An extensible decision support system for bridge maintenance

Jeremy Barisch-Rooney

November 11, 2019

Contents

1 Introduction

The probability of a bridge to fail increases over time until it is no longer considered safe for use. Maintenance of a bridge is typically carried out when something goes wrong or according to a preventative maintenance schedule based on expert knowledge, neither approach making the best use of limited maintenance resources. Sensors can provide real-time information without the delay or cost of a manual maintenance check. What are the costs and benefits of installing a decision support system (DSS) based on real-time sensor data for the purpose of maintenance of a no-prestress no-postension concrete slab bridge? This question is the topic of this thesis, to answer it we need to answer a few sub questions. What analyses of sensor data do and do not provide valuable information to the user of a DSS? How can we collect sensor data for the purpose of analysis? And what are the costs and benefits of installing different types and quantites of sensors on a bridge?

A **decision support system** for bridge maintenance** is a software system that provides the user of the system with information on the current state of a bridge. The provided information should enable the user of the

system to make a more informed decision about when and/or where maintenance should be carried out. The provided information can include real-time sensor data and an analysis thereof. The primary aim of the analysis techniques applied to sensor data in this thesis are to classify data in two states, a normal or healthy state, and an abnormal or damaged state.

In the development of a DSS for bridge maintenance it is **necessary to have sensor data** corresponding to both the damaged and undamaged states of the bridge in question. Due to the expensive nature of a bridge it is usually and unsurprisingly not permitted to impose a damaged state on a bridge. The use of finite element (FE) software allows us to create simulated sensor data via a finite element model (FEM) of a bridge on which different damage states can be imposed.

A FEM is a [1] model of a structure in software. All models are wrong {TODO:REF Mike Lees' referenced something in his lecture} and thus the simulated sensor data generated by the FEM will **differ from sensor data collected from sensors installed in real life**. The simulated sensor data should be close enough to reality to provide confidence that the anaysis techniques in this thesis can also work with real sensor data, while acknowleding that additional work would likely be needed to tune the anaysis techniques once real sensor data is available. After all, "in theory there is no difference between theory and practice, while in practice there is", TODO:REF ambiquity of who said this.

A finite element simulation of a bridge can produce sensor data that is necessarily different from data collected from real installed sensors. It is necessary to **verify the simulated data against some ground truth** in order to have confidence in the accuracy of the simulated data. And it is necessary to **test the developed analysis techniques on real data** to have confidence in the techniques, for when a DSS for bridge maintenance is installed in real life.

OpenSees (The Open System for Earthquake Engineering Simulation) is an open-source FE software package that anyone with a Windows, macOS or Linux machine, and an internet connection, can download and install. Depending on open-source rather than proprietary FE software enables users to explore or extend this research, thus allowing for reproducible research.

A FEM of bridge 705 in Amsterdam was available at TNO (Netherlands Organisation for Applied Scientific Research) for the FE software package Diana. This FEM was verified against measurements from an experimental campaign. The existence of a verified FEM for bridge 705 allows the generated FEM for OpenSees to be verified by comparing responses under explicit loading conditions from the OpenSees model of bridge 705 to responses from

the verified Diana model under the same load.

This thesis could have gone one of two ways. The verified FEM of bridge 705 for Diana could have been used to simulate sensor responses for analysis. However Diana requires a relatively expensive proprietary licence for use (you must ask for a quote) and the file format of FEMs in Diana is rather awkward to modify. support this claim. By using OpenSees it was easier to target a greater number of bridges, by generating FEMs based on a high-level bridge specification. More importantly however OpenSees does not require a licence for use and is additionaly available for macOS users thus allowing for the research to be reproduced or extended.

Extensibility is a measure of the ability to extend a software system. According to TODO:REF "An important kind of reuse is extensibility, i.e., the extension of software without accessing existing code to edit or copy it." The research in this thesis is not just reproduceable but also extensible. The benefit of this research being extensible is that an interested party should be able to download the software and perform research on another bridge or investigate how a new classification technique performs on sensor data, without having to start from scratch. By creating an extensible system it is my hope to facilitate further research in this area and prevent duplication of effort.

TODO:Paragraph on classification

Bridge data corresponding to states normal and abnormal was not available, however data was avilable from viaducts corresponding to two states, high and low temperature. In this thesis the **analysis techniques are tested** on this data to provide, an albeit limited, test that the techniques can perform a classification between states on real data.

This thesis first gives an overview of existing research into machine learning approaches for structural health monitoring (SHM), decision support systems and classification techniques. The methods section presents an in-depth description of how an extensible system is created for the collection of simulated sensor responses, how the inputs to this system should be structured, and what form the data-driven classification experiments will take. In the results section we take a look at the data generated by the data collection system, analyze the results of the classification experiments, and finally, present the costs and benefits of installing a decision support system for bridge maintenance.

2 Abbreviations & Terminology

3 Background & Motivation

{This is where I tell the story, why I am doing this (motivation) and what has been and not been done before (background)}

Existing work on DSSs for bridge maintenance can be split into two categories. There are papers on.

3.1 Existing Bridges

The Dutch national main road network consists of 3,200km of road. Assets in the road network are divided into four categories: pavements, structures, traffic facilities and environmental assets. Each structure is categorized into a type that has its own maintenance characteristics. Table 1 outlines the catgorization of the 3,283 structures in the network.

Table 1: Structures in the Dutch national main road network. Each type of structure has its own maintenance characteristics. The table lists for each structure type the total number in the Dutch national main road network and the total deck area.

Stucture type	Number	Deck Area (m2)
Concrete bridge	3,131	3,319,002
Steel bridge (fixed)	88	301,997
Movable bridge	43	347,876
Tunnel	14	475,228
Aqueduct	7	86,491
Total	3,283	4,530,593

3.2 Bridge Maintenance

In this subsection we briefly review the costs of bridge maintenance, in particular focusing on the cost of maintenaining concrete Dutch bridges

Bridge maintenance is a requirement in the life-cycle of a bridge in order to extend the life of a bridge and keep it within operational conditions. The aims of bridge maintenance are

- Effective management of operational programs
- Realistic budgeting at national level

- Tuning bridge mainagement with other maintenance programs

TODO: Paragraph on overview of operational programs

Bridges are a type of structure that require a large investment, though they also have a long service life of 50 to 100 years. Annual maintenance costs are relatively small compared to the initial investment cost (<1%), however over the lifetime of the bridge the maintenance costs are on the order of the initial investment. The annual maintenance cost and the cost of replacement are given for each type of structure in the Dutch national main road network in Table 3.2.

Stucture type	Total Replacement Cost (€M)	Annual Maintenance Cost (€ M)
Concrete bridge	6,600	37
Steel bridge (fixed)	600	7
Movable bridge	1,100	10
Tunnel	1,700	13
Aqueduct	250	1
Total	10,250	68

The maintenance cost of a concrete bridge can be estimated by determining the maintenance cost of frequently used components such as concrete elements, extension joints and bearings. These costs estimates of the frequently used components first require a description of minimal acceptable condition of the components. Then, in combination with an estimation of maintanence intervals (which can come from subjective and conflicting sources) and prioritization of the available budgets, a maintenance plan of a bridge can be presented. An example of such a plan for a typical concrete highway bridge is shown in Figure 1.

The Dutch national road network contains over 3,000 highway bridges. Of these, most are 30 or more years old. A significant amount of bridges were constructed in the 1970s, which is typical for many Western European road networks. Fitting a Weibull distribution to the lifetime of demolished concrete bridges suggests an expected lifetime of 41 years. This in turn would mean that the many concrete bridges constructed in the 1970s and earlier would be due for replacement. However, of these demolished bridges, many were demolished due to a change in functional or economical requirements, rather than due to technical failure. Including the ages of current bridges in the fitted distribution increases the expected lifetime to 75 years, which is more in line with the design for 80 years of most Dutch highway bridges,

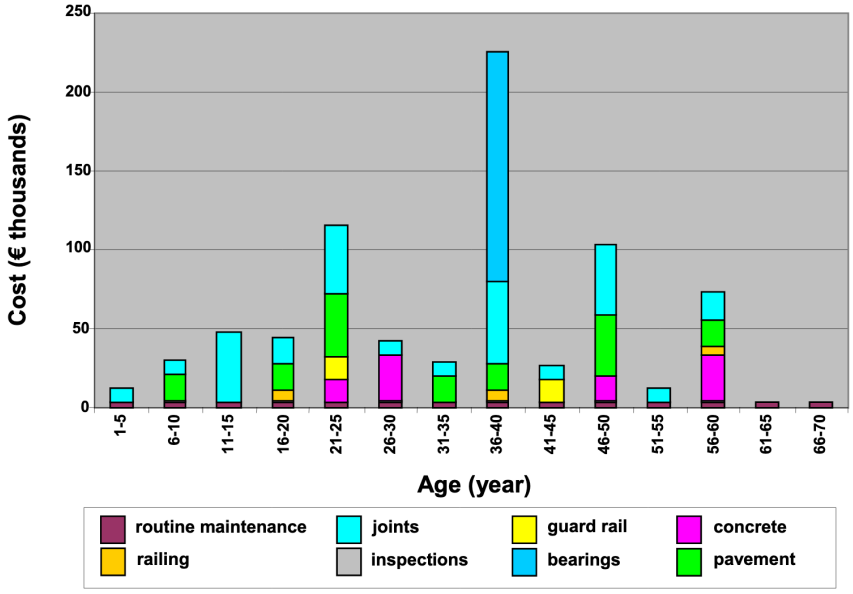

Figure 1: The maintenance cost of a typical concrete highway bridge. The y-axis shows the cost in thousands of euros. Each bar is for a period of five years and the cost is based on underlying components as indicated by the legend.

design codes in the Netherlands require a design lifetime between 50 and 100 years.

Figure 2 shows an initial peak in the expected cost of replacement of Dutch bridges, this is due to a combination of the distribution of when the current bridges were originally built (largely in the 1970s), their expected lifetime and their replacement cost. In an aging bridge stock the cost of maintenance can be assumed constant, averaged over the large number of structures. After a long time the cost of replacement will be approximately 85€ million, approximately half the cost of annual maintanence of concrete bridges at 37€ million.

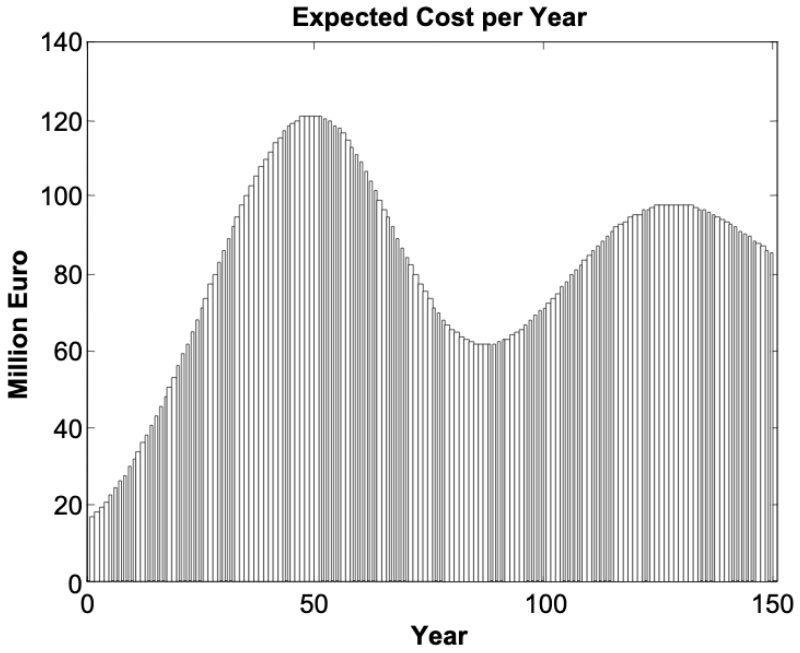

Figure 2: The expected cost of replacement of concrete bridges in the Dutch national main road network. The expected cost is calculated by summing over all concrete bridges, their ages and replacement costs. The initial peak is largely due to a surge in construction around the 1970s. The cost of replacement will tend to 85€ million in the long run.

3.3 Extensibility

In order for the developed DSS to be truly extensible it is not limited to depend on a single finite element program. The system has as a parameter a method of communication with a finite element program, such that data can be collected and analyzed from different finite element programs, in this case OpenSees and Diana.

Due to the expensive nature of installing sensors in real life and of damaging a bridge which is likely prohibited, the software system includes a component for simulating sensor responses from reinforced concrete bridges. In order for this simulation to be extensible and allow for further research on bridges other than bridge 705, the specification of the bridge is simply a parameter of the system.

The developed decision support system has a number of **parameters** such that users wishing to extend the software further are not limited to focus on bridge 705 or to use a specific finite element program. The specification of a bridge is a parameter of the system, as is the type and intensity of traffic on the bridge. Furthermore, as mentioned earlier, different finite element programs can be integrated with this system, which may be useful if a finite element model of a bridge for a different finite element program is already available to the user.

For a software system to be extensible, the source code must be available to any user wishing to extend said software. The benefits of **open source software** are well known, in particular open source software allows *any individual with an interest* to develop or *extend* the software. Open source software can thus leverage the knowledge of the community and prevent duplication of efforts which can occur when software is developed behind closed doors. Open source software also provides transparency to anyone wishing to investigate the software and may produce more reliable software due to more people having eyes on it.

3.4 Existing Work

A more in-depth look at what has been done before

This section contains a review of the most relevant material studied during this thesis work. The section begins with an overview of related works followed by a more in-depth look at the most relevant material. The aim of this section is to place the thesis in context and to provide background information to the reader on employed techniques. The section concludes by relating the reviewed material back to this thesis.

3.4.1 An overview

TODO: overview of related works

3.4.2 The application of machine learning to structural health monitoring

[2] illustrates the utility of a data-driven approach to structural health monitoring (SHM) by a number of case studies. In particular the paper focuses on pattern recognition and machine learning (ML) algorithms that are applicable to damage identification problems.

The question of *damage detection* is simply to identify if a system has departed from normal (i.e. undamaged) condition. The more sophisticated problem of *damage identification* seeks to determine a greater level of information on the damage status, even to provide a forecast of the likely outcome of a situation. The problem of detection and identification can be considered as a hierarchy of levels as described in [1].

- Level 1. (Detection) indication that damage might be present in the structure.
- Level 2. (Localization) information about the probable position of the damage.
- Level 3. (Assessment) an estimate of the extend of the damage.
- Level 4. (Prediction) information about the safety of the structure.

This paper argues that ML provides solutions to these problems at upto level 3, and that in general level 4 cannot be addressed by ML methods.

Applying ML for the purpose of SHM is usually only a single step in a broader framework of analysis. Figure 3 shows the waterfall model ([3]) which begins with sensing (when to record responses) and ends with decision making. ML methods are only step four in this model. An important part of this entire process is feature extraction, step three, which can be regarded as a process of amplification, transforming the data to keep only information that is useful for the ML analysis. Another aim of feature extraction is to reduce the dimensionality of the data, to avoid the explosive growth of the data requirements for training with the data dimensions, known as the *curse of dimensionality* TODO:REF.

An experiment was setup to identify damage on the wing of a Gnat artefact. Damage scenarios for testing were created by making a number of

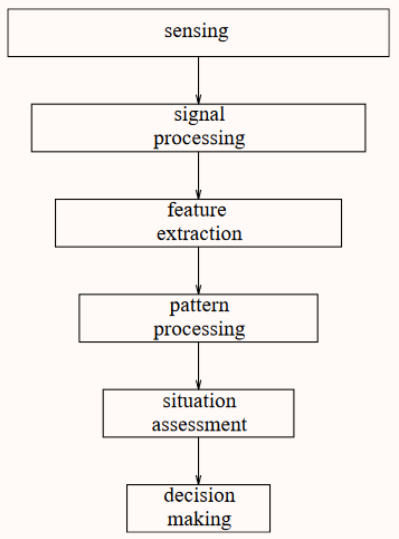

Figure 3: The *waterfall* model.

cuts into copies of the wing panel. Transmissibility between two points was chosen as a measurement based on success in a previous study TODO:REF, it is the ratio of the acceleration spectra between two points $A_j(\omega)/A_i(\omega)$. This was measured for two pairs of perpendicular points on each wing; in the frequency range 1-2kHz, which was found to be sensitive to the type of damage investigated. The measurements were transformed into features for novelty detection by manual investigation of 128-average transmissibilities from the faulted and unfaulted panels, selecting for each feature a range of spectral lines as shown in TODO:FIG. 18 features were chosen.

To address the first level of Rytter's hierarchy, damage detection, an outlier analysis was applied. This outlier analysis calculates a distance measure (the squared Mahalanobis distance) for each testing observation from the training set. 4 of the 18 features could detect some of the damaged scenarios and could detect all of the unfaulted scenarios, other features produced false positives and were discarded. Two combined features managed to detect all damage types and raised no false positives.

The second level of Rytter's hierarchy is damage localization. This problem can be approached as a regression problem, however here it is based on the classification work done for damage detection where transmissibilities are used to determine damage classes for each panel. A vector of damage indices for each of the panels is given as input to a multi-layer perceptron

(MLP) which is trained to select the damaged panel. The paper argues that "it may be sufficient to classify which skin panel is damged rather than give a more precise damage location. It is likely that, by lowering expectations, a more robust damage locator will be the result". This approach has an accuracy of 86.5%, the main errors were from two pairs of adjacent panels, whose damage detectors would fire when either of the panels were removed. The approach depends on the fact that damage is local to some degree, and the damage detectors don't fire in all cases, which was true in this case.

, the assessment was based on the previous detection technique.

3.4.3 Neural Clouds for monitoring of complex systems

In one-class classification, a classifier attempts to identify objects of a single class among all objects by learning from a training set that consists only of objects of that class. One-class classifiers are useful in the domain of system condition monitoring because often only data corresponding to the normal range of operating conditions is available. Data corresponding to the class of abnormal conditions, when a failure or breakdown of a system has occurred, is often not available or is difficult or expensive to obtain.

The Neural Clouds (NC) method presented in [4] is a one-class classifier which provides a confidence measure of the condition of a complex system. In the NC algorithm we are dealing with measurements from a real object where each measurement is considered as a point in n-dimensional space.

First a normalization procedure is applied to the data to avoid clustering problems in the subsequent step. The data is then clustered and the centroids of the clusters extracted. The centroids are then encapsulated with "Gaussian bells", and these Gaussian bells are normalized to avoid outliers in the data.

The summation of the Gaussian bells results in a height `h` for each point `p` on the hyperplane of parameter values. The value of `h` at a point `p` can be interpreted as the probability of the parameter values at `p` falling within the normal conditions represented by the training data.

In comparison to other one-class classifiers, the NC method has an advantage in condition monitoring in that it creates this unique plateau where height can be interpreted as probability of the system condition. Figure 4 shows this plateau in comparison with other one-class classifiers, Gaussian mixture and Parzen-window.

It is important to note that when significant changes occur in the normal state of the system, perhaps due to environmental changes, then the NC classifier should be retrained in order to avoid a false alarm. However, if a NC classifier is continually being retrained with real-time data then it may

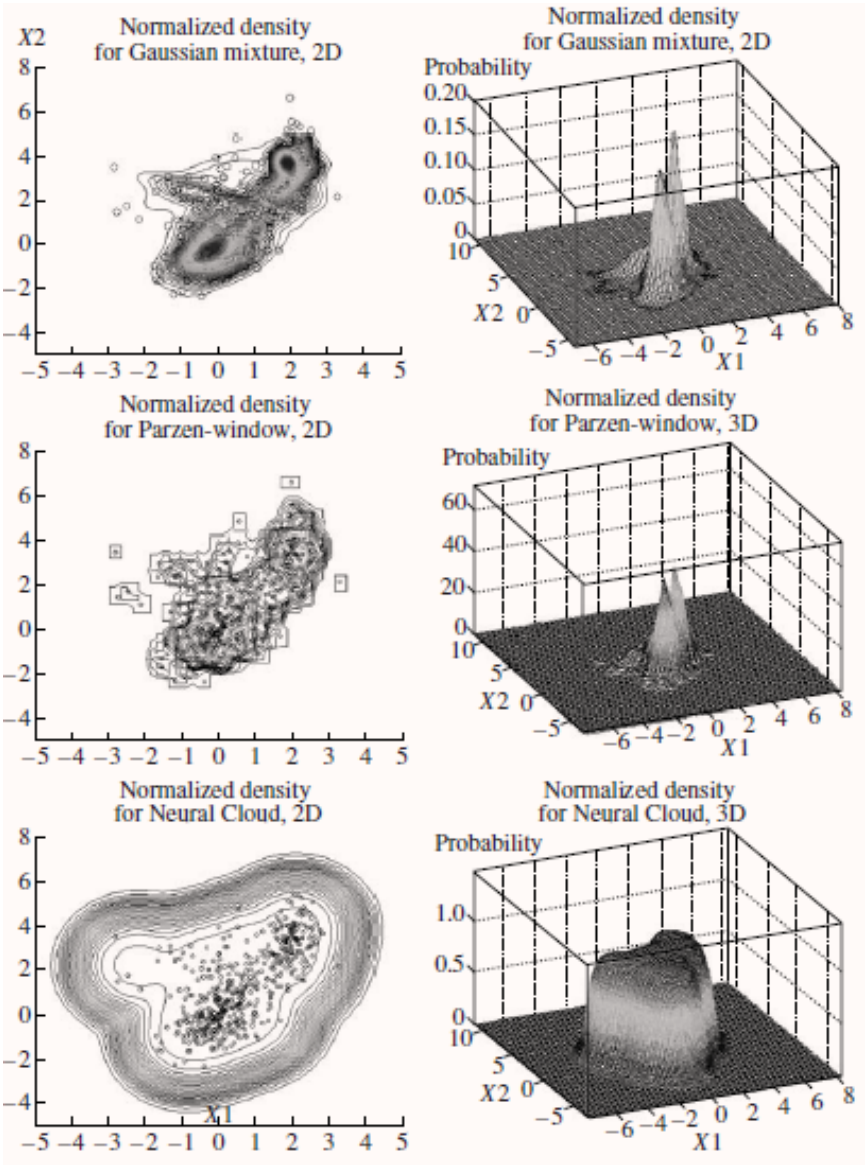

Figure 4: Comparison of Neural Clouds with other approaches, namely Gaussian mixture and Parzen-window. At the left side 2D contour line plots are pictures and at the right normalized density 3D plots.

not detect a gradual long-term change to the system.

3.4.4 Combining data-driven methods with finite element analysis for flood early warning systems

In [5] a system for real-time levee condition monitoring is presented based on a combination of data-driven methods and finite-element analysis. Levee monitoring allows for earlier warning signals incase of levee failure, compared to the current method of visual inspection. The problem with visual inspection is that when deformations are visiable at the surface it means that levee collapse is already in progress.

Data-driven methods are model-free and include machine learning and statistical techniques, whereas finite-element analysis is a model-based method. One advantage of data-driven methods are that they do not require information about physical parameters of the monitored system. As opposed to finite-element analysis which in the case of levee condition monitoring requires parameters such as slope geometry and soil properties. The model-based methods provide more information about the monitored object, but are more expensive to evaluate and thus difficult to use for real-time condition assessment.

In this paper the data-driven and finite-element components of the system which were developed are referred to as the Artificial Intelligence (AI) and Computer Model (CM) respectively. The AI and CM can be combined in two ways. In the first case the CM is used for data generation. Data is generated by the CM corresponding to normal and abnormal conditions. The normal behaviour data is used to train the AI and both the normal and abnormal behaviour data can be used for testing the AI. In the second case shown in Figure 5 the CM is used for validation of the alarms generated by the AI. If the AI detects abnormal behaviour then the CM is run to confirm the result. If the AI was correct a warning is raised, else the new data point is used to retrain the AI.

3.4.5 Flood early warning system: design, implementation and computational modules.

In [6] a prototype of an flood early warning system (EWS) is presented as developed within the UrbanFlood FP7 project. This system monitors sensors installed in flood defenses, detects sensor signal abnormalities, calculates failure probability of the flood defense, and simulates failure scenarios. All of this information is made available online as part of a DSS to help the

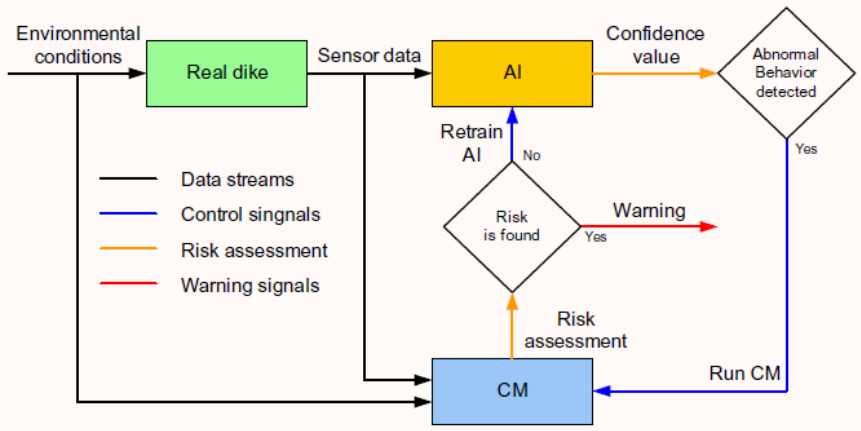

Figure 5: AI and CM...

relevant figure of authority make an informed decision in case of emergency or routine assessment.

Some requirements that must be taken into account in the design of an EWS include:

- Sensor equipment design, installation and technical maintenance.
- Sensor data transmission, filtering and analysis.
- Computational models and simulation components.
- Onteractive visualization technologies.
- Remote access to the system.

Thus it is clear that the development of an EWS or DSS consists of much more than the development of the software components, but must also take into account the installation of hardware and the transmission of information between components of the system. These many interacting components are shown in Figure 6 along with a description.

3.4.6 A clustering approach for structural health monitoring on bridges

In [7] a clustering based approach is presented to group substructures or joints with similar behaviour and to detect abnormal or damaged ones. The presented approach is based on the simple idea that a sensor located at a

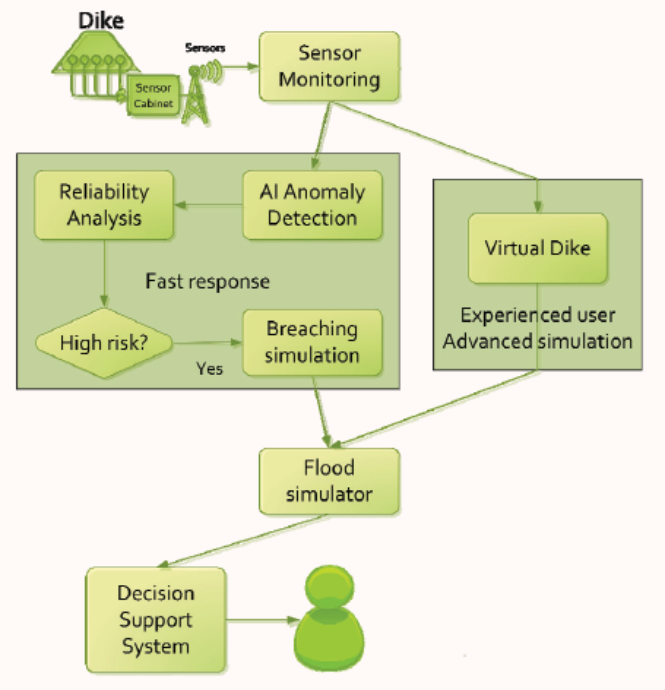

Figure 6: The *Sensor Monitoring* module receives data from the installed sensors which are then filtered by the *AI Anomaly Detector*. In case an abnormality is detected the *Reliability Analysis* calculates the probability of failure. If the failure probability is high then the *Breach Simulator* predicts the dynamics of the dike failure. A fast response is calculated beginning with the *AI Anomaly Detector* and ending with the *Breaching Simulator*. The *Virtual Dike* module is additionaly available for the purpose of simulation by expert users, but takes longer. The fast response and the response from the *Virtual Dike* module are both fed to the *Flood Simulator* which models the flooding dynamics, this information is sent to the decision support system to be made available to the decision maker.

damaged substructure or joint will record responses that are significantly different from sensors at undamaged points on the bridge.

The approach was applied to data collected from 2,400 tri-axial accelerometers installed on 800 jack arches on the Sydney Harbour Bridge. An *event* is defined as a time period in which a vehicle is driving across a joint. A pre-set threshold is set to trigger the recording of the responses by each sensor, each event is then represented by a vector of samples X.

Prior to performing any abnormality detection the data is preprocessed. First each event data is transformed into a feature $V_i = |A_i| - |A_r|$ where A_i is the instantaneous acceleration at the \$i\$th sample and A_r is the "rest vector" or average of the first 100 samples. The event data is then normalised as $X = \frac{V-\mu(V)}{\sigma(V)}$.

After normalisation of the event data, k-nearest neighbours is applied for outlier removal. One might consider that outliers are useful in the detection of abnormal conditions, since they represent abnormal responses. However if outlying data per joint are removed, then a greater level of confidence can be had when an abnormal condition is detected knowing that the result is not based on any outliers. In this outlier removal step the sum of the energy in time domain is calculated for event data as $E(X) = \sum_i |x_i|^2$. Then for every iteration of k-nearest neighbours, the k closest neighbours to the mean of the enery of the joint's signals μ_{joint} is calculated.

The event data is then transformed from the time domain into a series of frequencies using the Fast Fourier Transform (FFT), such that the original vibration data is now represented as a sequence that determines the importance of each frequency component in the signal. After this transformation a distance metric is calculated for each pair of event signals, this metric is used for k-means clustering of the data for anomaly detection. The distance metric used is the Euclidean distance: $dist(X, Y) = ||X - Y|| = \sqrt{\sum(x_i - y_i)^2}$.

Two clustering methods were applied, event-based and joint-based. In the event-based clustering experiment it was known beforehand that joint 4 was damaged. All event data was clustered using k-means clustering with $K = 2$ which resulted in a big cluster containing 23,849 events and a smaller cluster of 4662 events mostly located in joint 4. The percentage of events per joint in the big cluster are shown in Figure 7 where joint 4 is clearly an outlier.

A frequency profile of both the big and small cluster are shown in Figures 8 and 9. In case there is no knowledge of abnormal behaviour then this method can be used to separate outliers and obtain a profile of normal behaviour. In this research on SHB there was prior knowledge of a damaged joint. A frequency profile of an arbitrary joint and the damaged joint before

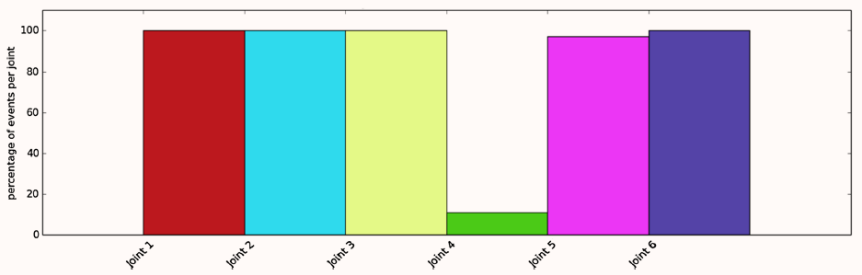

Figure 7: ...

and after repair is shown in Figure 10. The difference of the damaged profile to the other two is clear, which indicates that there is sufficient information in frequency information from accelerometers to detect abnormal joints.

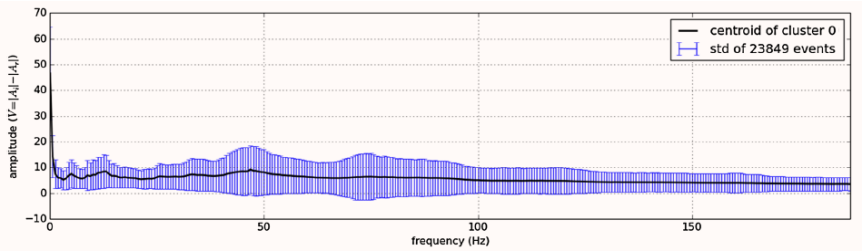

Figure 8: ...

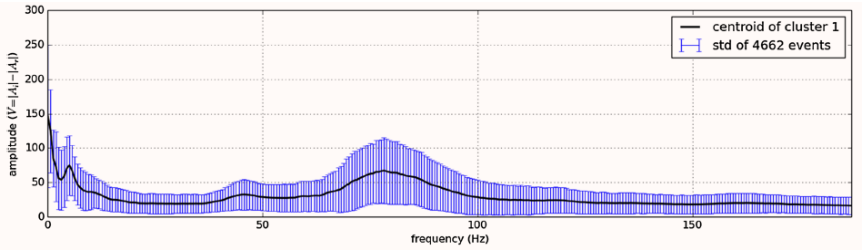

Figure 9: ...

In joint-based clustering a pairwise map of distances is calculated between each pair of joint representatives. A joint representative is calculated as the mean of the values of all event data for one joint, after the outlier removal phase. Two experiments were conducted. One experiment consisted only of 6 joints, including the damaged joint 4. The clustering method detected

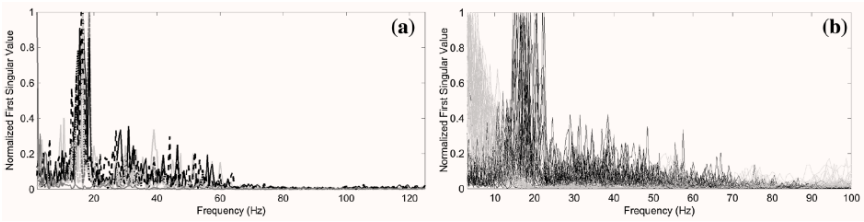

Figure 10: ...

the damaged joint as can be seen in 11. The second experiment was run on data from 71 joints. The resulting map can be seen in 12 which accurately detected the damaged joint 135. Damage was also detected in joint 131 but this result was not verified.

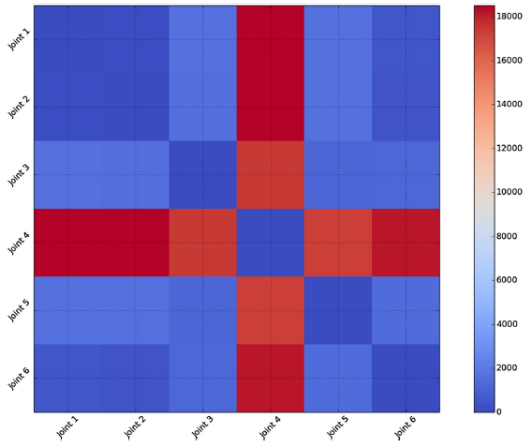

Figure 11: TODO:CAPTION

3.4.7 DSS

TODO: Overview of bridge DSS

3.4.8 Summary

TODO: conclude the literature review

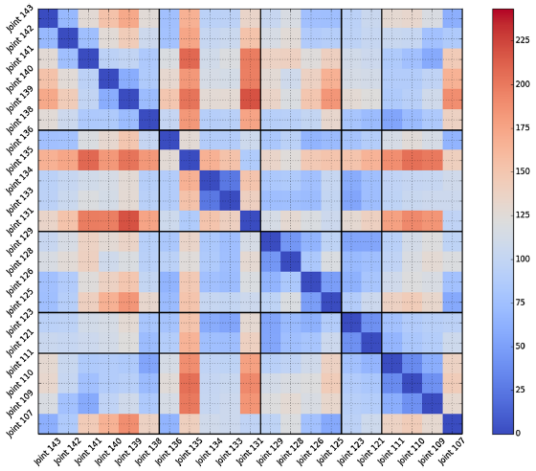

Figure 12: TODO:CAPTION

4 Methods

4.1 Data Collection

This section describes the data collection system which has been created to model a bridge in software and to collect data from simulating the bridge's response under a damage scenario and traffic scenario. Following a brief overview of how the data collection system operates, this section describes in detail the model of a bridge's geometry (a `Bridge`), of a damage scenario (a `BridgeScenario`) and a traffic scenario (a `TrafficScenario`), the FE software used to simulate a bridge's response, how the data collection system operates form input to output, a description of the collected data, validation of the model, and finally an overview of the assumptions that were made in modeling.

First a brief overview of the data collection system. A simulation scenario is defined as a combination of `BridgeScenario` and `TrafficScenario`. For a given simulation scenario and `Bridge`, a FEM is built according to the `Bridge` and `BridgeScenario`. A number of simulations of the FEM are then run, each time a concentrated load is placed at a different point on the bridge deck. The point is chosen so that it is on a track where a vehicle's wheels will be when the vehicles are later "driven" along the bridge. Vehicles are then sampled according to the given `TrafficScenario` and driven along the bridge on a traffic lane, the vehicle's wheels moving on the previously mentioned tracks. Using the principle of superposition, responses collected

from the previous simulations can be summed together, one for each vehicle's wheel, to calculate a response at a requested point, due to a vehicle or many vehicles on the bridge deck. This will all be explained more thoroughly in Subsection System Operation but it is useful to present a brief overview at the beginning of this section.

4.1.1 Bridge Modeling

META: How we model a static bridge

A high-level modeling language to describe no-prestress no-posttension concrete slab bridges was created for Python. A bridge is described at the top level by an instance of the class `Bridge`. An instance of this class (which we will from now on simply call a `Bridge`) can be transformed into a 2D or 3D FEM for OpenSees which can then be used to run FE simulations to collect responses.

A `Bridge` comes in two variants, 2D and 3D. Both can be represented by the same `Bridge` class. is described by a number of classes.

The model includes the types `Vehicle`, `Load`, `MovingLoad`,

{Lane, `Fix`, `Bridge`, `Section`, `Patch`, `Layer`, `Material`, `Response` and `ResponseType`. These types are used to model traffic, bridges and sensor responses.

The data collection system is parameterized by the traffic on the bridge and a specification of a bridge itself. A bridge is modeled by length, width, `Fix`es, `Lane`s, and `Patch`es and`Layer`s that are combined to form a `Section`. The `Fix`es are used to define fixed nodes of the FEM which represent a bridge's piers. The `Lane`s define where vehicles are "driven" along in simulation. A `Section` determines the cross-sectional area of the bridge in terms of `Patch`es (rectangular patches with a number of fibers) and `Layer`s (a number of fibers along a line). Listing **??** shows the programatic specification of bridge 705. Figure 13 shows the bridge that is modeled based on the specification.

```
bridge705 = Bridge {
    length :: 102
  , width  :: 33.2
  , lanes  :: [Lane(4, 12.4), Lane(20.8, 29.2)]
  , spans  :: [12.75, 15.3, 15.3, 15.3, 15.3, 15.3, 12.75]
  , cross  :: Section(...)
  }
```

A vehicle is represented by a number of variables.

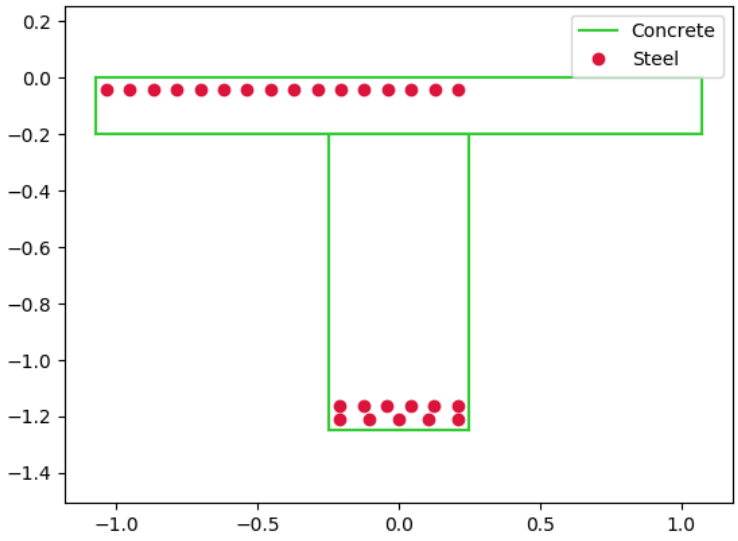

Figure 13: Cross section of bridge 705.

1. Discretization

    - Material properties may vary according to a continuous function on a real bridge while material properties in the FEM change at given discretization points.

4.1.2 Bridge Scenario

The goal of the damage identification model is to identify that damage in a number of selected damage scenarios. Damage scenarios can be classified as short-term or long-term. Short-term scenarios are defined as a change of the properties of structural materials and elements, and of the behaviour of the whole structure, due to effects that occur during a very short period of time. Long-term scenarios are time-dependent and may not only be related to external factors but also due to a change of state of materials with time. Tables 2 and 3 [8] outline some of the predominant types of damage due to short-term and long-term scenarios respectively.

TODO: Use table.el to fix tables

Of the damage scenarios listed in Tables 2 and 3, four scenarios are

Table 2: Types of damage due to short-term events.

Event	Examples/Consequences
Collision	Impact by overweight vehicle or boat in the river
Blast	Impact by vehicle followed by explosion
Fire	Impact by vehicle followed by explosion and fire
Prestress loss	Sudden failure of a prestress tendon
Abnormal loading conditions	Loading concentration and/or overloading in a specific site along the l
Excessive vibration	Earthquake
Impact	Impact pressure by water and debris during floods

Table 3: Types of damage due to long-term events.

Event	Examples/Consequences	Critical compo
Corrosion	Degradation of the bearings	Deck
	Loss of cross-section area in the prestressing tendons	Deck
Time-dependent properties of the structural materials	Excessive creep & shrinkage deformations	Deck
	Concrete deterioration	All
Low stress - high frequency fatigue	High frequency and magnitude of traffic loads	Deck
High stress - low frequency fatigue	Temperature induced cyclic loading	Abutment
Environmental effects	Freezing water leading to concrete expansion	All
Water infiltration/Leaking	Deterioration of the expansion joints; concrete degradation in the zone of the tendon anchorages	Deck
Pier settlement	Change in the soil properties	Deck

selected for identification by the DIM in addition to one unlisted damage scenario. These scenarios are chosen due to the practicality of simulating them in a FEM of bridge 705.

Pier settlement can be simulated by displacing a pier by a fixed amount, this is achieved in practice by applying an increasing vertical force known as a *displacement load* to the deck until the desired displacement is achieved.

Abnormal loading conditions can be simulated relatively easily by applying the heavy loads in the FE simulation. Care must be taken regarding the axle configuration because extreme heavy loads typically have a different axle configuration than less heavy vehicles.

Cracked concrete can be simulated by reducing the value of Young's modulus for the cracked concrete section. In practice, Young's modulus is often reduced to $\frac{1}{3}$ of its original value ([9]).

Corrosion of the reinforment bars can be simulated by increasing the size of the reinforcement bars TODO:WHY. Finally, a damage scenario is considered where it is not the bridge that is damaged but rather a sensor is malfunctioning.

A *malfunctioning sensor* can be simulated by adding a significant amount of noise to the simulated sensor responses or adding a constant offset to the responses TODO:LITERATURE. From discussions with Sousa TODO:REF, detecting malfunctioning sensors is useful to accomplish.

4.1.3 Vehicles

The data collection system is parameterized by the distribution of the vehicles that drive over it. The system has as parameter a filepath `vehicle_data_path`, a column name `vehicle_pdf_col`, and at `vehicle_pdf` a list that describes the probability density function (PDF) of vehicles in terms of the data in that column. The parameter `vehicle_data_path` must point to a `.csv` file which contains descriptions of vehicles. This `.csv` file will be loaded as a Pandas `DataFrame` and should contain data as described in Table 4.

Table 4: Example of Pandas `DataFrame` containing descriptions of vehicles that will be sampled. "axle$\setminus_{\text{load}}$" is the load per axle in kilo Newton, "load" is the sum of these values. "axle$\setminus_{\text{distance}}$" is the distance in meters between each pair of subsequent axles, "distance" is the sum of these values.

load	axle$\setminus_{\text{load}}$	distance	axle$\setminus_{\text{distance}}$
225.55	[79.44, 101, 45.11]	.79	[6.02, 1.32]
...	...	...	...

For example, a Pandas `DataFrame` will be loaded from `vehicle_data_path`, then vehicles will be sampled from this `DataFrame` based on the PDF. A vehicle that is sampled from this `DataFrame` will have a speed of 40kmph, and an axle-width of 2m, the inter-axle distances and the axle weights are taken from the `DataFrame`.

1. Traffic To train a classifier to distinguish between normal and abnormal traffic conditions it is necessary to define normal traffic conditions and additional traffic conditions.

   Traffic is simulated by

4.1.4 FE Program

Two FE programs are used for the collection of sensor responses, OpenSees ([10]) and DIANA ([11]). OpenSees is used because it is open source software, such that anyone can download and use the software without a licence. On the other hand is proprietary software, if you want to do research with Diana a licence must be purchased. The reason Diana is supported is because a verified 3D FEM of bridge 705 is available for Diana. In this thesis the Diana FEM is used in limited capacity for the verification of results obtained via OpenSees. The focus is instead on OpenSees because it is software that anyone with a laptop can use for free to extend this research. In addition it is useful to have two FE programs available, one (OpenSees) can be used to run less accurate but faster 2D FE simulations, allowing for a more rapid research cycle. The results can then be compared and verified against results from more accurate but also more computationally expensive 3D FE simulations (Diana). It is noted that the 2D model will ignore some aspects in the transverse direction of the bridge deck. For example the 3D model of bridge 705 has two lanes, but the 2D model ignores the concept of lanes entirely.

OpenSees stands for the *Open Sysem for Earthquake Engineering Simulation*, it is "an open source software framework for creating applications for the nonlinear analysis of structural and soil systems using either a standard FEM or an FE reliability analysis. It is object-oriented by design and—in addition to achieving computationally efficiency—it's designed to be flexible, extensible, and portable" [12].

DIANA (**DI**splacement **ANA**lyzer) is developed by DIANA FEA BV which is a spin-off company from the Computational Mechanics department of TNO Building and Conctruction Research Institute in Delft, The Netherlands. DIANA is a FE software package that is dedicated to problems in civil

engineering, including structural and geotechnical, and engineering related to tunnelling, earthquake, and oil and gas.

TODO: Image of the 705 Diana model.

4.1.5 System Operation

The goal of the data collection system is to translate a `Bridge`, along with a `TrafficScenario` and `BridgeScenario`, into a time series of responses.

A `Bridge` is transformed into a FEM for OpenSees, the resulting FEM is a 2D or 3D model depending on the dimensionality of the `Bridge`. In each case the FEM takes the form of a `.tcl` file (written in the TCL language). A `.tcl` file for Opensees consists of a sequence of commands for declaring a structure's geometry, material properties, and other settings of a FE simulation. For example, a `.tcl` file created from a `Bridge` will consist of a number of `node` and `element` commands, where nodes are points in space with degrees of freedom and elements are a mathematical relation of how degrees of freedom relate between nodes. In the case of the FEMs built from a `Bridge`, four nodes are connected by a *shell* element. Shell elements are used when the thickness is significantly smaller than the other dimensions. In the case of bridge 705's deck the length is 102.75m, width is 33.2m, and thickness is varying from 0.5m to 0.739m.

For each `BridgeScenario` for a `Bridge`, a number of simulations are run the first time that a response is requested to a point load or vehicle. For each wheel track a number of simulations are run. The number of simulations per wheel track is specified by the system parameter `Config.il_num_loads`. In each of these simulations a load of unit intensity `I` is placed at a point on the wheel track and responses of the bridge are recorded. The responses are translation from each node, and stress and strain from each element. Thus in summary, for each of the `Config.il_num_loads` points on one wheel track, the responses from the bridge are recorded. Each of these simulations we will call a unit load simulation, and the responses to such a simulation, unit load responses.

Unit load simulations are simulations that must only be run once, and then the principle of superposition can be used to determine the response to a vehicle, based on the responses recorded in the unit load simulations. Furthermore, the response to traffic (multiple vehicles on the bridge) can be calculated by simply summing the response to each individual vehicle on the bridge, as outlined in Listing 1.

There are only `Config.il_num_loads` number of unit load simulations per wheel track. And there are only a limited number of responses collected

```
response = 0
p = Point(x=35, y=0, z=25)
for wp, wi in vehicle:
    unit_load_simulation = load_sim_closest_to(wp)
    ru = unit_load_simulation.response_at(point)
    response += ru * (wi / ul)
```

Listing 1: Pseudo-code showing the use of the principle of superposition to determine the response to a vehicle from unit load simulations. When requesting the response at a point `p` to a vehicle on a bridge, the vehicle is first decomposed into loading positions `wp` and intensities `wi`, one position and one load intensity for each of the vehicle's wheels. Then for each wheel position `wp`, the unit load simulation is selected where `wp` is closest to the unit load applied in that simulation. From this unit load simulation, the response `ru` at the recorded point closest to point `p` is considered. Thus the response `ru` is the response one of the vehicle wheel's positions, except not to the vehicle's load but instead to a load of unit intensity, thus it must be multiplied by `wi / ul` where `ul` is the unit load intensity.

from each unit load simulation, as determined by the mesh density parameters. To clearly state an important point: the responses from unit load simulations which are used to calculate a response at a point `P` to a vehicle, are the responses at the recorded point closest to `P`, and the unit load simulations from which these responses are taken are those for which the unit load is closest to each of the vehicle's wheels position on the bridge. Thus the parameters which define the mesh density, and `Config.il_num_loads`, determine the discretization step and thus the accuracy of the responses which are calculated.

For each wheel track on the bridge, `Config.il_num_loads` amount of unit load simulations are generated. Then for any point on the bridge, the response at that point can be calculated to a load on one of the wheel tracks. The function of the response at a point to a changing load is called an influence line, commonly used in structural engineering to describe a response function. Figure 14 shows a number of influence lines of the displacement at different points on the wheel track at `z = -9.4m`, in each influence line plot a unit load is moved along the same wheel track.

Furthermore we can stack the influence lines for a number of points against each other, flipping each influence line by 90 so it is vertical. For example, we can consider a number of equidistant points along a slice in

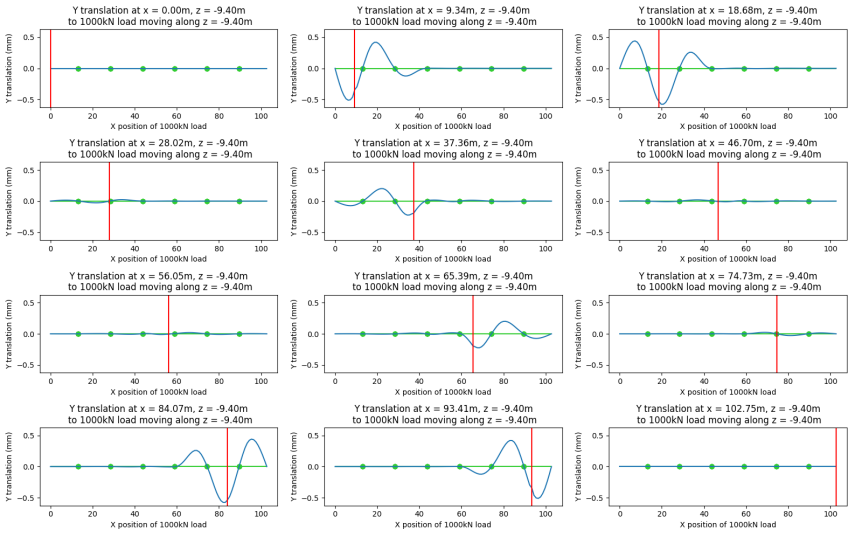

Figure 14: Displacement at different points on the wheel track at `z = -9.4m`, in each influence line plot a unit load is moved along the same wheel track. The red vertical line depicts the position of the load.

the longitudinal direction of a bridge, and for each of these points consider the response to a load moving along the same slice. Figure 15 shows such a matrix for `z = -9.4m`. Each column of the matrix is an influence line, each row shows the response along the bridge deck for `z = -9.4m` for a different loading position.

Another of the damage scenarios is pier displacement. To calculate responses to a load under this damage scenario, all of the unit load simulations need to be run again for this damage scenario. The name of the pier displacement damage scenario in the data collection sytem is `PierDisplacement`. `PierDisplacement` specifies a displacement in meters of one of a bridge's piers.

When creating a FEM of a `Bridge` under pier displacement for OpenSees, each of the bottom nodes of the piers under displacement are not fixed for y translation (to allow for the displacement of the piers to occur). An important step when creating a FEM under this damage scenario for OpenSees is to set the method of integration with the `integrator` command. Under the undamaged scenario the integrator used is `LoadControl`, which specifies that, among other things, the predictive time step of the simulation is driven by the loads applied. In the case of pier displacement the `DisplacementControl`

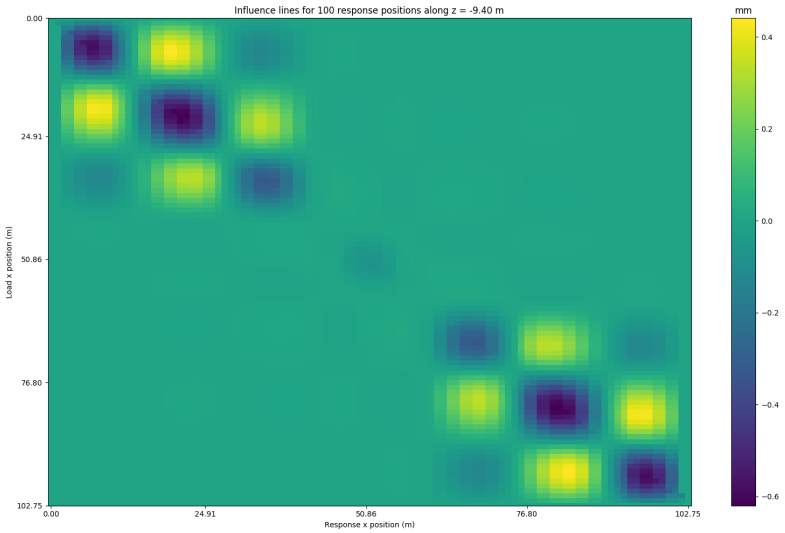

Figure 15: A number of vertical influence lines stacked together. Each influence line (column) shows displacement at a different point on the wheel track at `z = -9.4m`. Each column of the matrix is an influence line. Each row shows the response along the bridge deck for `z = -9.4m` for a different loading position. This image shows how, closer to the center of the bridge, the bridge does not suffer as much displacement.

integrator is used instead, this is used to specify that in an analysis step, the displacement control algorithm will seek the time step that will result in a specified increment for a particular degree of freedom of a specified node. For example the command `integrator DisplacementControl 1 2 0.1` specifies that the displacement control algorithm will seek an increment of 0.1 at node 1 in the second degree of freedom.

When running a pier displacement simulation the `DisplacementControl` command is used to specify that the central bottom node of the pier should be displaced by 1m. A load is placed on this node, though the load intensity is ignored by the `DisplacementControl` algorithm, the load intensity is instead increased until a displacement of 1m is reached. Figure 16 shows a contour plot of the displacement of the deck of bridge 705 due to a single pier being displaced by -1m.

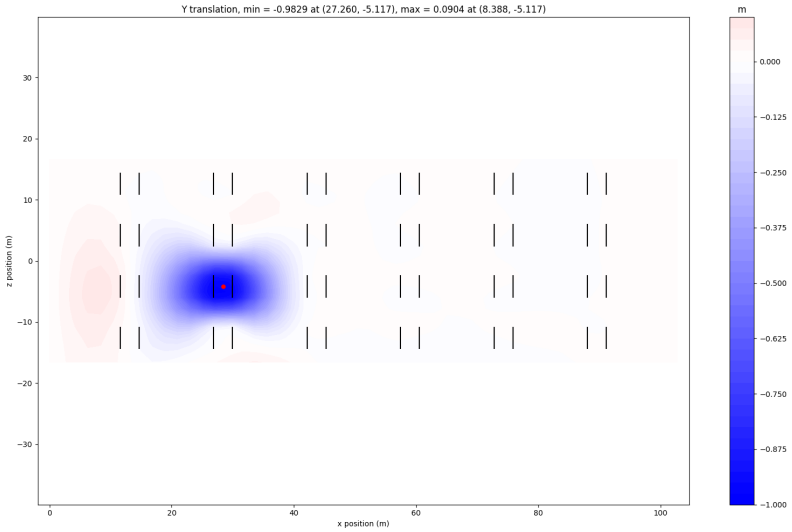

Figure 16: A contour plot of the displacement of the deck of bridge 705 due a pier being displaced by 1m. The node onto which a load is applied, and the same node that is watched by the `DisplacementControl` algorithm until the specified displacement of 1m is reached, is indicated by a red circle. This node is the central bottom node of the pier indicated by vertical black bars on either side of the red circle. The maximum displacement on the bridge deck is slightly less than 1m, this is because the piers are not infinitely stiff but have some elasticity.

Due to the linear elastic accumption made when modeling, only one pier

displacement simulation needs to be run per pier. One simulation is run for each pier, until that pier has been displaced by unit amount, one meter in the case of this data collection system. After these simulations have run, the response at any point on the bridge can be calculated due to any combination of piers being displaced by different amounts, as outlined in Listing 2.

```
response = 0
p = Point(x=35, y=0, z=25)
for vehicle in traffic:
    for wp, wi in vehicle:
        unit_load_simulation = load_sim_closest_to(wp)
        ru = unit_load_simulation.response_at(point)
        response += ru * (wi / ul)
```

Listing 2: Calculation of the response

4.1.6 Collected Data

The outputs of the system are time series of responses from sensors distributed across the bridge model, these time series of responses we call *events*. Events are labelled by simulation scenario and simulation time.

4.1.7 Model Assumptions

- All vehicles drive at the same speed.
- All vehicles drive along the center of a lane.
- All vehicles have the same axle-width.
- Vehicles arrive at a bridge according to a poisson process.
- measurements from the verified campaign only for a small subset (positions and existing healthy/damage state/)
- the model is linear elastic
- bridge scenarios
- The behaviour of a bridge captured in FE simulation is sufficiently close to the real behaviour of a real bridge that the analysis techniques explored on the simulated data can also work on real data.

This assumption is verified by (A) applying the analysis techniques explored on real data in addition to the simulated data and (B) verifying the collected responses against sensor measurements collected in real life.

Note that the accuracy of the responses depends on the discretization density of the FEM. This is a trade-off of time versus accuracy which can be chosen by the user. Discretization of the FEM is covered in Section 1. The accuracy of the FEM is shown to converge for bridge 705 in TODO: Convergence plot.

- The simulated noise that is applied to responses from FE simulation is sufficiently close to noise from sensors in real life that the analysis techniques explored on the simulated data can also work on real data.

  This assumption is verified by (A) applying the analysis techniques explored on real data in addition to the simulated data with varying levels of noise and (B) verifying the simulated noise is comparable to the noise from measurements collected in real life as shown in noise.

4.2 Anomally Detection

In this section the process of building the damage identification model is described. First there is an introduction to the damage scenarios that it is desirable for the model to identify, followed by a description of the setup for testing iterations of the model. After this an analysis is presented of the sensor responses with respect to the useful information in different sensor types for each damage scenario. Finally the damage identification model that is built is discussed.

References

[1] Anders Rytter. *Vibrational based inspection of civil engineering structures*. PhD thesis, Dept. of Building Technology and Structural Engineering, Aalborg University, 1993.

[2] Keith Worden and Graeme Manson. The application of machine learning to structural health monitoring. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1851):515–537, 2006.

[3] Mark Bedworth and Jane O'Brien. The omnibus model: a new model of data fusion? *IEEE Aerospace and Electronic Systems Magazine*, 15(4):30–36, 2000.

[4] B Lang, T Poppe, A Minin, I Mokhov, Y Kuperin, A Mekler, and I Liapakina. Neural clouds for monitoring of complex systems. *Optical Memory and Neural Networks*, 17(3):183–192, 2008.

[5] Alexander L Pyayt, DV Shevchenko, Alexey P Kozionov, Ilya I Mokhov, Bernhard Lang, Valeria V Krzhizhanovskaya, and Peter MA Sloot. Combining data-driven methods with finite element analysis for flood early warning systems. *Procedia Computer Science*, 51:2347–2356, 2015.

[6] Valeria V Krzhizhanovskaya, GS Shirshov, NB Melnikova, Robert G Belleman, FI Rusadi, BJ Broekhuijsen, BP Gouldby, J Lhomme, Bartosz Balis, Marian Bubak, et al. Flood early warning system: design, implementation and computational modules. *Procedia Computer Science*, 4:106–115, 2011.

[7] Alberto Diez, Nguyen Lu Dang Khoa, Mehrisadat Makki Alamdari, Yang Wang, Fang Chen, and Peter Runcie. A clustering approach for structural health monitoring on bridges. *Journal of Civil Structural Health Monitoring*, 6(3):429–445, 2016.

[8] Helder Sousa, Arpad Rozsas, Arthur Slobbe, Wim Courage, and Agnieszka Bigaj van Vliet. Development of a novel pro-active shm tool devoted to bridge maintenance based on damage identification by fe analysis and probabilistic methods. 2019.

[9] Yongzhen Li, JC Walraven, Ningxu Han, CR Braam, PCJ Hoogenboom, and Ir LJM Houben. Predicting of the stiffness of cracked reinforced concrete structures, 2010.

[10] Silvia Mazzoni, Frank McKenna, Michael H Scott, Gregory L Fenves, et al. Opensees command language manual. *Pacific Earthquake Engineering Research (PEER) Center*, 264, 2006.

[11] FEA DIANA. Diana 10.3 user's manual. 2019.

[12] Frank McKenna. Opensees: a framework for earthquake engineering simulation. *Computing in Science & Engineering*, 13(4):58–66, 2011.

,