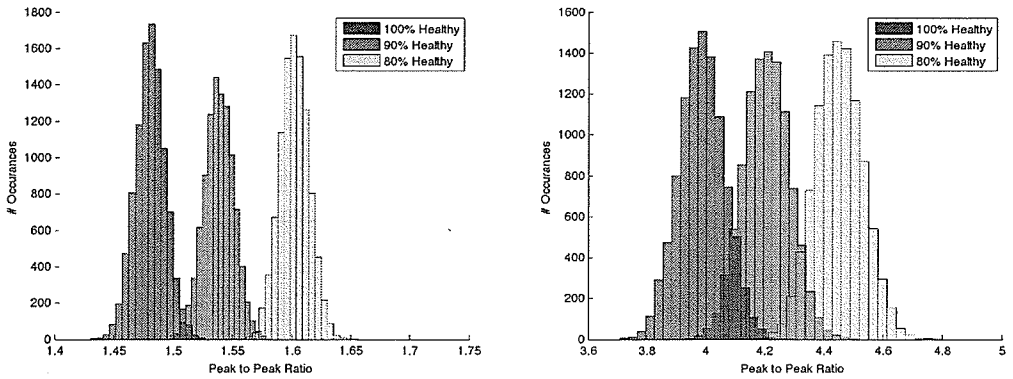
5.4.8 Failure Simulation

Since it has been shown that modeling failure using a decrease in I is reasonable, and that small cracks will result in larger variations in I, the next step in the project is to simulate realistic failure to try and generate a classification approach. Simulating realistic failure involves adding noise to the SECAN results, and then performing the same processing as described in Section 4.1.2, mainly by smoothing the signal.

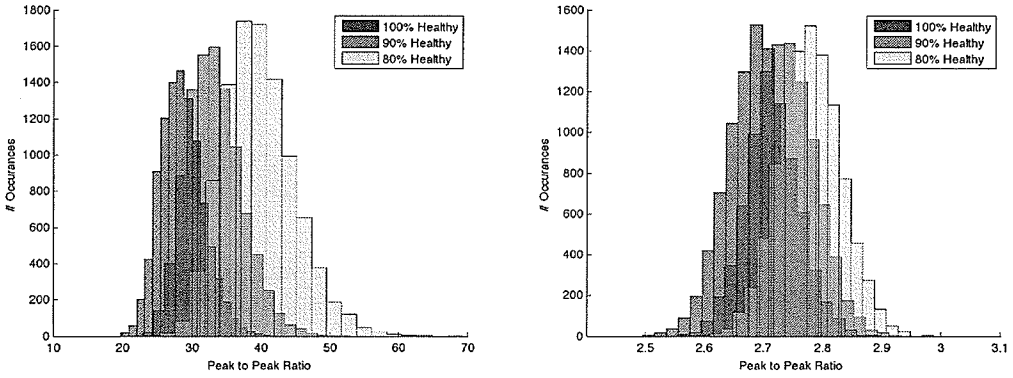
5.4.8.1 Failure Simulation with Noise

Noise is simulated into the system by using the Matlab 7.7 random number generator to generate a vector of Gaussian Noise, distributed $N(0, 216.80)$. This vector is then added to the SECAN results to create a noisy event curve. This adds noise to one curve; in order to generate a variety of cases, this process is done 10,000 times. The same SECAN vector is used in each case, with a different noise vector being used. The resultant vector is then smoothed using the Gaussian Smoothing algorithm described in Section 4.1.2.1. The ratios described in Table 5.1 are then computed for all 10,000 cases and then stored. This process was repeated for different levels of failure and for all four possible girder failures. The results were plotted in histograms; an example of these histograms is shown in Figure 5.12.

Figure 5.12 shows histograms for three failure cases for the four ratios when a truck is in the Passing Lane and Girder 1 is failing. These three failure cases are with I healthy, with $C_i = 0.9$ and with $C_i = 0.8$. The SECAN data for Figure 5.12 was generated with Truck A at full weight. To show the impact of weight classes when noise is involved, Figure 5.13 shows the same case, but with Truck A at 50% weight. By using the amount of visual separability as a measure of ease of classification, it appears more difficult to classify the 50% case over the 100% case. Extrapolating on this idea, the amount of visual separability can be measured to determine the expected ability of a ratio to perform as a classifier. This was done subjectively using a simple ranking system of 5 (very accurate classification) down to 1 (impossible to classify). Ratios that are ranked a 3 or higher were deemed usable to classify. Figure 5.14 shows an example of the 5 rankings. Non-integer rankings were assigned for some cases (for example 3.5), indicating that they are slightly more visible than one ranking, but not enough to qualify for the next ranking.

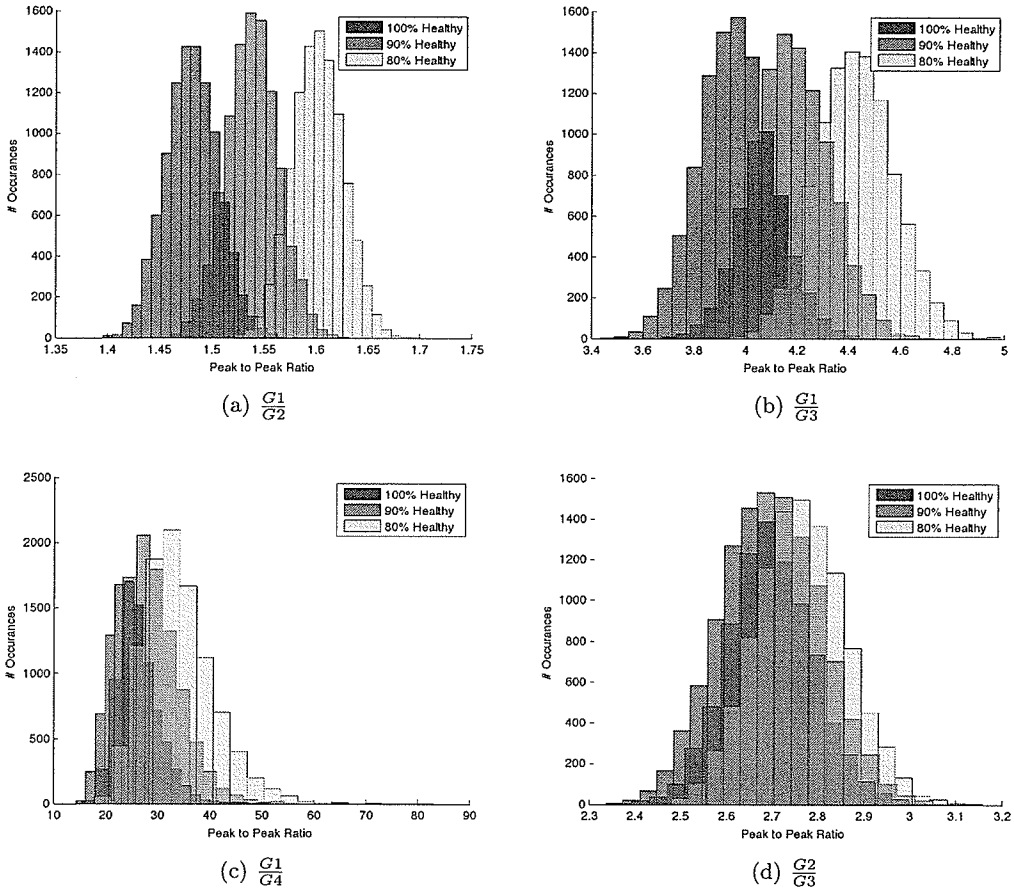


(a) $\frac{G1}{G2}$ (b) $\frac{G1}{G3}$



(c) $\frac{G1}{G4}$ (d) $\frac{G2}{G3}$

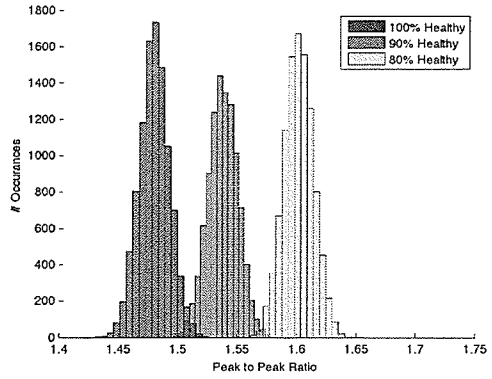
Figure 5.12: Girder Ratio Histograms for Varying Degrees of Failure.
Girder 1 Failing, Truck in Passing Lane, Full Weight.



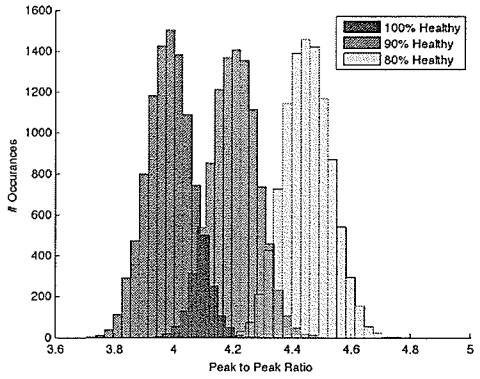
(a) $\frac{G1}{G2}$ (b) $\frac{G1}{G3}$

(c) $\frac{G1}{G4}$ (d) $\frac{G2}{G3}$

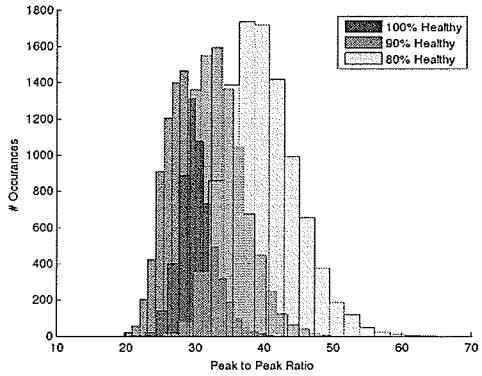
Figure 5.13: Girder Ratio Histograms for Varying Degrees of Failure.
Girder 1 Failing, Truck in Passing Lane, 50% Weight.



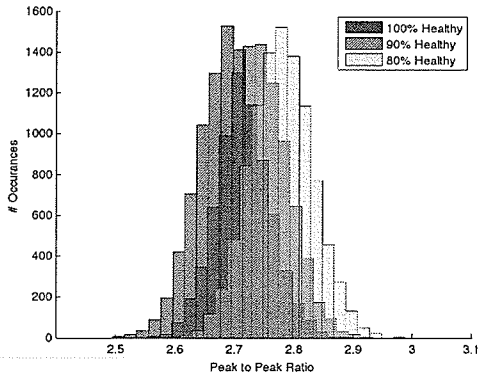
(a) Separability Level 5.



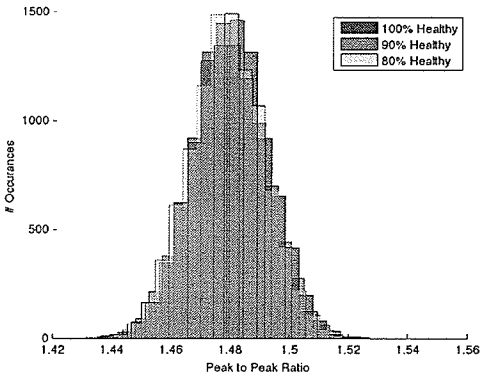
(b) Separability Level 4.



(c) Separability Level 3.



(d) Separability Level 2.



(e) Separability Level 1.

Figure 5.14: Separability Examples.

Many simulations were used to generate the separability rankings. Three different weight classes were used, 100%, 75% and 50%, with all four girders failing, and with vehicles in both lanes. This resulted in 3 x 4 x 2 = 24 different cases, each with four ratios. The complete set of rankings are provided in Tables 5.3 and 5.4.

Table 5.3: Separability of Passing Lane Girder Ratios.

Weight	Failed Girder	$\frac{G1}{G2}$	$\frac{G1}{G3}$	$\frac{G1}{G4}$	$\frac{G2}{G3}$
100	1	5	4	3	2
75	1	5	4	2	1
50	1	4	3	2	1
100	2	5	3.5	2.5	1
75	2	5	3	1	1
50	2	4	2	1	1
100	3	1	3.5	1	1
75	3	1	3	1	1
50	3	1	1.5	1	1
100	4	1	1	1	1
75	4	1	1	1	1
50	4	1	1	1	1

Table 5.4: Separability of Normal Lane Girder Ratios.

Weight	Failed Girder	$\frac{G3}{G1}$	$\frac{G3}{G2}$	$\frac{G3}{G4}$	$\frac{G2}{G1}$
100	1	5	2	1	5
75	1	5	1.5	1	4.5
50	1	3.5	1	1	3
100	2	4	5	1	1
75	2	3	3.5	1	1
50	2	2	3	1	1
100	3	3	4	1	1
75	3	2	3	1	1
50	3	1	2	1	1
100	4	1	1	3.5	1
75	4	1	1	3	1
50	4	1	1	1	1

In Tables 5.3 and 5.4, the cells highlighted in green show the best channel for each set of ratios, provided that the channel is rated 3 or higher. Any cell in blue is unusable for classification. Cells in white are usable for classification but do not provide the best discrimination. Examining the colouring of columns allows the removal of certain ratios. In Table 5.3, it can be seen that neither the $\frac{G1}{G4}$ ratio nor the $\frac{G2}{G3}$ ratio can detect any failures. As a result, there is no need to monitor these two ratios. In Table 5.4, there is no ratio that is not used, so all four ratios must be monitored. The results of Tables 5.3 and 5.4 are promising, as it shows that there is at least one ratio that allows for failure detection for any failing girder. These ratios are given in Table 5.5.

Table 5.5: Girder Ratios which are able to Classify.

Failing Girder	Ratio(s) to Use, Passing Lane	Ratio(s) to Use, Normal Lane
1	$\frac{G1}{G2}, \frac{G1}{G3}$	$\frac{G3}{G1}, \frac{G2}{G1}$
2	$\frac{G1}{G2}, \frac{G1}{G3}$	$\frac{G3}{G1}, \frac{G3}{G2}$
3	$\frac{G1}{G3}$	$\frac{G3}{G2}$
4	N/A	$\frac{G3}{G4}$

5.4.9 Classifier Creation

Once the ratios have been identified in terms of separability, the next task is to determine how exactly to use these results in a classifier. The histograms in Figure 5.15 show overlapping Gaussian distributions; this is in essence a textbook problem which would typically be used to demonstrate Bayesian classification [24]. This would work by determining the probability that a given ratio falls in any of the distributions and classifying the ratio according to the distribution from which it is most likely to have come. The problem with this

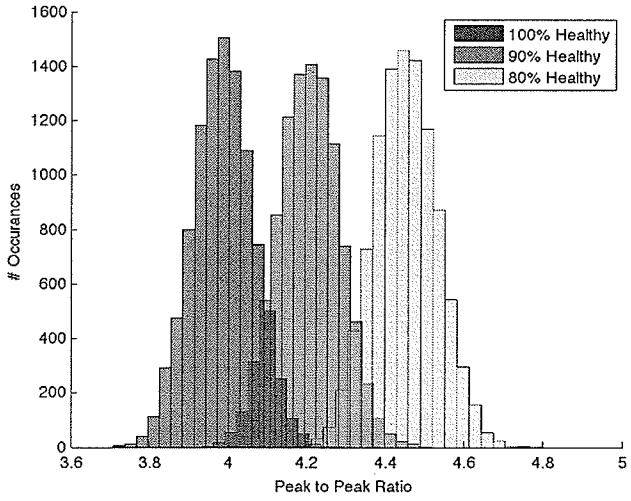


Figure 5.15: Histograms for Classification.

approach is that girder failure is not divided into a set of discrete distributions. The failure spectrum is continuous and to classify using a distribution generated from 90% healthy does not make for a realizable solution. The main reason this solution is unrealizable is that most bridges that are newly built do not have the data necessary to generate the failure distribution.

A different solution to the classification problem is simply to use the healthy distribution, represented by blue in Figure 5.15. Since the histogram is shaped like a normal distribution, it is possible to model the probability that a ratio belongs to the healthy distribution. The proprieties of the standard deviation say that 95% of the values of a normal distribution lie within two standard deviations of the mean (that is, $P(|X - \mu| < 2\sigma) = 0.95$). Since both the mean and standard deviation of the healthy distribution can be obtained from a physical system through data mining and training algorithms, this method is easily realizable.

The classification system is implemented by dividing the problem into 18 sub-problems. These are the following ratios: $\frac{G1}{G2}$, $\frac{G1}{G3}$, $\frac{G3}{G1}$, $\frac{G3}{G2}$, $\frac{G3}{G4}$ and $\frac{G2}{G1}$, each with three weight classes. The weight classes are defined by the relative amplitude of the peak in the main girder to the noise level in the system, N, and are shown in Table 5.6. The class boundaries were found experimentally; any peak that is below the left boundary for Class 3 is considered as as having an insignificant amplitude with respect to noise, and is thrown out.

Table 5.6: Weight Class Boundaries.

Lane	Class Identifier	Left Boundary	Right Boundary
Passing	3	$10N$	$30N$
Passing	2	$30N$	$50N$
Passing	1	$50N$	∞
Normal	3	$10N$	$28N$
Normal	2	$28N$	$38N$
Normal	1	$38N$	∞

Each of the 18 sub-problems contains its own mean and standard deviation to represent a healthy distribution.

Equation 5.10 shows the formula for updating an individual bin count in the classification algorithm, where μ_b is the mean of the bin, σ_b is the standard deviation of the bin, and rat_i is the ratio currently being examined.

$$\text{count}(b) = \begin{cases} \text{count}(b) + 1 & |rat_i - \mu_b| > 2\sigma_b \\ \max(0, \text{count}(b) - 1) & \text{otherwise} \end{cases} \tag{5.10}$$

For events in the Passing Lane, each event updates two bins; for events in the Normal Lane, each event updates four bins. Events that are classified as an Unknown Lane do not update anything. The classification parameters found are listed in Table 5.7, and were obtained from analyzing the tests done in Section 5.4.8.1.

Table 5.7: Classification Parameters.

Failing Girder	Weight Class	μ	σ
$\frac{G1}{G2}$	1	1.4796	0.0127
$\frac{G1}{G2}$	2	1.4799	0.0169
$\frac{G1}{G2}$	3	1.4800	0.0257
$\frac{G1}{G3}$	1	3.9829	0.0772
$\frac{G1}{G3}$	2	3.9760	0.1005
$\frac{G1}{G3}$	3	3.9564	0.1458
$\frac{G3}{G1}$	1	3.3392	0.0752
$\frac{G3}{G1}$	2	3.3275	0.0971
$\frac{G3}{G1}$	3	3.3031	0.1394
$\frac{G3}{G2}$	1	1.2979	0.0143
$\frac{G3}{G2}$	2	1.2978	0.0189
$\frac{G3}{G2}$	3	1.2977	0.0286
$\frac{G3}{G4}$	1	2.0695	0.0318
$\frac{G3}{G4}$	2	2.0671	0.0420
$\frac{G3}{G4}$	3	2.0620	0.0616
$\frac{G2}{G1}$	1	2.5730	0.0599
$\frac{G2}{G1}$	2	2.5642	0.0770
$\frac{G2}{G1}$	3	2.5462	0.1112

5.4.10 Classifier Testing

In order to determine how well the classifier presented above works, some testing needed to be done. To test the classifier, SECAN data was used once again in place of real event data. However, to ensure realism, data was taken from the physical system to help steer the generated events in the right direction.

5.4.10.1 Generating Realistic Events

In order to generate realistic events, distribution parameters needed to be extracted from the physical system. The parameters that were needed are the percentage of events that are in the passing lane and the normal lane, as well as the distribution of event amplitudes. For simplicity, it is assumed that each event has a peak proportional to how much it weighs with respect to Truck A. Since Section 5.4.5 showed that, without noise, event curves can be directly scaled by a multiplying constant for differently weighted vehicles, the peak values found from the physical system can be normalized with respect to the peak of the controlled tests using Truck A to generate an amplitude distribution. To supply actual data for this, data was taken from the North Perimeter Bridge on August 22nd, 2007. Two SECAN vectors are used, one for Truck A in the Passing Lane, and another with Truck A in the Normal Lane.

Since Truck A generates different event signatures depending on the source lane, the source lane of each of the physical events are first found. For events originating in the Passing Lane, the amplitude is divided by 60 microstrain, the smoothed peak value for Controlled Test 5; for events originating in the Normal Lane, the amplitude is divided by 37 microstrain, the smoothed peak value for Controlled Test 4. Since the event detector found 1,509 events on August 22nd, 2007, a random number is generated between 1 and 1,509. This number is then used to select an event, giving an amplitude multiplier to use and a source lane.

5.4.10.2 Results

The simulation described above was run many times to test different error cases. Tables 5.8 and 5.9 show the results of classifying simulation data. For these tables, the category C_i indicates the percentage of I used, *Count* is the bin count at the end of the simulation,

and *Max Count* is the peak value that the bin count reached throughout the simulation. For each of these simulations, the number of events generated is 10,000.

Table 5.8: Baseline Simulation Classification, all Girders Healthy.

Bin	Bin Name	Count	Max Count
1	G12W1	0	2
2	G12W2	0	2
3	G12W3	1	8
4	G13W1	0	3
5	G13W2	0	2
6	G13W3	2	8
7	G31W1	1	3
8	G31W2	1	2
9	G31W3	0	6
10	G32W1	0	2
11	G32W2	0	3
12	G32W3	0	5
13	G34W1	0	3
14	G34W2	0	2
15	G34W3	1	6
16	G21W1	1	2
17	G21W2	1	3
18	G21W3	0	7

Table 5.9: Error Simulation Classification Results.

Bin	Bin Name	C_i	Count	Max Count	C_i	Count	Max Count	C_i	Count	Max Count
Failing Girder 1										
1	G12W1	95	273	273	90	601	601	85	619	619
2	G12W2	95	39	39	90	289	289	85	304	304
3	G12W3	95	0	13	90	679	679	85	1190	1190
4	G13W1	95	7	30	90	487	487	85	615	615
5	G13W2	95	1	5	90	154	155	85	263	263
6	G13W3	95	0	7	90	1	21	85	446	447
7	G31W1	95	8	22	90	435	435	85	469	469
8	G31W2	95	1	6	90	286	286	85	393	393
9	G31W3	95	4	10	90	725	725	85	1948	1948
10	G32W1	95	0	4	90	0	3	85	9	9
11	G32W2	95	1	3	90	0	4	85	0	4
12	G32W3	95	0	5	90	0	7	85	0	8
13	G34W1	95	0	4	90	0	2	85	0	3
14	G34W2	95	0	2	90	0	3	85	0	2
15	G34W3	95	0	9	90	0	5	85	0	11
16	G21W1	95	2	11	90	377	377	85	469	469
17	G21W2	95	0	5	90	206	206	85	381	381
18	G21W3	95	2	10	90	228	235	85	1482	1482
Failing Girder 2										
1	G12W1	95	76	76	90	465	465	85	514	514
2	G12W2	95	10	16	90	232	233	85	294	294
3	G12W3	95	0	16	90	422	425	85	1020	1022
4	G13W1	95	0	6	90	164	165	85	464	464
5	G13W2	95	0	6	90	15	18	85	164	164
6	G13W3	95	8	17	90	2	22	85	445	448
7	G31W1	95	0	6	90	225	225	85	474	474
8	G31W2	95	1	4	90	26	27	85	232	232
9	G31W3	95	1	9	90	7	14	85	522	524
10	G32W1	95	0	13	90	376	376	85	514	514
11	G32W2	95	0	4	90	158	158	85	336	337
12	G32W3	95	0	8	90	2	40	85	1025	1028
13	G34W1	95	0	3	90	0	5	85	0	3
14	G34W2	95	0	2	90	0	2	85	1	3
15	G34W3	95	1	7	90	2	5	85	1	9
16	G21W1	95	0	4	90	0	3	85	0	4
17	G21W2	95	1	1	90	0	3	85	1	3
18	G21W3	95	1	7	90	0	6	85	0	7
Failing Girder 3										
1	G12W1	95	0	3	90	0	4	85	0	7
2	G12W2	95	0	4	90	1	4	85	0	4
3	G12W3	95	1	8	90	1	10	85	2	11
4	G13W1	95	0	4	90	92	93	85	366	366
5	G13W2	95	0	3	90	13	19	85	189	189
6	G13W3	95	1	14	90	0	13	85	204	206
7	G31W1	95	0	5	90	8	18	85	307	307
8	G31W2	95	0	3	90	1	8	85	34	34
9	G31W3	95	1	7	90	0	6	85	4	11
10	G32W1	95	0	5	90	258	258	85	502	502
11	G32W2	95	0	3	90	37	37	85	247	247
12	G32W3	95	1	6	90	0	21	85	149	159
13	G34W1	95	0	2	90	1	7	85	0	4
14	G34W2	95	0	3	90	0	4	85	2	4
15	G34W3	95	0	7	90	0	7	85	1	7
16	G21W1	95	0	2	90	0	4	85	0	3
17	G21W2	95	0	2	90	0	2	85	0	4
18	G21W3	95	0	6	90	2	6	85	0	7

Continued on next page...

Table 5.9: *Continued from previous page...*

Bin	Bin Name	C_i	Count	Max Count	C_i	Count	Max Count	C_i	Count	Max Count
Failing Girder 4										
1	G12W1	95	0	4	90	0	3	85	1	4
2	G12W2	95	2	4	90	0	3	85	0	3
3	G12W3	95	0	11	90	0	10	85	0	6
4	G13W1	95	0	4	90	0	2	85	0	3
5	G13W2	95	0	2	90	0	3	85	0	3
6	G13W3	95	0	9	90	0	7	85	1	11
7	G31W1	95	0	2	90	1	3	85	1	5
8	G31W2	95	0	2	90	0	3	85	1	3
9	G31W3	95	0	5	90	0	6	85	1	5
10	G32W1	95	1	3	90	0	4	85	2	5
11	G32W2	95	0	3	90	0	3	85	0	3
12	G32W3	95	4	8	90	4	9	85	0	8
13	G34W1	95	0	7	90	152	152	85	373	373
14	G34W2	95	0	2	90	0	8	85	166	167
15	G34W3	95	1	5	90	0	12	85	91	93
16	G21W1	95	0	2	90	0	4	85	1	4
17	G21W2	95	0	2	90	0	4	85	0	2
18	G21W3	95	0	6	90	0	6	85	1	5

Tables 5.8 and 5.9 show the results of 13 simulations. The baseline simulation, 100% Healthy where $C_i = 100$, is shown in Table 5.8 and can be used to show what type of counts can be expected from standard operation. The other simulations in Table 5.9 show varying degrees of failure and what type of bin counts can be expected for these failures. The results here are very useful; they agree with the results found in Tables 5.3 and 5.4 and show that for each girder failing, there are bins that will show the failure. By comparing bin counts to the maximum they attained in Table 5.9, it follows that for cases that can detect an error, the error count will steadily increase as more and more events are entered into the classifier.

5.4.10.3 Error Count Threshold

In order to make a decision on whether or not a failure is happening, some threshold must be placed on the error counts. This threshold should be high enough that it is impossible to reach under normal operating conditions, but not so high that it would take several months to identify a fault. Judging from the error counts in Table 5.8, the bin count in the fault free situation never exceeds a value of 10. In reality, this value might be a little tight, and a value between 30 and 50 may be more ideal. Once an error count has exceeded the pre-determined threshold, an alert can be raised that there is a problem with the bridge and that it should be examined. To summarize how the simulations and classifications work, Figure 5.16 consists of a flowchart of the algorithms used.

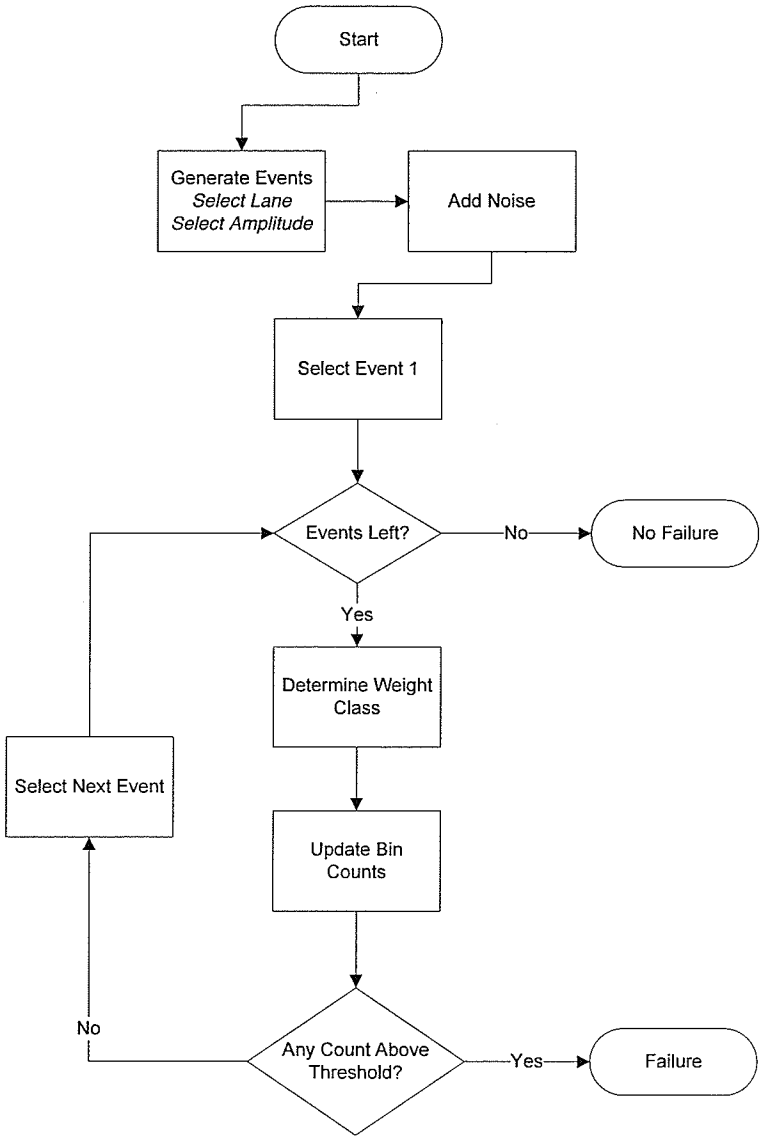


Figure 5.16: Flowchart of Simulation and Classification Algorithm.

5.5 Finalized Classifier

With the simulations complete and a theoretical classification system tested to be functioning, the classification system developed can be applied to the physical system from the Red River-North Perimeter Bridge to see how well it functions outside of the idealized simulation enviroment used for development.

5.5.1 Classifying Real Data with Simulation Parameters

The first attempt at classifying real data was to classify the data previously taken from the Red River-North Perimeter Bridge for August 22nd, 2007 with the classifier using the weight classes from Table 5.6 and the parameters given in Table 5.7. This proved to be a disastrous approach; the results are summarized in Table 5.10.

Table 5.10: Real Data Classification, Simulation Parameters.

Bin	Bin Name	Count	Max Count
1	G12W1	52	52
2	G12W2	22	23
3	G12W3	111	112
4	G13W1	80	80
5	G13W2	40	41
6	G13W3	165	167
7	G31W1	74	74
8	G31W2	48	48
9	G31W3	352	353
10	G32W1	38	39
11	G32W2	24	24
12	G32W3	152	152
13	G34W1	62	62
14	G34W2	34	35
15	G34W3	208	208
16	G21W1	76	76
17	G21W2	50	50
18	G21W3	394	394

Table 5.10 makes the claim that all four girders are failing all at the same time; this is clearly not the case as the Red River-North Perimeter Bridge has not failed.

5.5.2 Classifying Real Data with Real Parameters

Upon investigation of the poor results of the prior classification, the cause of the problem was quickly identified. The simulation parameters, while comparable to parameters obtained by analyzing the real data, were different enough that almost all events were flagged as errors. As a result, new parameters and weight classes were selected based on the August 22nd, 2007 data; these are summarized in Tables 5.11 and 5.12.

Table 5.11: Weight Class Boundaries, Real Data.

Lane	Class Identifier	Left Boundary	Right Boundary
Passing	3	$10N$	$25N$
Passing	2	$25N$	$40N$
Passing	1	$40N$	∞
Normal	3	$10N$	$20N$
Normal	2	$20N$	$30N$
Normal	1	$30N$	∞

The most notable difference between these parameters and the simulation parameters in Table 5.10 is that σ is almost a full order of magnitude larger in most cases. This can be attributed to the differences between the real data and the simulation data. The simulation assumes a constant level of noise, that physical effects are not present, and that each vehicle has the same shape of signature. When these assumptions are relaxed, the potential for greater variance between events exists. In order to test the new parameters, data was taken from the Red River-North Perimeter Bridge for August 23rd, 2007. The results of classifying these new events with the parameters in Table 5.12 are presented in Table 5.13, showing data that is consistent with the results recorded in Table 5.8 and lead to the same threshold that was discussed earlier in Section 5.4.10.3; that is, one between 30 and 50.

Table 5.12: Classification Parameters, Real Data.

Failing Girder	Weight Class	μ	σ
$\frac{G1}{G2}$	1	1.4147	0.0937
$\frac{G1}{G2}$	2	1.4295	0.1499
$\frac{G1}{G2}$	3	1.4970	0.2320
$\frac{G1}{G3}$	1	3.3110	0.6850
$\frac{G1}{G3}$	2	3.3528	0.9768
$\frac{G1}{G3}$	3	3.5177	1.2161
$\frac{G3}{G1}$	1	2.3476	0.3840
$\frac{G3}{G1}$	2	2.3137	0.6699
$\frac{G3}{G1}$	3	2.4015	0.7678
$\frac{G3}{G2}$	1	1.3824	0.1055
$\frac{G3}{G2}$	2	1.3540	0.1562
$\frac{G3}{G2}$	3	1.3886	0.2163
$\frac{G3}{G4}$	1	1.8730	0.1954
$\frac{G3}{G4}$	2	1.8545	0.1816
$\frac{G3}{G4}$	3	1.8675	0.2297
$\frac{G2}{G1}$	1	1.6912	0.1896
$\frac{G2}{G1}$	2	1.6863	0.2726
$\frac{G2}{G1}$	3	1.7069	0.2977

Table 5.13: Real Data Classification, Real Data Parameters.

Bin	Bin Name	Count	Max Count
1	G12W1	0	3
2	G12W2	0	2
3	G12W3	0	3
4	G13W1	0	2
5	G13W2	0	2
6	G13W3	0	2
7	G31W1	0	5
8	G31W2	0	3
9	G31W3	0	2
10	G32W1	0	6
11	G32W2	0	6
12	G32W3	0	4
13	G34W1	0	5
14	G34W2	2	8
15	G34W3	1	4
16	G21W1	0	4
17	G21W2	0	3
18	G21W3	0	2

5.5.3 Sensor Failures vs Structure Failures

The entirety of the discussion up to this point has stressed detecting structure failures. However, in a realistic implementation, the structure is not the only point of failure in the system. The measurement system can also fail, and a failure of the measurement system should not conclude that a structure failure has occurred.

There are two types of potential sensor failures for the measurement system. The first type of sensor failure is a non-fatal sensor failure. This type of failure is a short term anomaly, one that the sensor will eventually correct. An example of this is a sudden jump in the zero-offset, followed by a jump back down. For this category of sensor failure, the process of keeping an error count will succeed in preventing the system from concluding that a structure failure has occurred.

The second type of sensor failure is a complete and persistent failure, where something is physically wrong with the sensor. Examples of complete failure are the strain gauge becoming de-bonded from the girder, wire corrosion, or a power supply issue. These issues are impossible to differentiate from a structure failure without some form of redundancy in the system. Despite having two usable sensors per girder, a failure in the bridge is only detectable in a *girder height* $*$ *girder height* square located around a crack [19]; an

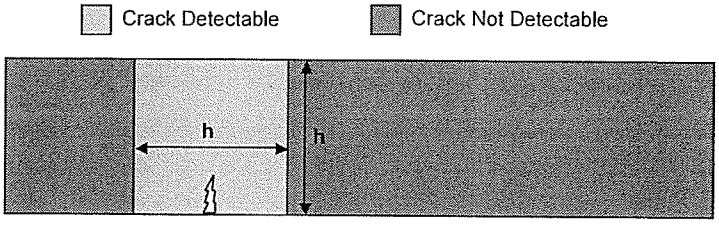


Figure 5.17: Locations where a Cracking Fault is Detectable by Strain Sensors.

example is shown in Figure 5.17. Figure 5.17 shows a cracking fault in a girder, and the area where a sensor would need to be located for this fault to appear in the strain values. Any sensors located outside this area will not show any variations in strain values [19]. This is an unfortunate situation since it removes any redundancy between sensors. Theoretically, a sensor failure could be detected provided the behaviour is so anomalous that it cannot possibly be caused by a slight crack in a girder. Examples of this would be strain readings on the order of thousands of microstrain, or a large negative strain.

The system described in this report is not fully realized; as a result, a method to determine fatal sensor failures was not implemented, but it would be a simple matter to place different error cases in the classification system, if it is required. In reality, however, a fatal sensor failure will require the sensor to be replaced; to replace a sensor, an individual must be sent to the bridge. When a girder failure is seen, the system will conclude that the bridge must be examined, thus having a fatal sensor failure also conclude that the bridge must be examined may not be an unwanted consequence.

Numerous methods exist to automate the collection of parameters for the algorithm. These are briefly discussed in Section 6.2.

Chapter 6

Conclusions

6.1 Conclusions

The algorithms developed in this project show exciting opportunities for the field of structural health monitoring. All three of these algorithms solve important problems that may prevent effective automated monitoring. The algorithms designed, while focused on solving specific problems encountered with the measurement system installed on the Red River-North Perimeter Bridge, should be capable of performing well on any general system, due to the fundamentals of the algorithms being model independent.

The preprocessing algorithm attempts to remove environmental fluctuations in recorded strain measurements in situations where a model for the fluctuations cannot be found. The algorithm shows that it is possible to remove unwanted effects without damaging the desired information; this eliminates the need for prior knowledge about when the effects are present and how they manifest themselves. It can also eliminate unexpected effects that were not predicted, provided they match the profile of slow, general trending effects.

The event detection algorithm accurately removes events using dynamic thresholds based on information present in the signal itself. This requires no pre-existing knowledge about expected amplitudes of events or distributions of the background noise. The detection algorithm succeeded in isolating events in the signal. While the detection rate is not 100%, it is high enough that the missed events do not pose any problems with the remainder of the project.

The event classification algorithm successfully identified failure in the case where failure was simulated by a decrease in the moment of inertia for a girder. A simulation was used to show that a classifier based on a derivative of Bayesian classification is accurate and able to detect when weakening occurs in the girders. While it is unrealistic to expect to have data on how an arbitrary bridge responds when failing, it is possible to derive a set of expected parameters for good operation, and monitor for deviations from these. The classification algorithm uses an error count to ensure that isolated or short term anomalies in the sensing network, or periods of larger than expected noise, do not cause an error on the structure to be reported. Using an error count can be interpreted as waiting for a persistent error condition to surface before reporting any problems.

The work presented in this project showed that, while using absolute peak values of the strain measurements was not a good feature for classification, several features that were immediately derivable from the peak values, the girder ratios, the weight class, and the source lane, were capable of performing the task of classification.

In conclusion, the project shows that it is possible to take imperfect measurement data, remove the unwanted effects and extract a classification feature, in this case, the event; both of these are shown to work on real-world data. In addition, the project shows that a classifier based on detecting anomalous behaviour can identify failure. This was demonstrated using simulation, but could not be tested using real-world measurements as there was no data available for the bridge in this study which represented a failure state.

6.2 Future Work

Despite the promise shown by this report, there is room to improve on the project. First and foremost, a learning algorithm can be designed and implemented to allow the system to determine automatically the proper parameters that it needs for classification, those being the means and standard deviations for each girder ratio for a healthy structure. As well, different classification schemes could be developed to try and complement the scheme presented, with a goal of improving the classification rate. An example of this would be to try and sort the events by vehicle type as well as weight. Work can also be undertaken in an attempt at lowering the standard deviation measured for a healthy structure.

Because of the potential for seasonal drifting, it would be a worthy exercise to determine the quality of the classifier when applied to a season that is not the training season, for

example, classify winter data using a summer training set. If the results are unusable, a possible solution would be to have two models for expected behaviour. One would cover the summer data, and another could cover winter data, with the proper model being used at specific times of the year.

In expanding on the knowledge of how the different girder ratios are expected to react, discussed in Section 5.4.8.1, it may be possible to implement an algorithm that can interpret error counts to determine the location of the failure. Using this information, it may also be worth trying to find redundancy that allows for fatal sensor errors to be determined.

In addition to expanding the concepts developed in this project, the problems discussed reveal ways that may improve the physical implementation of the system. A suggestion is to move the entire measurement resistor bridge to the same environment, so that all four of its components fluctuate identically with temperature. At the very least, the measurement node should be taken out of the cabinet where the heater resides.

Another potential application of this project to the physical implementation is in decimating data for storage. The event detector results in a bounding window for each event; it can be assumed that any data that is not part of a window is therefore simply background noise and contains no information worth storing. As a result, the amount of stored data could be reduced by a considerable amount if the only samples stored are those that fall in an event window.

The project also leads to a recommendation for proper monitoring of an entire structure. Because a fault is detectable only if a sensor falls within a portion of the girder surrounding the fault, a complete sensor network would need to have, at the very least, sensors placed at appropriate separations across entire girders; recall that this separation would be the girder height. It would also be beneficial to implement some sensor redundancy, which would require this distance to be smaller; this spacing should be at most half of the girder height.

References

[1] Manitoba Infrastructure and Transportation, *Construction Drawings of Red River Bridge on PTH-101*, Correspondence, 2009.

[2] J. S. Wilson, ed., *Sensor Technology Handbook.* Massachusetts: Newnes, 2005.

[3] J. Pople, "Errors and uncertainties in strain measurements employing metal foil gauges," (Tavistock, Devon, UK), pp. 532 – 74, 1984.

[4] C. Tanaphatsiri, W. Jaikla, and M. Siripruchyanun, "A current-mode wheatstone bridge employing only single do-cdta," in *Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on*, pp. 1494–1497, 30 2008-Dec. 3 2008.

[5] "Introduction to digital signal processing," Tech. Rep. TN-517, Vishay Micro-Measurements, Aug 2007.

[6] "Strain gage thermal output and gage factor variation with temperature," Tech. Rep. TN-504-1, Vishay Micro-Measurements, Aug 2007.

[7] M. Brauwers and F. Brouers, "Temperature and strain effect on electrical resistivity of transition metal alloys: application to strain gauges," *Journal of Physics F: Metal Physics*, vol. 6, no. 7, pp. 1331–1339, 1976.

[8] B. Studyvin, R. Doty, and R. Repplinger, "Temperature effects on strain gages used on aerospace nickel hydrogen batteries," in *Battery Conference on Applications and Advances, 1999. The Fourteenth Annual*, pp. 325–328, 1999.

[9] G. Rutherford, "Model-free bridge-based vehicle classification," Master's thesis, University of Manitoba, Winnipeg, Manitoba, Aug 2008.

[10] V. Fil'chikov, V, "Normalizing transducers of strain-gauge sensors with correction of temperature sensors," *Telecommunications and Radio Engineering*, vol. 49, no. 11, pp. 92–97, 1995.

[11] J. Huang, "System engineering and data m anagement for structural health monitoring," Master's thesis, Dalhousie University, Halifax, Nova Scotia, June 2007.

[12] S. Haykin and M. Moher, *Introduction to Analog and Digital Communications.* New York: John Wiley & Sons, second ed., 2007.

[13] X. Luo, C. Peng, and X. Guo, "Using morphological filters to extract spiky transients in eeg," in *Neural Interface and Control, 2005. Proceedings. 2005 First International Conference on*, pp. 72–74, May 2005.

[14] R. C. Gonzalez and R. E. Woods, *Digital Image Processing.* New Jersey: Prentice Hall, second ed., 2002.

[15] P. Tadejko and W. Rakowski, "Mathematical morphology based ecg feature extraction for the purpose of heartbeat classification," in *Computer Information Systems and Industrial Management Applications, 2007. CISIM '07. 6th International Conference on*, pp. 322–327, June 2007.

[16] M. Sedaaghi, "Direct implementation of open-closing in morphological filtering," *Electronics Letters*, vol. 33, pp. 198–199, Jan 1997.

[17] P. Kersten, "Fuzzy order statistics and their application to fuzzy clustering," *Fuzzy Systems, IEEE Transactions on*, vol. 7, pp. 708–712, Dec 1999.

[18] Robert Bosch GmbH, *CAN Specification, Version 2.0*, Sept 1991.

[19] Dr. A. Mufti, Correspondence, 2009.

[20] Dr. D. Sidhu, Correspondence, 2009.

[21] A. Mufti et. al., "Load tests on north perimeter red river bridge." First Draft, 2008.

[22] X. Jin, Q.-p. Zhong, and G.-s. Yang, "Stress damage failure analysis and prediction of beams under pure bending loads," *Gong Cheng Li Xue/Engineering Mechanics*, vol. 16, no. 1, pp. 123 – 127, 1999. [Abstract].

[23] Dr. E. El-Salakawy, Correspondence, 2009.

[24] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification.* New York: John Wiley & Sons, second ed., 2001.

Appendix A

SECAN Parameters

This appendix contains a list of structural parameters to be used for modeling the Red River-North Perimeter Bridge in SECAN, as supplied by Dr. D. Sidhu [20]. All distances are in meters.

- No. of Harmonics: 40
- No. of Girders: 5
- Span Length: 24
- E of Girder Material: 2e8
- G of girder Material: 8e7
- No. of Diaphragms: 4
- No. of intermediate supports: 0
- Girder Spacings, starting from left:
  - 2.126
  - 2.743
  - 2.743
  - 2.743
- Moment of inertia of girders, starting from left:
  - 0.03767352
  - 0.03387336
  - 0.03387336
  - 0.03387336
  - 0.03456791
- Torsional inertia of girders, starting from left:
  - 0.2099809e-4
  - 0.1358832e-4
  - 0.1358832e-4
  - 0.1358832e-4
  - 0.1362817e-4
- Slab Thickness: 0.2
- E of Slab Material: 2.8e7
- G of Slab Material: 1.2e7
- Equivalent Shear Area: 0

- E value of Diaphragms: 2e8
- Distance of Diaphragms from left abutment:
  - 4.087
  - 8.931
  - 14.985
  - 19.828
- I value of Diaphragms:
  - 0.109881e-2
  - 0.109881e-2
  - 0.109881e-2
  - 0.109881e-2

Appendix B

Matlab Code

The following appendix contains the Matlab source files for the algorithms.

B.1 Event Detection (Shell)

```
function EventDetection(input_file, output_file)

outfile = char(output_file);

STDEV_COEFF = 0.6745;
PEAK_DISTANCE_THRESHOLD = 100;

FIRST_TO_READ = 1; CURRENT_OFFSET = 0; READING_AMOUNT = 15000;

done = false;

dataFormat = '%f, %f, %f, %f, %f, %f, %f, %f, %f, %f,
   %f, %f, %f, %f, %f, %f, %f, %f';
FI = fopen(char(input_file),'r');

incData = zeros(18,READING_AMOUNT);

peaks = ones(15,250000) .* Inf;
allData = ones(250000,3) .* Inf;

totalEvents = 0;

while (~done)
    if (FIRST_TO_READ == 1)
        READING_AMOUNT = READING_AMOUNT * 2;
    else
        incData(:,1:READING_AMOUNT/2) = incData(:,READING_AMOUNT/2+
           1:READING_AMOUNT);
        CURRENT_OFFSET = CURRENT_OFFSET + READING_AMOUNT/2;
    end

    for j = FIRST_TO_READ:(FIRST_TO_READ-1) + READING_AMOUNT/2
        tempData = fgets(FI);
        tempData = tempData(find(tempData==',',1)+2:length(tempData));
        tempData = sscanf(tempData,dataFormat);
        incData(:,j) = tempData;
        if (feof(FI))
            done = true;
            break;
        end
    end
    incData = incData(:,1:j);

    if (FIRST_TO_READ == 1)
        READING_AMOUNT = READING_AMOUNT/2;
```

```
        FIRST_TO_READ = FIRST_TO_READ + READING_AMOUNT/2;
        currentPeak = zeros(1,5);
        currentEnd = zeros(1,5);
        currentStart = zeros(1,5);
    end

    [temp numElements] = size(incData);
    clear temp;

    [allMod,allSmoothed,dummyNoise] = Preprocessing(incData,numElements);

    [totalEvents,peaks,allData,currentStart,currentEnd,currentPeak] =
       EventDetector(CURRENT_OFFSET,STDEV_COEFF,PEAK_DISTANCE_THRESHOLD,
       READING_AMOUNT,done,dummyNoise,allSmoothed,allSmoothed,currentStart,
       currentEnd,currentPeak,totalEvents,peaks,allData);

    %Resets to help cascading
    currentStart = currentStart - READING_AMOUNT/2;
    currentStart(isinf(currentStart)) = 0;
    currentStart(isnan(currentStart)) = 0;

    currentEnd = currentEnd - READING_AMOUNT/2;

    currentPeak = max(currentPeak - READING_AMOUNT/2,0);
    currentPeak(isinf(currentPeak)) = 0;
    currentPeak(isnan(currentPeak)) = 0;
end

save(outfile)
fclose(FI);
```

B.2 Preprocessing Algorithm

```
function [allCleaned,allSmoothed,dummyNoise] = Preprocessing(incData,numElements)
%Performs the required preprocessing algorithms.

    allOrig = zeros(15,numElements);
    smallScaleFixed = zeros(15,numElements);

    for n = 1:16
        channel = incData(n+1,:);
        allOrig(n,:) = channel;

        y = ones(1,300);
        closed = imclose(channel,y);
        closed = imopen(closed,y);

        opened = imopen(channel,y);
        opened = imclose(opened,y);

        smallScaleFixed(n,:) = (closed+opened)/2;

        if (n == 16)
            events = channel .* (channel > opened);
            events(~events) = opened(~events);
            events = events .* (events < closed);
            events(~events) = closed(~events);
            events = channel - events;
        end
    end

    allCleaned = allOrig - smallScaleFixed -
             repmat(median(allOrig-smallScaleFixed,2),1,numElements);
    allSmoothed = zeros(15,numElements);
    dummyNoise = allCleaned(16,:) - events;

    for k = 1:16
        allSmoothed(k,:) = GaussianSmooth(allCleaned(k,:),5,21,2);
    end
end
```

B.3 Event Detection

```
function [totalEvents,peaks,allData,currentStart,currentEnd,currentPeak] =
   EventDetector(CURRENT_OFFSET,STDEV_COEFF,PEAK_DISTANCE_THRESHOLD,
   READING_AMOUNT,done,dummyNoise,scanningSignal,peakValueSignal,currentStart,
   currentEnd,currentPeak,totalEvents,peaks,allData)

    allEnds = ones(100,5) .* Inf;
    allStarts = ones(100,5) .* Inf;
    allMaxes = ones(100,5) .* Inf;
    allCurrMax = ones(100,5) .* Inf;
    numEvents = 0;
    currentMax = zeros(1,5);

    THRESHOLD = 4 * STDEV_COEFF * std(dummyNoise);

    eventDetected = 1;

    while (eventDetected > 0 && (done || min(currentStart) <= READING_AMOUNT/2))
        eventDetected = 0;
        for channel = 1:5
            channelFoundEvent = 0;
            chan = scanningSignal(channel,:);
            currMax = 0;

            if (currentPeak(channel) == 0)
                for j = max(currentEnd(channel)+1,1):length(chan)
                   if (chan(j) >= THRESHOLD && (currMax == 0 || chan(j)
                      > chan(currMax)))
                       currMax = j;
                   end

                   if (currMax > 0 && chan(j) < STDEV_COEFF*std(dummyNoise))
                        currentEnd(channel) = j;
                        currentPeak(channel) = currMax;
                        currentMax(channel) = chan(currMax);
                        eventDetected = 1;
                        channelFoundEvent = 1;

                        for k = currMax:-1:1
                            if (chan(k) < STDEV_COEFF*std(dummyNoise))
                                currentStart(channel) = k;
                                break;
                            end
                        end
                        break;
                   end
```

```
            end
        else
            if (currentPeak(channel) < Inf)
                eventDetected = 1;
                channelFoundEvent = 1;
            end
        end
        if (channelFoundEvent == 0)
            currentPeak(channel) = Inf;
            currentStart(channel) = Inf;
        end
    end

    newPeak = min(currentPeak);

    if ((done || min(currentStart) <= READING_AMOUNT/2)  &&
       length(find(isinf(currentPeak)==1)) < 5)
        numEvents = numEvents + 1;
        for channel = 1:5
            if (abs(currentPeak(channel) - newPeak) < PEAK_DISTANCE_THRESHOLD)
                allEnds(numEvents,channel) = currentEnd(channel);
                allStarts(numEvents,channel) = currentStart(channel);
                allMaxes(numEvents,channel) = currentMax(channel);
                allCurrMax(numEvents,channel) = currentPeak(channel);

                currentPeak(channel) = 0;
            else
                allEnds(numEvents,channel) = -Inf;
                allStarts(numEvents,channel) = Inf;
                allMaxes(numEvents,channel) = -Inf;
                allCurrMax(numEvents,channel) = -Inf;
            end
        end
    end
end
[temp numElements] = size(peakValueSignal);
clear temp;

[totalEvents,peaks,allData] = EventParameterDetection(
   CURRENT_OFFSET,numElements,numEvents,peakValueSignal,allMaxes,
   allEnds,allStarts,totalEvents,peaks,allData);
end
```

B.4 Event Parameter Detection

```
function [totalEvents,peaks,allData] = EventParameterDetection(CURRENT_OFFSET,
   numElements,numEvents,signal,allMaxes,allEnds,allStarts,
   totalEvents,peaks,allData)
%Determines window and peak values

    i = 1:numElements;

    wind = ones(numEvents,numElements);
    for eventNumber = 1:numEvents;
        maxes = allMaxes(eventNumber,:);
        ends = allEnds(eventNumber,:);
        starts = allStarts(eventNumber,:);

        peakChan = find(maxes==max(maxes));

        starts = starts(max(1,peakChan-2):min(5,peakChan+2));
        ends = ends(max(1,peakChan-2):min(5,peakChan+2));

        starts = starts(~isnan(starts));
        starts = starts(~isinf(starts));
        ends = ends(~isnan(ends));
        ends = ends(~isinf(ends));


        MADstart = median(abs(starts - median(starts)));
        MADend = median(abs(ends - median(ends)));

        startDevs = (starts-median(starts))/MADstart;
        endDevs = (ends-median(ends))/MADend;

        for j = 1:length(starts)
            if (abs(startDevs(j)) > 3.5)
                starts(j) = inf;
            end
            if (abs(endDevs(j)) > 3.5)
                ends(j) = -inf;
            end
        end

        theStart = min(starts);
        theEnd = max(ends);

        wind(eventNumber,:) = ones(1,numElements) .* (i>=theStart) .* (i<=theEnd);
    end
```

```
    b = totalEvents;
    for j=1:15;
        for k = b+1:b+numEvents
            peaks(j,k) = max(signal(j,:).*wind(k-b,:));
            allData(k,:) = [CURRENT_OFFSET find(wind(k-b,:)==1,1)
               find(wind(k-b,:)==1, 1,'last')];
        end
    end

    totalEvents = totalEvents + numEvents;

end
```

B.5 Event Classification (Simulation)

```
clear; close all;

fail =  0;
STD_COEFF = 2;
%define error bins
G12W = [1 2 3];
G13W = [4 5 6];
G31W = [7 8 9];
G32W = [10 11 12];
G34W = [13 14 15];
G21W = [16 17 18];

bins = zeros(1,18);
binMaxes = zeros(1,18);

means = zeros(1,18); stdevs = zeros(1,18);

%Define classifier parameters
means(G12W(1)) = 1.4796; stdevs(G12W(1)) = 0.0127;
means(G12W(2)) = 1.4799; stdevs(G12W(2)) = 0.0169;
means(G12W(3)) = 1.4800; stdevs(G12W(3)) = 0.0257;
means(G13W(1)) = 3.9829; stdevs(G13W(1)) = 0.0772;
means(G13W(2)) = 3.9760; stdevs(G13W(2)) = 0.1005;
means(G13W(3)) = 3.9564; stdevs(G13W(3)) = 0.1458;

means(G31W(1)) = 3.3392; stdevs(G31W(1)) = 0.0752;
means(G31W(2)) = 3.3275; stdevs(G31W(2)) = 0.0971;
means(G31W(3)) = 3.3031; stdevs(G31W(3)) = 0.1394;
means(G32W(1)) = 1.2979; stdevs(G32W(1)) = 0.0143;
means(G32W(2)) = 1.2978; stdevs(G32W(2)) = 0.0189;
means(G32W(3)) = 1.2977; stdevs(G32W(3)) = 0.0286;
means(G34W(1)) = 2.0695; stdevs(G34W(1)) = 0.0318;
means(G34W(2)) = 2.0671; stdevs(G34W(2)) = 0.0420;
means(G34W(3)) = 2.0620; stdevs(G34W(3)) = 0.0616;
means(G21W(1)) = 2.5730; stdevs(G21W(1)) = 0.0599;
means(G21W(2)) = 2.5642; stdevs(G21W(2)) = 0.0770;
means(G21W(3)) = 2.5462; stdevs(G21W(3)) = 0.1112;

%Generate Events
numEvents = 5000;
noise = 216.8;

PLcount = zeros(1,3);
NLcount = zeros(1,3);
```

```
events = EventGenerator(numEvents,3,100);

events = GaussianSmooth(events,5,21,2);

for i=1:numEvents
    peaks = squeeze(max(events(1:5,:,i),[],2)); peaks(5:-1:1) = peaks(1:5);
    peaks = peaks';

    lane = find(peaks==max(peaks)); lane = (lane-1)/2 + 1;
    maxPeak = max(peaks);
    if (lane == 1)
        %Determine weight class 1, 2 or 3
        if (maxPeak > 10*noise && maxPeak <= 30*noise)
            w = 3;
        elseif (maxPeak > 30*noise && maxPeak <= 50*noise)
            w = 2;
        elseif (maxPeak > 50*noise)
            w = 1;
        else
            w = -1;
        end

        if (w>0)
            G1G2 = peaks(1)/peaks(2);
            G1G3 = peaks(1)/peaks(3);

            PLcount(w) = PLcount(w)+1;

            if (abs(G1G2 - means(G12W(w))) > STD_COEFF*stdevs(G12W(w)))
                bins(G12W(w)) = bins(G12W(w))+1;
                binMaxes(G12W(w)) = max(binMaxes(G12W(w)),bins(G12W(w)));
            else
                bins(G12W(w)) = max(0,bins(G12W(w))-1);
            end
            if (abs(G1G3 - means(G13W(w))) > STD_COEFF*stdevs(G13W(w)))
                bins(G13W(w)) = bins(G13W(w))+1;
                binMaxes(G13W(w)) = max(binMaxes(G13W(w)),bins(G13W(w)));
            else
                bins(G13W(w)) = max(0,bins(G13W(w))-1);
            end
        else
            fail = fail+1;
        end
    elseif (lane == 2)
        %Determine weight class 1, 2 or 3
        if (maxPeak > 10*noise && maxPeak <= 28*noise)
            w = 3;
        elseif (maxPeak > 28*noise && maxPeak <= 38*noise)
```

```
            w = 2;
        elseif (maxPeak > 38*noise)
            w = 1;
        else
            w = -1;
        end

        if (w>0)

            NLcount(w) = NLcount(w)+1;

            G3G1 = peaks(3)/peaks(1);
            G3G2 = peaks(3)/peaks(2);
            G3G4 = peaks(3)/peaks(4);
            G2G1 = peaks(2)/peaks(1);

            if (abs(G3G1 - means(G31W(w))) > STD_COEFF*stdevs(G31W(w)))
                bins(G31W(w)) = bins(G31W(w))+1;
                binMaxes(G31W(w)) = max(binMaxes(G31W(w)),bins(G31W(w)));
            else
                bins(G31W(w)) = max(0,bins(G31W(w))-1);
            end
            if (abs(G3G2 - means(G32W(w))) > STD_COEFF*stdevs(G32W(w)))
                bins(G32W(w)) = bins(G32W(w))+1;
                binMaxes(G32W(w)) = max(binMaxes(G32W(w)),bins(G32W(w)));
            else
                bins(G32W(w)) = max(0,bins(G32W(w))-1);
            end
            if (abs(G3G4 - means(G34W(w))) > STD_COEFF*stdevs(G34W(w)))
                bins(G34W(w)) = bins(G34W(w))+1;
                binMaxes(G34W(w)) = max(binMaxes(G34W(w)),bins(G34W(w)));
            else
                bins(G34W(w)) = max(0,bins(G34W(w))-1);
            end
            if (abs(G2G1 - means(G21W(w))) > STD_COEFF*stdevs(G21W(w)))
                bins(G21W(w)) = bins(G21W(w))+1;
                binMaxes(G21W(w)) = max(binMaxes(G21W(w)),bins(G21W(w)));
            else
                bins(G21W(w)) = max(0,bins(G21W(w))-1);
            end
        else
            fail = fail+1;
        end

    else
        display(['Error with lane = ' num2str(lane)]);
    end
end
```