A Platform for Mobile Visualization of SHM Data

Matthew Woelk[1] and Dean K. McNeill[2]

ABSTRACT: By their very nature, structural health monitoring systems collect and store large quantities of measurement data gathered from a wide array of sensors. This is particularly true for installations that employ continuous monitoring, which can easily amass gigabytes of data each day. For this raw data to be meaningful it must be processed to extract salient detail, and presented in a form that clearly shows the underlaying features that are relevant and meaningful for assessment of the structure's condition or response. Data acquisition systems used to gather the SHM measurements do not generally provide effective visualization tools for presenting the data to engineers in an easily accessible way. Providing this capability, particularly to engineers in the field, will enhance the ability of those engineers to observe and assess a structure's performance, even while on-site.

To address this need, we have developed visualization tools that provide efficient access to SHM data on a variety of mobile devices including Android and iOS based smartphones and tablets. In addition, this same functionality is made available through a browser based interface for access in traditional desktop environments.

This article describes the details of the mobile visualization system's implementation and the supporting back-end server infrastructure employed. A major objective of the design approach undertaken is the delivery of a platform that utilizes open-source tools and server technologies. As such, the resulting system may be deployed on a range of servers and mobile devices, and is therefore not tied to a particular manufacturer or device family. This makes the solution highly portable and suitable for a wide variety of infrastructure monitoring applications.

KEYWORDS: Data visualization, SHM.

1. INTRODUCTION

This paper describes a system for mobile data visualization specifically targeting structural health monitoring (SHM) systems. Current methods of analysing SHM data often involve rendering static images of portions of the data as a time-series, where each raw data point is shown. For charts where the time between samples is such that one pixel on the screen, or point on a plot, must represent several successive measurements, the resulting image/figure will appear as a blur of lines from which only the minimum and maximum of a section will be extractable. An example of one such image is Figure 1. In addition, rendering large data sets requires that each point be gathered from a database and rendered onscreen. The communications bandwidth required to perform this operation scales linearly with the number of data points being accessed. This means that looking at large ranges of data is both inconvenient and uninformative.

To address these problems, we have developed a system in which the data is compressed into a simpler and more concise visual representation, which allows the user to view the data in a multi-scale manner. The proposed method retains important metrics of the data, while compressing it enough to be suitable for streaming to a mobile device over the Internet.

Simplified plots are composed using a binning algorithm which generates a variation of the statistical five number summary as described in [1]. These statistics are initially computed using the raw data for displays at the finest resolution. As the users view expands to encompass longer time periods, statistics from the lower level bins are combined to maintain a meaningful display of the underlaying information. As a result, charts with the same horizontal pixel space requires the transfer of the same amount of data

[1]University of Manitoba, Winnipeg, Canada
[2]University of Manitona, Winnipeg, Canada

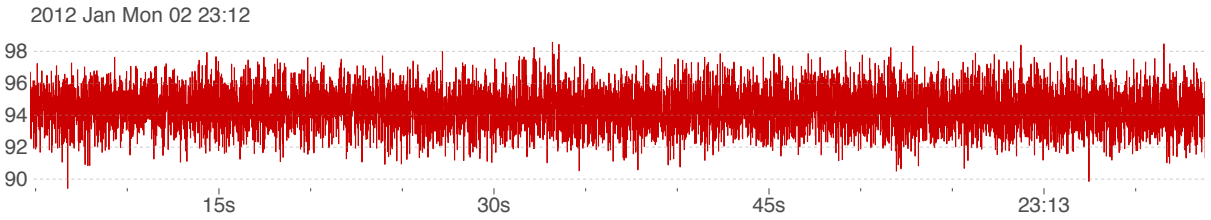

Figure 1. Graph showing 1 minute of raw data.

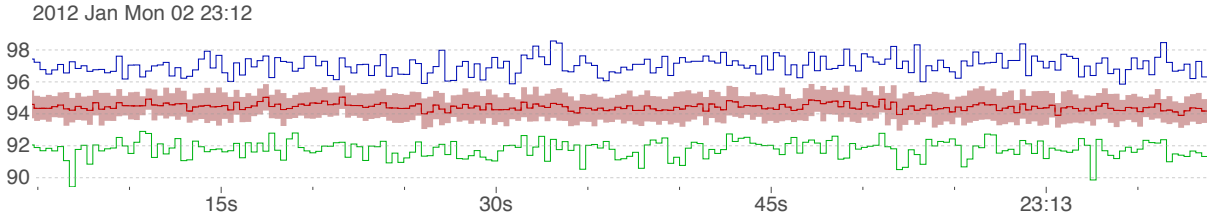

Figure 2. Graph showing 1 minute of binned data.

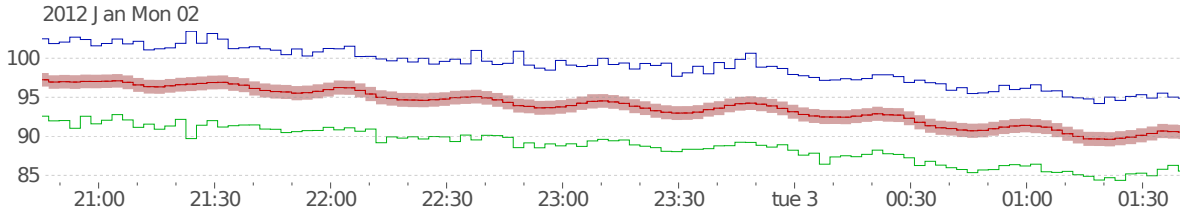

Figure 3. Basic graph showing 3 hours of data

from the server, no matter what range of times are being displayed. An example of the display is given in Figure 2, with a wider view shown in Figure 3. In addition, the chart is interactive, allowing the user to explore the data at arbitrary scales and ranges by dragging and zooming the chart. This chart is both more convenient and more informative than the typical raw data view of Figure 1.

2. BINNING ALGORITHM

To deal with the display and communication of the large quantities of data involved in SHM, a binning algorithm was designed to extract useful properties of the data at different levels of resolution. At the lowest level (resolution), pairs of successive data points are grouped together, and then the following properties for each group are found: the maximum value, the minimum value, the average value, the first quartile, and the third quartile. The resulting tuple of five values describes the data at the highest resolution and is referred to as bin level one. Higher levels are calculated in the similar manner. First, successive pairs of tuples from the next lower bin level are grouped, and then the five properties are calculated for each of those groups to make the next level of bins.

The five properties forming each tuple were chosen for two reasons. The first is that they are a standard set of statistical properties with which many people are familiar. These five properties are equivalent to the Five-number summary as described in [1], with the exception of the median being replaced by the average, and the quartiles being an approximation. The second is that the statistics of any level can be calculated from those of the next lower level. For example, data from level 14 is all that is needed to calculate level 15.

The five properties are calculated in the following ways, where subscripts are used to denote which of the two bins of the lower level a statistic relates. For example $average_1$ is the average of the first bin, while $average_2$ is that of the second.

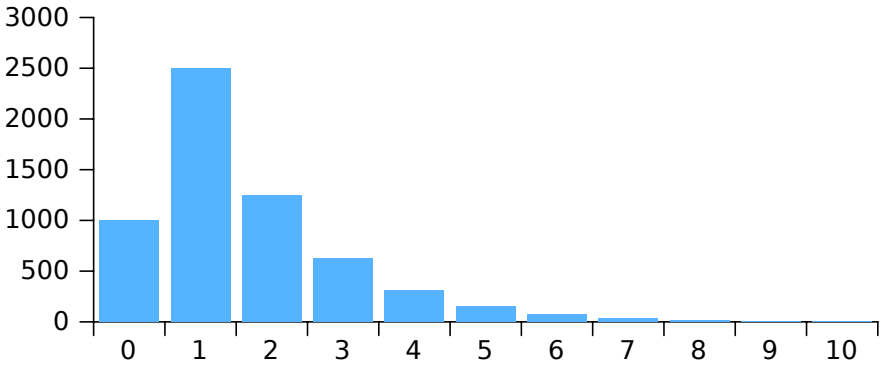

Figure 4. Storage requirement per level, where raw data is level 0.

The average for a bin is found by taking the average of the averages of the two bins. The equation is

$$average = \frac{average_1 + average_2}{2} \tag{1}$$

The maximum value is found using

$$maximum = max(maximum_1, maximum_2) \tag{2}$$

where $max()$ is a function which returns the highest of two values, and the minimum value is found using

$$minimum = min(minimum_1, minimum_2) \tag{3}$$

where $min()$ is a function which returns the lowest of two values.

The first quartile, $q1$, is calculated using

$$q1 = avg(mintwo(q1_1, q3_1, q1_2, q3_2)) \tag{4}$$

where $avg()$ is the average function used in equation 1 to calculate *average*, and $mintwo()$ is an equation which returns the two smallest of four values.

Similarly, the third quartile, $q3$, is calculated using

$$q3 = avg(maxtwo(q1_1, q3_1, q1_2, q3_2)) \tag{5}$$

where $avg()$ is the average function used in equation 1 to calculate *average*, and $maxtwo()$ is an equation which returns the two largest of four values.

Each level of generated binned data contains half of the information of the previous level for the same period of time, with the exception of level one which contains two and a half times the number of data points as the raw data. The relationship between level and amount of data needing to be stored for the same time period is shown in Figure 4, and follows the relationship

$$storage = \frac{5n}{2^{level}} \tag{6}$$

where n is the number of raw data points.

3. DISPLAY OF BINNED DATA

A chart was devised to display these levels, made up of three line charts and an area chart. Figure 3 shows an example of the chart, with the colours representing the following properties: blue for the maximum

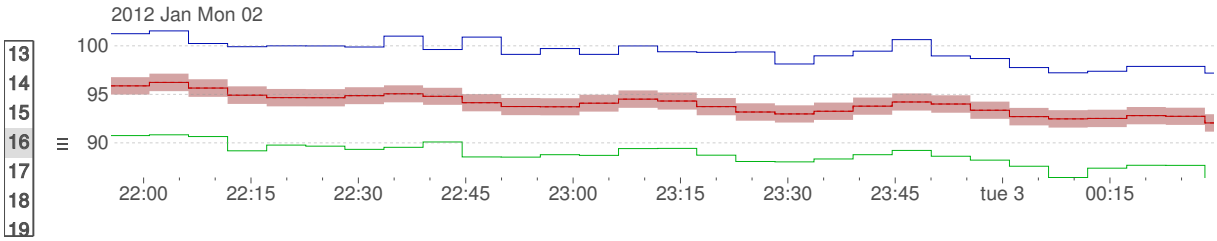

Figure 5. Handle in position to show level 16.

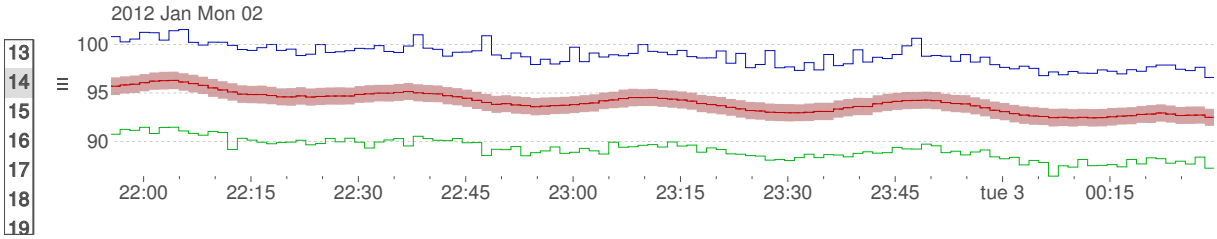

Figure 6. Handle in position to show level 14.

value (upper line), green for the minimum value (lower line), red for the average (middle line), and a light-red area or band for the region between the 1st and 3rd quartiles.

A user is able to zoom in and out of the plot, and as they do the current bin level being displayed will change automatically. This is done so that the size of an on-screen bin stays within a certain range of sizes. The user is able to change this range using the handle on the slider to the left of the chart, as can be see in Figure 5, with Figure 6 showing the result of changing from level 16 to level 14 for the same time period originally depicted in Figure 5.

The y-axis of the plot shows strain gauge readings in terms of microstrains, while the x-axis combined with the label above that chart show the time when the sample was taken. Both the x-axis and the upper label change according to the current zoom level of the plot. The upper label displays the context necessary for the user to determine the current data range being shown, and corresponds to the left-most edge of the plot. The label will show the least amount of information necessary for the user to be guaranteed, at the current zoom level, to have enough information to understand the context of the data in time. A period of time (year, month, day, etc.) is shown in the upper label if and only if at most one of those periods are guaranteed to be visible in the x-axis labels at any one time. This can be seen in Figures 7 and 8.

Simply by zooming and panning the chart, a user is able to view and visually analyse data from levels zero (the raw data itself) to level 27 without waiting for data to be rendered.

The *underscore.js*[2] javascript library is used to organize and manipulate the data, and the *d*3.*js*[3] javascript library is used to map the manipulated data to scalable vector graphics (SVG) elements in order to render the chart in a web browser. HTML and javascript are used to layout the chart, the slider and their elements, and CSS is used for colouring.

4. SYSTEM ARCHITECTURE

The entire system consists of three physical components. They are the database server, the visualization server, and the client device. The database server stores the original raw data which is never modified. The visualization server requests raw data from the database server and generates the binned data representation. It serves both binned and raw data as needed in response to the user's request, which the client device then renders and displays. Figure 9 shows the system architecture and the flow of information through it.

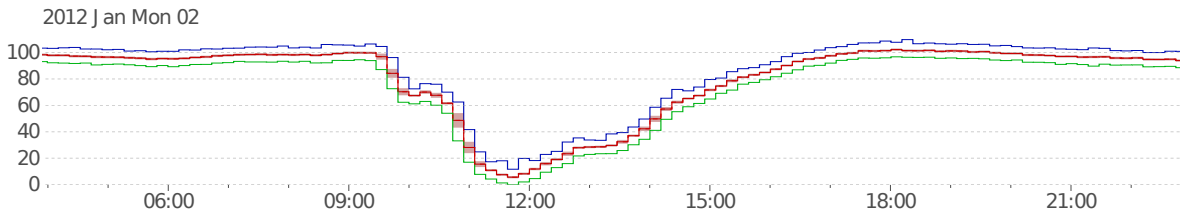

Figure 7. Time context label showing day of the month.

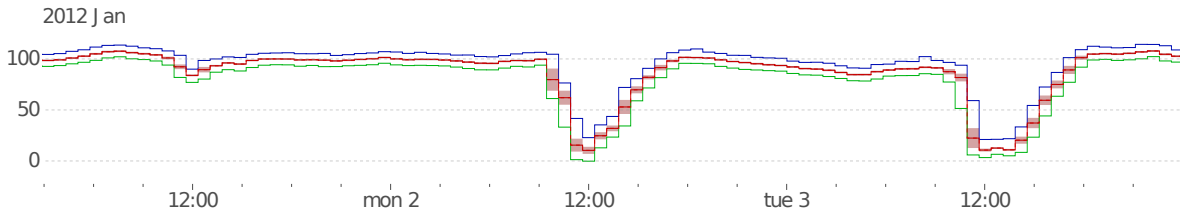

Figure 8. Time context label showing month of the year.

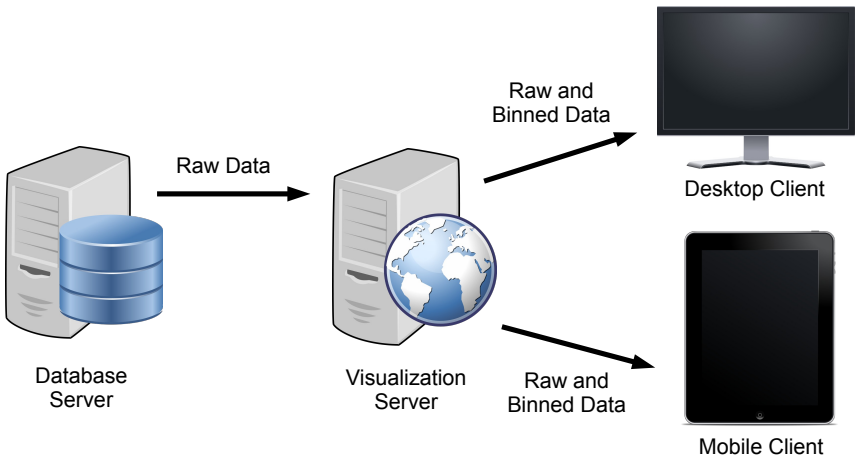

Figure 9. System Architecture

4.1 DATABASE SERVER

The database server stores all of the raw measurement data obtained by the data acquisition equipment situated on the bridge structure, and archives it in a MySQL database. The visualization server communicates with the database server across an encrypted connection to obtain data as needed. While the database server would be capable of performing the additional functions of the visualization server, the two have been deliberately separated to enhance security of the data by ensuring that the visualization system is incapable of modifying information in the original database. The database used to test this system contains nine months of data acquired at a sampling rate of 200Hz. This is 4.7×10^9 samples for each strain gauge present in the structure.

4.2 VISUALIZATION SERVER

The visualization server scans and bins the data it obtains from the database server and passes on both raw and binned data to the client as required. To bin the data, the server first requests a portion of raw data from the database server and processes it as described previously in Section 2. Then the server stores that data in its own local *couchdb* database.

Both the *couchdb* database and the client store bins in *containers* consisting of 32 bins each. This means that the server, accesses 1 to 3 containers per request from the client. This significantly reduces overhead by decreasing the number of requests by the client, and results in fewer keys in the local database. This in turn results in a smaller database, but at the cost of delivering more than the required amount of data to the client than is actually being displayed to the user at that time. However, having this additional data available on the device allows for faster and more responsive updates of the display as the user adjusts the resolution and time period being viewed.

In order to save space in the visualization server's local database, the lowest six bin levels are not stored. Should the client request bins of levels zero through five, the server accesses the raw data from the database server and sends it to the client, where the requested level is generated and rendered. Tansmission of the raw data requires a maximum of 6.4 times the number of bytes to be sent in comparision to pre-generated data. Each lower level kept by the server increases its local database size by 50%, while every level left out means twice the amount of raw data must be sent from the database when it is needed. Choosing to not store levels zero through five means that our total local database contains at most $7.4 * 10^8$ pieces of data, which is 16% of the amount of the raw data points. It was found with our particular setup that this struck a good balance between quick response time for raw data and storage space efficiency.

4.3 CLIENT

The client is any device with an internet connection and an up-to-date web browser. This includes the full range of devices from smartphones to tablets to desktop computers. Also, as a result of using web technologies for the client software, the client should run on all major mobile and desktop operating systems. The system has been tested on Windows, OSX, Linux, iOS, and Android. This ensures the maximum flexibility on the part of the end user to access the visualization tools from whichever device and location is most convenient at the present time.

4.4 SYSTEM COMPONENTS

Since a key requirement of the system was the use of open source tools, the system components do not rely on hardware or software from any one proprietary vendor. The system implemented here uses an Apple XServe with MySQL as the database server, together with an Apple MacMini Server as the visualization server. Clients interact with the visualization server using a standard web interface provided by the Apache web server software. A functionally identical system could be setup using hardware from other vendors and deploying these same or similar software tools on those systems.

5. CONCLUSION

This paper has described a system for displaying SHM sensor data on mobile devices. A server generates binned representations of raw data obtained from a database and serves it to client devices in a compressed format. This system has shown to be capable of displaying nine months of data from a strain gauge in a compressed way which allows for convenient and informative visual analysis.

While the system has been setup to initially display basic time-series plots of strain, other information sources may be easily added to extend the system.

6. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the financial support of the Province of Manitoba and the ISIS Resource Centre in conducting this work.

7. REFERENCES

[1] David C. Hoaglin, Frederick Mosteller and John W. Tukey. ”Understanding robust and exploratory data analysis”. *Wiley*, 1983. ISBN 0-471-09777-2.

[2] Document Cloud, Underscorejs Library, http://underscorejs.org

[3] Michael Bostock, Data Driven Documents, http://d3js.org