REAL-TIME DISTRIBUTED CLOUD COMPUTING ARCHITECTURE FOR STRUCTURAL HEALTH MONITORING

Owen Searls[1], Zhiyong Zhao[2] Alva L. Couch, Ph.D.[3], and Masoud Sanayei, Ph.D., Life Member ASCE.[4]

[1] Undergraduate Student, Department of Computer Science, Tufts University, Medford, MA, USA, 02155, email: owen@searls.net
[2] Graduate Student, Department of Civil and Environmental Engineering, Tufts University, Medford, MA, USA, 02155, email: zhiyong.zhao@tufts.edu
[3] Associate Professor, Department of Computer Science, Tufts University, Medford, MA, USA, 02155, email: alva.couch@tufts.edu
[4] Professor, Department of Civil and Environmental Engineering, Tufts University, Medford, MA, USA, 02155, email: masoud.sanayei@tufts.edu; Corresponding Author

ABSTRACT

Real-time fatigue health monitoring has the potential to serve as a valuable complement to structural health monitoring (SHM) for bridge inspections. SHM is an objective supplement to visual bridge inspections and provides data on a bridge's fatigue condition between often infrequent visits from inspectors. Current methods of continuous structural health monitoring for condition assessment are performed by collecting measured bridge response subjected to operational traffic from an array of sensors installed on fatigue critical members of a bridge. The measured responses are used to determine the remaining fatigue life of the bridge.

The large amount of data involved in this process complicates the design of a system that will automate the data collection process at a bridge, analyze that data, and display information about bridge health to researchers and engineers. Variations in bridge designs and condition assessment algorithms also necessitate that such a system be modular and adaptable to allow for expansion to additional structures. A new system has been developed that separates bridge SHM from the data storage and communication system. This architecture creates a reliable interface for sending data from one or more bridges to a cloud server where it can be processed using modular algorithms that can be adapted for different use cases. The cloud-based web service and data repository makes bridge structural health data available to researchers at all steps of the process. This system provides significant advantages over previous platforms for structural health monitoring and condition assessment, most notably in the areas of modularity, extensibility, and reliability.

Keywords: Real-time computing, distributed architecture, REST API, data and system preservation, fatigue health monitoring, structural health monitoring, bridge signature, statistical hypothesis testing

INTRODUCTION

There are about 640,000 bridges in the US. About 40% of these bridges are over 50 years old, and 9.1% were judged to be structurally deficient by the American Society of Civil Engineers in 2017. The average age of a bridge in the US is about 43 years, meaning that in the coming years many of these bridges will reach the end of their intended lives. These bridges are essential pieces of infrastructure, with an average of 188 million trips across structurally deficient bridges occurring each day (ASCE, 2017). For steel bridges, cyclical loading by heavy trucks is a principal cause of fatigue damage and cracking.

Most state Departments of Transportations (DOTs) do not have sufficient funds for the upkeep of their bridges and require federal funds. Currently required 2-year bridge inspections are visual and subjective. We need tools and objective measurement-based methods for bridge owners for bridge management and decision making. In addition, steel bridges prone to fatigue cracks need assessment of their remaining fatigue lives, since they are subjected to cyclic heavy truck loadings.

Real-time structural health monitoring (SHM) is an objective supplement to visual bridge inspections and provides data on a bridge's condition between often infrequent visits by bridge inspectors. Current methods of continuous structural health monitoring for condition assessment are performed by collecting measured bridge response subject to operational traffic from an array of sensors installed on a bridge. The measured operational responses are used to assess the bridge condition and identify any potential damage.

In this research bridge condition assessment is implemented using statistical bridge signatures and probabilistic hypothesis testing. A ***bridge signature envelope*** prototype is generated using a bootstrapping method which gives the expected response of the bridge under live load, operational traffic events. This signature envelope represents a confidence interval for the response of a healthy bridge that can be used to monitor bridges for potential damage. A bridge is considered healthy if a sample of recent events falls within range of this signature envelope. ***Hypothesis testing*** based on a nonparametric rank sum test is used to determine the likelihood that a sample of recent events represents a significant divergence from the expected bridge response based on all prior events. This test generates a condition assessment of bridge health as a single value that can be used to generate alerts for bridge managers based on the severity of the bridge's divergence from the expected response.

The results of this condition assessment as well as subsequent predictions of remaining fatigue life of bridge components are displayed to researchers and engineers through a Fatigue Health Portal **(FHP).** The FHP can simultaneously monitor bridges owned by each state DOT or each district's bridge manager. The FHP can automatically send alerts if a bridge behavior exceeds the normal bounds of operational behavior by exceeding predetermined confidence intervals and can display more detailed bridge health visualizations that can be used by engineers for quick decision making. Using this measurement-based objective approach, if it is deemed to be deficient, the target bridge will be inspected earlier than the required 2-year period. More frequent inspections of targeted bridges for in-depth study can lead to timely bridge repairs, retrofits, or replacement. This early detection of structural damage due to fatigue is cost saving, by preventing further bridge damage and leading to a more reliable and safer highway system.

INNOVATIONS

The most significant innovations in the design of a real-time structural health monitoring system made during this research have to do with improving the computational, data, and network efficiency of the system, simplifying maintenance and expansion to new structures,

and increasing system reliability. The key innovations used in this research are: distributed architecture, extensibility to new structures, REST API, and data and system preservation.

Distributed Architecture. The components of the fatigue life monitoring system are distributed between computers housed at the bridges being monitored, in the cloud, and on the workstations of researchers and engineers who are monitoring a bridge. This significantly reduces the computing power and storage space required by the system housed at the bridge. The cloud part of the infrastructure further allows the workflow for a bridge to be broken up into two distributed phases, with preliminary data processing steps performed on the computer housed at the bridge, and the rest performed in the cloud. The total amount of data collected at each bridge is very large, but once usable traffic events are extracted, the amount of data that needs to be sent to the cloud is manageable, so this architecture allows for a significant reduction in the network bandwidth required to send bridge data to the cloud. All original data is still stored at the bridge as a backup, but only the selected data required to analyze a bridge's condition is sent to the cloud server.

Extensibility to new structures. Unlike the prior prototype (Zhao, 2017), the new system is designed to ease extending the monitoring to new types of bridge such as truss, girder, cable stayed, and suspension bridges as well as other structures such as roller coasters, mechanical systems, airplanes, and aerospace structures. The structural health assessment algorithms for each bridge are defined as a scientific workflow, which is a series of data manipulation and analysis steps sequenced to create a reproducible information pipeline. Once the algorithms for a new kind of bridge are written, setting up a workflow and initializing that structure in the web service requires minimal setup. While the system currently implements one analysis workflow for the Powder Mill Bridge, researchers can compare different algorithms by creating more workflows that run in parallel with the first. For example, this could be done to validate a small change to one workflow step, or to compare different decision-making algorithms such as one algorithm based on machine learning and another based upon statistical hypothesis testing. The data and scientific processes for each bridge are kept separate, so that workflows for new bridges can be added. These workflows can express differences in analysis algorithms based on structural attributes such as the number of girders or layout of sensors, as well as larger differences such as bridge materials and construction methods.

REST API. The interaction between the different components of the fatigue life monitoring system is made possible by an application programming interface (API) design according to the principals of ***representational state transfer*** (REST). This architectural style is the standard for complex web services and allows either computers at the bridge or end users of the system to access resources and data in the cloud, through a predefined set of stateless operations. The API allows these interactions to be standardized even when there are differences in data types between bridges.

Data and system preservation. The workflow processing system stores data and scientific programs in separate directories. These directories are synchronized with a backup cloud storage platform, so that a data loss incident affecting the cloud server does not lead to data loss. All of the data and programs associated with each new bridge are stored in a new directory, so that changes to a bridge workflow cannot affect the workflow of another bridge. Additionally, all bridge data is easily accessible through the web interface to the cloud storage platform and will remain accessible even while the system is being restored to address hardware malfunction.

POWDER MILL BRIDGE DATA ACQUISITION (DAQ) ARCHITECTURE

The data acquisition system at the Powder Mill Bridge in Barre, MA was installed by Dr. Sanayei and his team when the bridge was constructed in 2009 and instrumented by Geocomp Inc., Acton MA. This system includes 100 strain gauges, 69 temperature sensors, 16 uniaxial-accelerometers, 16 biaxial-tiltmeters, and 2 pressure plates. These sensors are connected to high speed data acquisition boxes, which in turn relay data to the bridge computer to be uploaded to the cloud (Saberi, M. R. et al. 2016). Figure 1 shows data acquisition boxes installed underneath the Powder Mill Bridge.

Figure 1. Above: Powder Mill Bridge in Barre, Massachusetts, USA; Below: Data acquisition system underneath the bridge by Geocomp

Fatigue Health Portal (FHP) Physical Architecture

Raw data from the DAQ system at a bridge is first relayed to an internet-connected computer located at the bridge. The Live Load Monitoring computer is represented by the leftmost blue column in Figure 2. The bridge computer converts that data to formats that can be easily processed by common software, and then can perform any number of initial processing steps from the overall workflow. Where to divide the workflow between the bridge and cloud computers is generally decided as a point where file sizes drop sharply, in this case immediately after single truck extraction. The portion of the system can also be easily replaced by a software simulation for testing, without affecting the operation of later stages of the system. This is the Simulated Structure described in Figure 2.

The cloud web service which supports the fatigue health portal is currently run in a no-cost environment hosted on an Amazon Web Services (AWS) EC2 server. The Backend Web Service is represented by the central orange box in Figure 2. AWS is currently the industry standard for cloud hosting and provides a high degree of freedom in deploying services and technologies. The current server is an EC2 t2.micro type instance, which is eligible for the AWS free tier and is currently sufficient for processing around 1000 events an hour. This server could be scaled up to a more powerful EC2 instance without major changes, and without large increases in cost, should the need arise (Amazon Elastic Compute Cloud (EC2) Documentation). This server runs Ubuntu 16.04 LTS (Long Term Support), guaranteeing that major upgrades will not be necessary for the next five years (Ubuntu Server Guide, 2016). This type of Linux server allows for flexibility in the type of research scripts that can be run and simplifies development and reuse of the web service software. Notably, this system can execute compiled MATLAB programs using the MATLAB Runtime Environment, which can be installed natively on Linux systems without the need for a MATLAB license (MATLAB Compiler Documentation).

All operational data and programs for each bridge are backed up to the cloud storage platform Box using a custom sync tool (Box API Overview). Box allows for almost unlimited data storage in a directory structure analogous to that already in use on the cloud server. Box is also already used extensively by many academic institutions, meaning that researchers will be automatically familiar with the interface, and the process of sharing research data with collaborators within and between universities will be simplified.

Fatigue Health Portal (FHP) Software Architecture

The central software component of the fatigue life monitoring system shown in Figure 2 is a web service that receives data from the computer located at the bridge, stores it, and invokes processing of this data into results that drive the web interface to the fatigue life portal. The web service both receives and serves data through its Representative State Transfer (REST) application programming interface (API) (Narumoto, M. et al., 2018). The web service triggers the workflow processing system when new events are uploaded and can also be instructed to run this processes at set intervals. This system contains the scientific code and algorithms that perform intermediate processing steps and draw conclusions about bridge health. Finally, the fatigue life portal is implemented as a separate web application that requests bridge data from the web service as needed and displays the bridge status to engineers or researchers.

There must always be at least one computer at each instrumented bridge or structure to record the measured data, extract the live load events, and upload the usable data to the cloud, but the number of cloud servers in the system could range between one for each bridge to only one server that handles the data for several bridges being monitored. The software

components of the system are designed so that some share of the data processing can be performed on the bridge computer and the rest in the cloud, configurable based on the available bandwidth and data storage capacity. The Fatigue Health Portal (depicted in green in Figure 2) is a separate software component that can be hosted on the same cloud server that handles bridge data, or on a separate server.

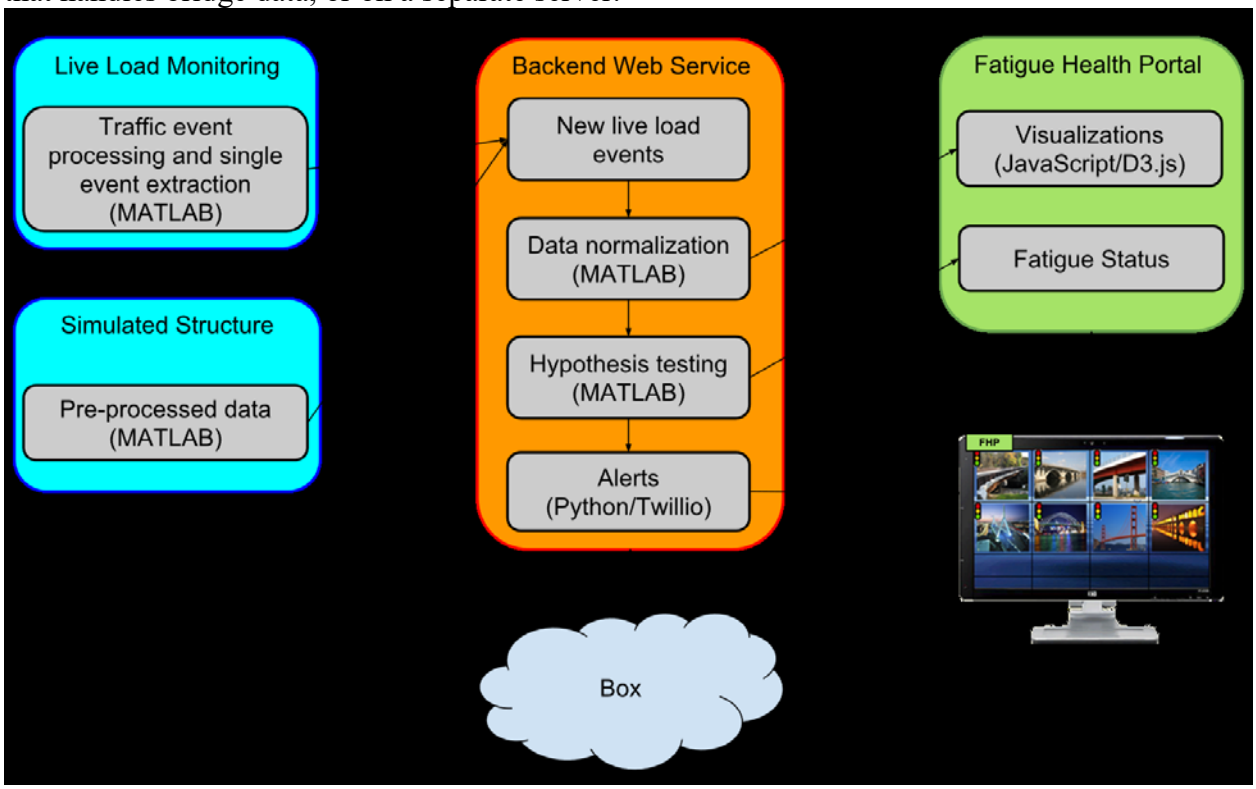

Figure 2. Fatigue Health Portal (FHP) system architecture

The cloud bridges web service is implemented in Python using the Flask web framework (Python 3.6.5 Documentation, Flask 1.0.2 Documentation). This includes the REST API, an indexing system for the list of supported bridges, and the code to trigger the workflow processing system when new events are added. This module interacts with the software running on the bridge computer and the fatigue life portal through the REST API, and interacts with the workflow processing system using a limited set of pre-specified directories.

The workflow processing system is implemented in Python and is capable of running any kind of scientific scripts that can be executed in a Linux environment. This system works by defining a set of processes along with the locations of their input and output data. Therefore, the structure of the data directory for a workflow can be defined in almost any consistent way when writing the rules of that workflow, although it will almost always be preferable to create a single directory containing all bridge data and a second directory containing every program in a workflow, which can be stored side by side in the workflow directory. Each process is also defined as having a set of dependent processes that must be run before that process. For example, the process that performs hypothesis testing currently specifies the process that convert strain data directly from the bridge to girder distribution factors as a dependency. The workflow processing system is independent of the cloud bridges web service and can be run without any knowledge of the web service. Thus, a workflow may contain more than two components and – in fact – can contain any number of components.

The FHP displays the condition of each bridge to researchers and engineers based on a statistical Bridge Signature and probabilistic Hypothesis Testing. The primary visual representation of the health of a bridge is the signature graph that is rendered for each girder.

This graph is generated using a statistical bootstrapping method to generate a 95% confidence interval for the expected response of a healthy girder. The current response of the bridge based on the last 50 events is then plotted on this signature graph (Follen et al., 2014). A bridge girder can be considered healthy if its current response is in range of the bridge signature. Figure 3 shows the signature of a girder with different types of simulated damage, while Figure 4 shows the healthy bridge signatures for the six girders at the Powder Mill Bridge.

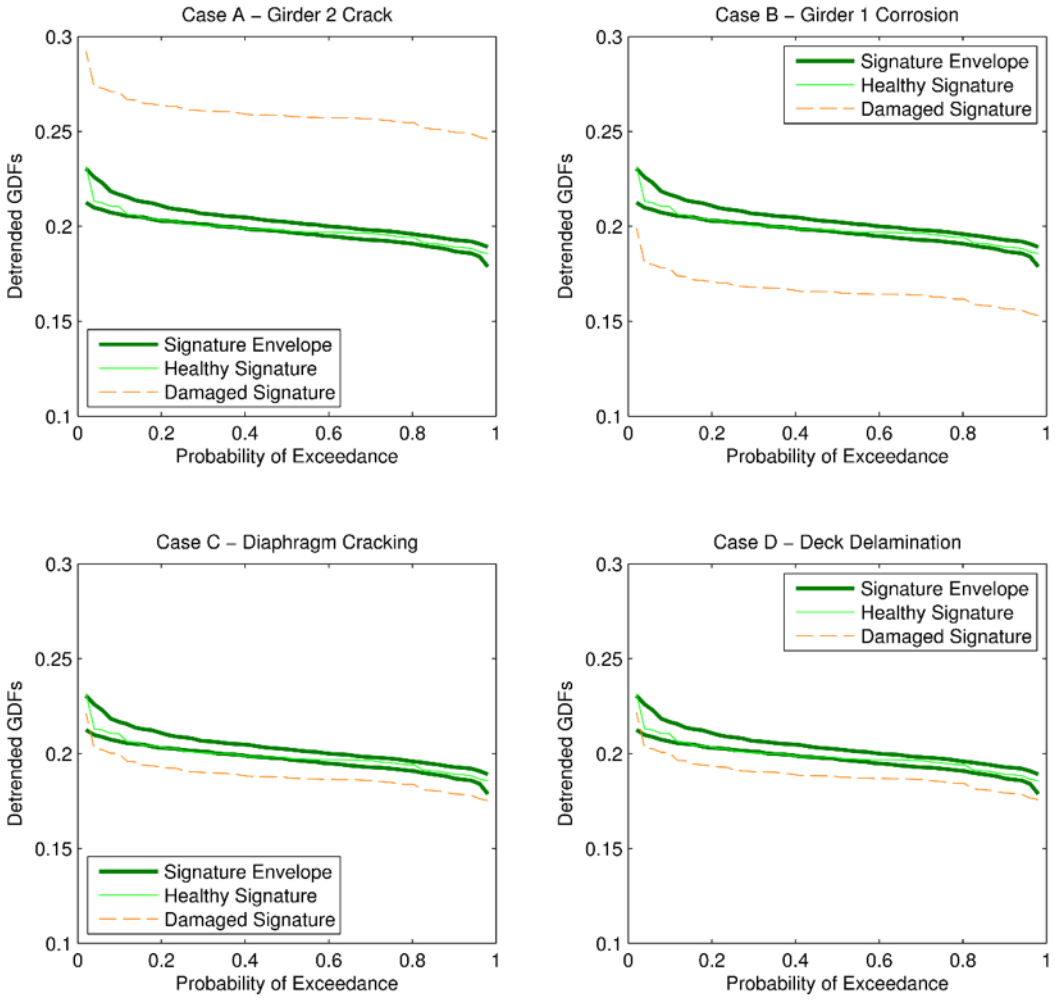

Figure 3. Bridge signature damage cases for girder one

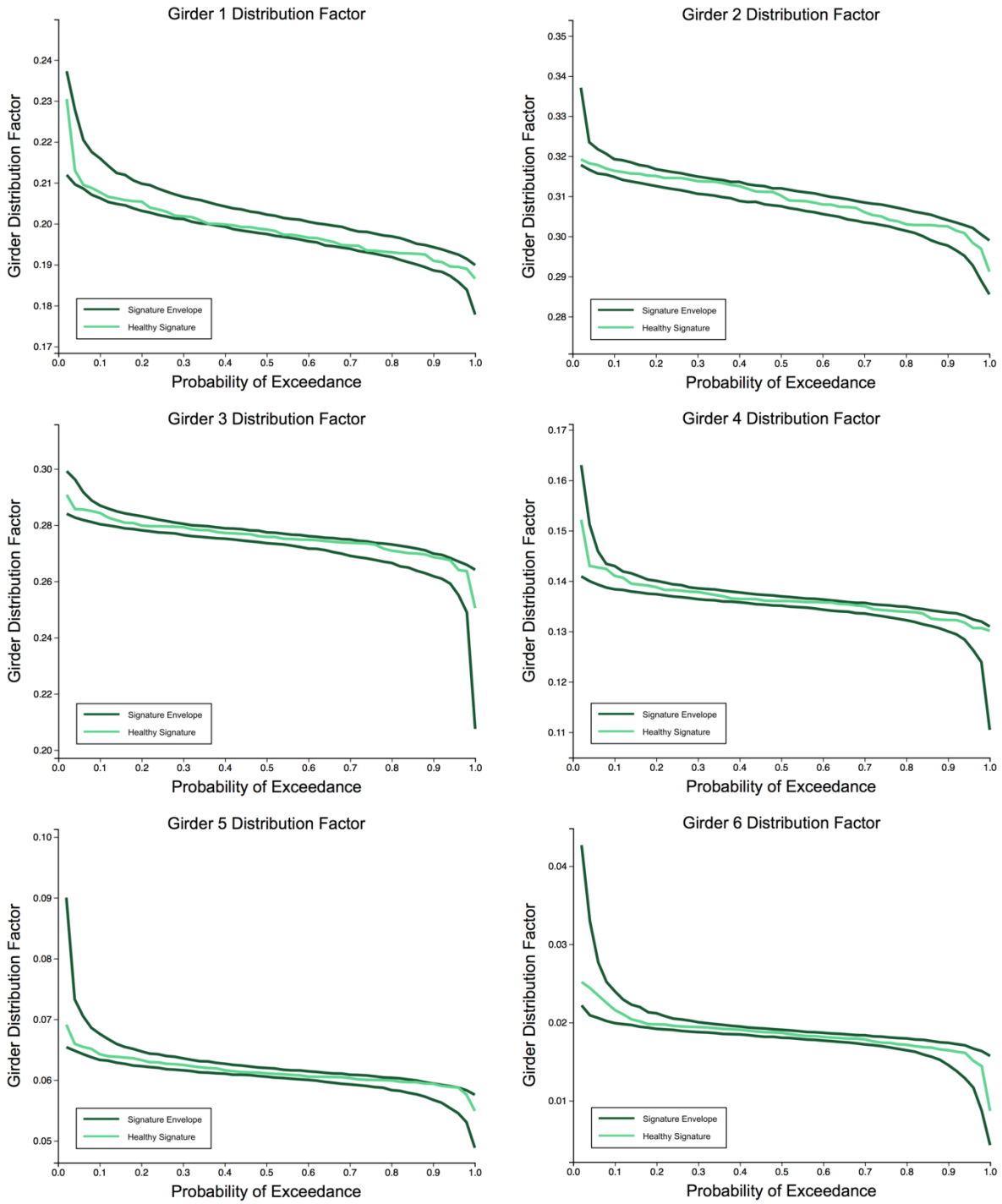

Figure 4. Healthy signatures for all girders at the Powder Mill Bridge

		Truth	
		H_0 Bridge is Not Damaged	**H_A** Bridge is Damaged
Decision	Conclude Bridge is Not Damaged	No Error $1 - \alpha$ (No Damage)	Type II Error β (Under-preparedness)
	Conclude Bridge is Damaged	Type I Error α (Over-preparedness)	No Error $1 - \beta$ (Damage)

Figure 5. Damage Detection Hypothesis Test Decision Matrix

A single measure of damage for a bridge is also calculated using probabilistic Hypothesis Testing to give an immediate view of bridge health to the bridge manager. The damage detection module implemented for the Powder Mill Bridge is based on a Wilcoxon rank-sum test, a widely used nonparametric hypothesis test which determines whether the two samples arise from the same distribution at a defined significance level. In the proposed method developed by Zhao (2017), the most recently collected 50 events, which is approximately the average number of usable single-vehicle events collected in a week at the PMB, are considered as a sample. All previously collected events are considered as the base set. The Wilcoxon rank-sum test is performed between detrended GDFs from the recent set of samples and the base set with a given significance level. These detrended GDFs are generated based on a multiple regression model which removes the effects of environmental factors such as temperature, frozen ground, and vehicle travel path from the data, leaving only leaving only the live load distribution due to the geometry of the bridge. Figure 5 shows the decision matrix for the damage detection hypothesis, which was originally developed by Reiff et. al (2016). As shown in Figure 5, there are two types of errors presented. Type I errors result from concluding a healthy bridge damaged, leading to over-preparedness. Type II errors result from concluding a damaged bridge healthy, leading to under-preparedness. The significance level also corresponds to the probability of a Type I error for individual girders. The outcome variable from the rank-sum test, h, is a binary variable equal to 0 or 1, with a value of unity implying that the recent set is different from the base set. These binary outcome variables of the rank-sum test are used to construct a bridge Damage Index (DI) as (4) (Reiff et. al, 2016).

The overall Type II error is defined as having all the girders with detrended GDFs having the same detrended GDF distributions as the base set for a damaged bridge. For the undamaged case, 18.9% of the trials have DI values larger than zero, which is a measure of the overall likelihood of a Type I error across all trials. For the damaged cases, 100% of the trials have DI values larger than zero, which implies that all the damaged cases are identified, thus the overall likelihood of a Type II error is zero for the 10,000 trial simulations considered. For all damaged cases, 100% of trials have DI values that are equal or larger than 3/6, which means at least three girders can be identified as damaged for all the scenarios considered. In Figure 7, the simulation trials are performed with significance level equal to 0.1%. In this simulation, lowering the significance level does reduce the capability to detect damage, which is illustrated by comparing the results in Figure 6 and Figure 7 with high DI values. Nevertheless, the overall Type II error is still zero for the 10,000 trials considered and the overall Type I error for this case dropped significantly. From the simulated experiments, the proposed method illustrates that the proposed approach, which is based on detecting

damage using changes in the detrended GDFs, was found to be very effective for detection of damage and simultaneously led to a very low probability of false damage alarms. The DI values are used to generate a single bridge condition represented by a color that is presented to the bridge manager. These are the same colors used in Figure 6 and Figure 7 so the condition will be green if zero or one girders are damaged, yellow if two or three girders are damaged, and red if more than half of the girders are damaged.

The FHP website is implemented as a static web page using AJAX to dynamically load bridge data and the JavaScript visualization library D3 to render bridge health visualizations. The portal always displays the most recent data and is a separate software component from the analysis modules, so changes may be made to bridge visualizations independently of the rest of the system. Additionally, it is possible to create multiples versions of the portal using the same data source, either to accommodate different audiences or experiment with new kinds of visualizations.

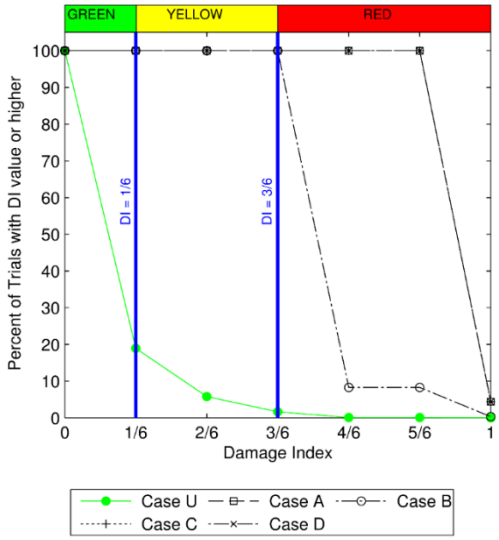

Figure 6. Damage Detection Rate from 10000 Trials with α=5%

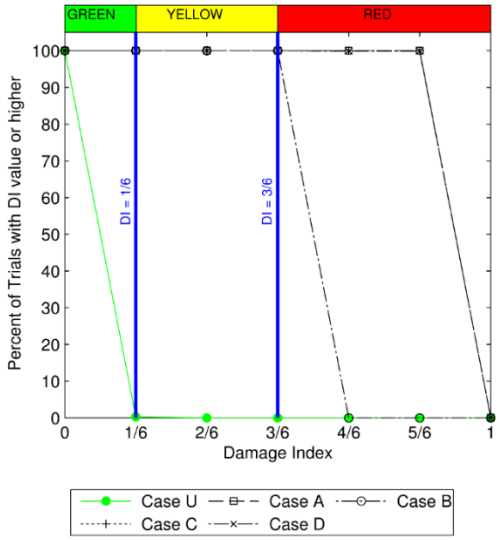

Figure 7. Damage Detection Rate from 10000 Trials with α=1%

CONCLUSIONS

The fatigue health portal is a powerful tool for managing our ageing steel bridges subjected to operational loading due to daily traffic. Bridge owners, engineers, and researchers have real-time access to structural fatigue and health data for monitoring and management. The cloud-based architecture of the fatigue life portal is efficient both in its use of computing resources and the labor required by bridge engineers and researchers. This system eliminates the need for manual data transfers and processing and provides engineers “real-time” protocol for access to bridge data and bridge condition assessments. The modular design of the Fatigue Health Portal (FHP) makes it easy to expand to new bridges. Website code and scientific code are kept entirely separate so that bridge engineers can make changes without knowing website code. This separation also simplifies maintenance because improvements can be made to the website without influencing the scientific workflow. Storing bridge data in Box provides a reliable cloud backup with a very simple restoration procedure. Additionally, most researchers in academic settings are already familiar with Box, so the Box web interface provides a valuable secondary interface for accessing and sharing bridge data.

FUTURE WORK

In the immediate future, new bridges and structures could be added to "remaining fatigue life" monitoring using FHP. Data collection is already ongoing at the Memorial Bridge in Portsmouth, NH, and could be integrated into the existing system using the REST API. Each new bridge added to the system will require a unique workflow, with the degree of difference between that workflow and the existing workflow for the Powder Mill Bridge depending largely on the degree of difference between the structure and instrumentation of those bridges. The fatigue life portal also allows for easy experimentation between different damage detection algorithms. Techniques such as artificial neural networks have been shown to be valuable in assessing the condition of steel bridges (Weinstein, 2018), and could be implemented as an alternate workflow for the Powder Mill Bridge by substituting only one workflow step. The FHP can be used for any structural system subjected to cyclic loading such as: wind turbines, transmission towers for power lines, airplanes, rotating machinery, petrochemical plants, nuclear power plants, and automobiles. The market for FHP is not limited to the U.S., as steel fatigue is a systemic problem in most industrial and developing countries.

REFERENCES

Amazon, Inc. (n.d.). Amazon Elastic Compute Cloud (EC2) Documentation. <https://aws.amazon.com/documentation/ec2/> (May 9, 2018).

ASCE (American Society of Civil Engineers). (2017). 2017 Infrastructure Report Card. <https://www.infrastructurereportcard.org/wp-content/uploads/2017/01/Bridges-Final.pdf> (Aug. 23, 2018).

Box API Overview. (n.d.). <https://developer.box.com/v2.0/reference> (May 9, 2018).

Christopher W. Follen, Masoud Sanayei, Brian R. Brenner, Richard M. Vogel (2014) "Statistical Bridge Signatures," Journal of Bridge Engineering, January 2014. 10.1061/(ASCE)BE.1943-5592.0000596.

MathWorks, Inc. (n.d.). MATLAB Compiler Documentation. <https://www.mathworks.com/help/compiler/> (May 9, 2018).

Narumoto, M. et al. (2018). API Design Guidance. <https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design> (May 9, 2018).

Pallets. (n.d.). Flask 1.0.2 Documentation. http://flask.pocoo.org/docs/1.0/ (May 9, 2018)

Python Software Foundation. (n.d.). https://docs.python.org/3/ (May 9, 2018).

Alexandra J. Reiff, Masoud Sanayei, and Richard M. Vogel, (2016) "Statistical Bridge Damage Detection using Girder Distribution Factors," Engineering Structures, Vol. 109, 15 February 2016, Pages 139-151, 10.1061/(ASCE)BE.1943-5592.0000596.

Mohammad Reza Saberi, Alireza Rahai, Masoud Sanayei, and Richard M. Vogel, (2016) "Bridge Fatigue Service Life Estimation Using Operational Strain Measurements," Journal of Bridge Engineering, January 2016, Paper No. 04016005, 10.1061/(ASCE)BE.1943-5592.0000860.

Ubuntu Documentation Project. Ubuntu Server Guide. (2016).

Jordan C. Weinstein G, Masoud Sanayei, and Brian R. Brenner (2018), "Bridge Damage Identification Using Artificial Neural Networks," Journal of Bridge Engineering, Vol. 23 (11), November 2018, 10.1061/(ASCE)BE.1943-5592.0001302.

Zhao, Z. (2017). "Cloud-based Real-time Continuous Bridge Monitoring: Bridge Weigh in Motion and Condition Assessment" Master's Thesis No. 1872375351, Tufts University.