A SENSOR NETWORK DATA STRUCTURE FOR CENTRALIZED POST-PROCESSING OF EVENTS AND TRENDS

Joost Kuckartz[*,1] and Philip Collier[1,2]

ABSTRACT:

Structural health monitoring systems contain many different types of sensors, each often measuring at a different frequency. For a complete overview of the monitored structure, data is stored and processed in a centralized computer. The challenge of structural health monitoring systems is the management and analysis of the large amount of data for short-term event detection and long-term trends. Additionally, not only raw measurements but calculated values are required to be stored and analysed.

Proposed is a data structure consisting of an indexing system and file system. The indexing system first divides at the sensor level and then on time. The sensor level division can be any spatial or key tree structure (such as the kd-tree), the time division refers to the data files of the sensors and calculations. The time division stores raw data in its lowest level while storing average values (and other calculated values such as maximum, minimum and standard deviation) in the higher levels of the tree, allowing for rapid trend analysis and faster data searching (i.e. thresholds of events).

The new data structure is designed to deal with the nature of sensor measurements (stream data to files) and is flexible in the choice of levels. In its design for a specific application, choices for calculations and time division are made representing the current and future analysis to be performed.

Due to the flexibility of the data structure and the nature of sensor measurements, several data reduction techniques, such as omission of raw data when no events occur, are discussed.

KEYWORDS:

Structural health monitoring, sensor networks, data management, data analysis, data structure, trend detection, event detection, TB-tree

1. INTRODUCTION

The goal of structural health monitoring (SHM) is to provide a comprehensive picture of the monitored structure with the available sensor data. This requires processing which however cannot always be performed on a single data point, therefore requiring data storage and post-measurement data access.

Even though distributed sensor networks can handle certain data storage and processing at the sensor location, eventually this data is transmitted and gathered into a centralized location. The reason SHM, and many other sensor networks, cannot be decentralized is the requirement for post-processing and data comparison [1-3]. Although a lot of research effort is going into the reduction of data transmission by implementing data processing algorithms at distributed nodes, post-processing often requires data from multiple sensors and from longer time periods than the distributed sensors can handle.

[*] Corresponding author, email: j.kuckartz@student.unimelb.edu.au
[1] Department of Infrastructure Engineering, The University of Melbourne, Australia.
[2] Cooperative Research Center for Spatial Information, Melbourne, Australia

At the central computer, data from all sensors come together, and for SHM this constitutes data from a range of sensor types, each with its own measurement frequency and different data point description. This results in different storage requirements, often also depending on how sensors already transmit data. Some sensors stream data directly to a database, others output to file.

Processing however requires all data to be combined into a single software package and due to the lack of a combinatory data structure being available, each SHM system has its own developed program [3-5]. There are however efforts to create a combinatory system [6,7] but limitations still exist in how data is being stored, accessed and processed.

For long-term SHM systems very large amounts of data can accrue, up to the point that data storage is insufficient. Several techniques exist to deal with data storage limitations:

- Move data to an archive [1]
- Delete data not associated with events of interest [2,3,8]
- Delete data after a certain time period [1]
- Only store processed data, thus omit all raw data [9]
- Omit all data except for predefined periods [10,11]

These techniques only deal with the storage limitation on the central processing computer, and certain techniques can already be implemented in distributed sensor networks (such as event association, only using processed data and monitoring for predefined periods). One or more data storage techniques are always required as data storage limitations are always present.

Data storage is not the only limitation in a SHM, but it is an issue very often mentioned in research papers. In the post-processing phase, data access efficiency can become an issue when large files need to be read, or large databases need to be accessed. Although the archiving of data is possible for large databases, this approach makes historical data difficult to access. Other techniques include the operation of several databases depending on the historical data length [12] or data splitting [8,9], possibly with the additional storage of statistical parameters [3,13].

To summarize, the following data management issues exist:

1. Data from sensors use different storage systems (files, database)
2. Sensor data is streamed at different frequencies
3. Sensor data representations are different (e.g. temperatures, tri-axis acceleration, strain)
4. Post-processing requires data from multiple sensors
5. Post-processing requires data for different lengths of time
6. Post-processing creates new data
7. Post-processing requires efficient data access
8. Multitude of data storage limitation mitigation techniques

Many researchers recommend the creation of intelligent algorithms to generate metrics, to detect damage and to form a generalized theory [6,14,15]. All require the evaluation of historical data to obtain these results, however none have addressed above mentioned issues of storing, accessing and processing of the monitoring data. The new data structure proposed here endeavors to address these issues and will hopefully be a stepping stone towards better sensor data management and processing.

Aktan et al. [15] stressed the need of tackling the information technology (IT) issues for SHM in 2001 and to date, many SHM systems still focus on sensor usage, deployment architecture and comparison of post-processed measurements with simulation analysis. With the proposed data structure as an example, this paper intends to re-initiate the discussion towards further IT developments in SHM.

2. DATA MANAGEMENT LIMITATIONS

In the few papers where a data management method is mentioned in the context of SHM, discussions center on the use of files [8,13], a database [1,2,9,12] or a combination of both [3]. Focus is normally on the storage method, not on how data is efficiently accessed for processing.

To efficiently access data stored in files, a file indexing scheme should be present which captures the specifics of each file, for example an associated metadata file containing the measurement start and end times. In some cases this information is stored in the file header. For example, the monitoring system of the new Svinesund Bridge between Norway and Sweden mentions that for each 10-minute sensor file, a statistical file with mean, maximum, minimum and standard deviation is created, plus an identification of the data logging time [13]. For the SmartSync system, employed on the Burj Khalifa in Dubai, this statistical information is stored into a database [3].

In the examples where databases are used in SHM [1,2,3,9,12], no mention of the indexing scheme is made as the standard database indexing technique is assumed. There are however mentions of the database types used such as SQL [2], MySQL [3] and Oracle [12]. These traditional databases however are not designed for sensor systems, making them inefficient for SHM.

The reason for this inefficiency comes from the difference between sensor network data management requirements and the capabilities of traditional databases. These differences are summarized for distributed sensor networks by Li et al. [16]:

1. In database systems, the details of the network are of no concern. In sensor networks, the sensor's spatial location and its status are of importance.
2. Sensor networks produce infinite data streams where traditional database systems cannot manage infinite streams.
3. Continuous monitoring requires a continuous data query to keep track of the data inflow. Traditional database systems do not have a continuous query capability, so cannot deal with real-time responses.
4. Query processing techniques in traditional database systems are not suitable: optimization is based on statistical information while for sensors this statistical information is not readily present (it can only be generated after a certain time period) and is different for each sensor.

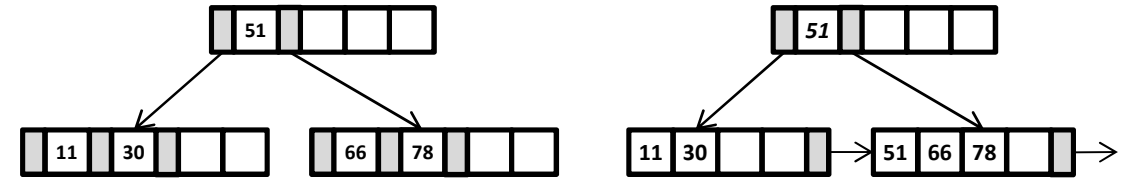
Nevertheless, database systems are much more efficient than simple file access and a combination of both files and database usage, with an 'application' dealing with the mentioned limitations above offers a workable solution. The data structure proposed in this paper is a step in this direction.

3. DATA STRUCTURES

At the heart of a data management system is a data structure, defined as a way of storing or organizing data so that it can be used efficiently. A data structure is usually designed with certain usage in mind and there are therefore many different data structures available [17]. This paper focuses only on the data structure and does not include any usage algorithms; also because these algorithms can be different for each SHM system.

Many data structures are "tree-like", as tree traversal is much more efficient than a full data search. The databases used in previously mentioned SHM systems (SQL [2], MySQL [3] and Oracle [12]) all have a B-tree [18] or a B-tree deviation as their basic data structure.

To explain how the tree structures work, figure 1a shows the B-tree, while figure 1b shows the B+-tree [18], an improvement of the B-tree. Each tree contains 5 data elements with values 11, 30, 51, 66 and 78. Each of the blocks is referred to as a node, the bottom nodes are leaves (a node without children) and the top node is the root (there can only be one root node). A search algorithm in the B-tree and B+-tree work the same, however data distribution is different. If one would try to find data value '71',



a) B-tree **b) B+-tree**

Figure 1: The B-tree and B+-tree both with 5 data elements. A grey element is a pointer to a child node and italicized text represents a key value only (no data).

the search algorithm would start with the root node, where value '51' occurs. As '71' is higher than '51', the algorithm would traverse to the child node to the right. In this node it would check the elements for their value. For the B-tree, if the pointer between '66' and '78' would point to another node, it would traverse to this node. Such node is not present, so the search algorithm returns 'not found'.

The only difference between the B-tree and B+-tree is that for the B+-tree all data is stored in the leaf nodes. The element '51' in the top node is only considered a key, not a data element. The definition of a key is "the part of a group of data by which it is sorted, indexed, cross referenced, etc." [17]. Thus if the value '51' were to be deleted, in the B+-tree that would only require the removal of the data element (the key would stay '51'), in the B-tree that would require a reorganization of the tree as in this tree every element is a data element.

The advantage of the B+tree is that all data are stored in the leaves and are sequential. To retrieve sequential data spanning multiple leaves, multiple tree traversals are not required in the B+-tree, but would be with the B-tree. It is therefore more efficient.

In traditional databases structures such as the B+-tree are only created for the selected index, often chosen to be the sensor identifier. Indexing time only with a B+-tree is inefficient due to the sequential nature of time and a search query would return the data of multiple sensors at once. Thus, data structures dealing with time, also called version indexes, were developed, such as the TSB-tree [19].

The disadvantage of the TSB-tree and many other developed version indexes [20] is that they split their nodes by time or by key, storing an index with both time and key. If version indexes are implemented in sensor networks, each measurement point would need to be inserted into the database with a timestamp and sensor identifier. This insertion is computationally inefficient as it creates many tree branches and the index plus timestamp for each data point creates a very large overhead.

In SHM and other sensor networks, each sensor has a certain location, thus spatial data structures should not be overlooked. One of the simplest spatial data structures, one which is in fact capable of

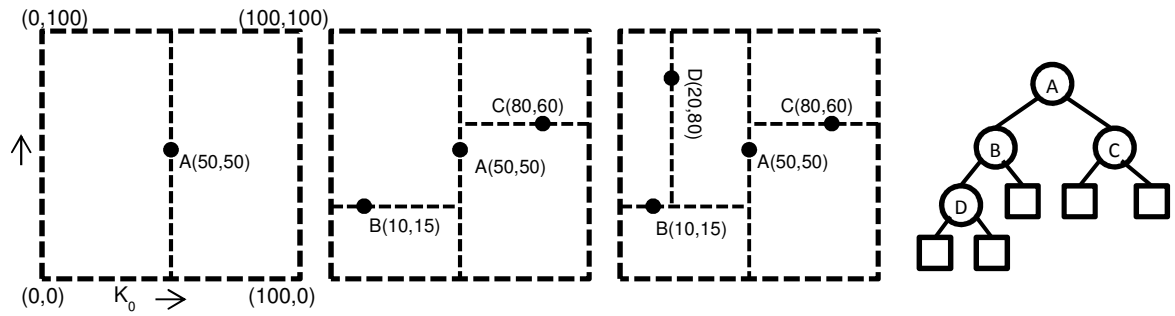


Figure 2: The kd-tree splits k-dimensional space into blocks, where each block is consequently split up, or contains data. In the tree representation a circle (○) represents a key node, while a square (□) represents a data node.

multidimensional indexing, is the kd-tree [21]. Instead of using a single value as the index as for the B-tree or their deviations, the kd-tree uses a k-dimensional index. Figure 2 shows this process for two dimensions.

The indexing process of the kd-tree example of figure 2 is as follows. Based on the amount of data in the space from coordinates (0,0) to (100,100), the space is split in two, with point A at coordinates (50,50) the reference coordinate for the vertical space split. What results are two spaces, each containing different groups of data.

In this example, the amount of data is still too large to be properly indexed, and therefore each of the two spaces is split again. On the left side, the horizontal split occurs at reference coordinate (10,15), marked as B. On the right side the horizontal split occurs at (80,60), marked as C. These locations are determined based on the coordinates of the data groups inside the space.

At this moment, the right side contains the correct amount of data to be indexed, and no other split is necessary. On the left top side however another split is needed, this time performed at reference coordinate (20,80), marked as D.

As can be seen, the split always follows a vertical, then horizontal pattern so that the space contained becomes gradually smaller. The reference points are used as key in the kd-tree, as shown on the far right of figure 2.

Another common spatial data structure is the R-tree [22], implemented in many databases supporting spatial queries. Figure 3 shows its spatial division is based on bounding boxes, where each bounding box is similarly traversed as with B-trees.

To include the temporal domain into spatial data structures, spatio-temporal data structures can be used. Two of the most common spatio-temporal trees are the historical R-tree (HR-tree) [23] and the time-parameterized R-tree (TPR-tree) [24], which both index on the spatial and temporal dimensions simultaneously.

The disadvantage of spatial or spatio-temporal data structures in sensor networks is that only the temporal domain is subjected to change, while the spatial domain hardly ever changes, unless sensors are added or removed. Spatio-temporal data structures are therefore more suitable for dealing with moving objects over time, such as in transportation tracking.

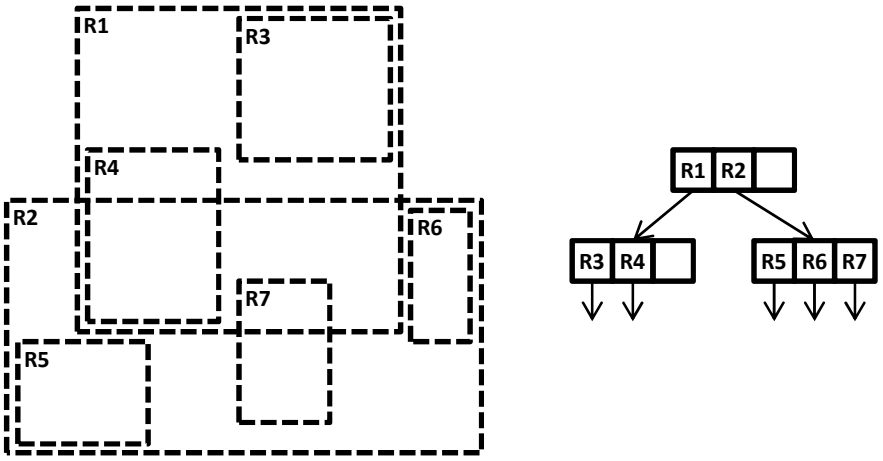


Figure 3: The R-tree uses bounding boxes as the keys for tree traversal. In the tree each square represents a bounding box key node, while the arrows from R3 to R7 represent data nodes.

For post-processing, the data itself is of most interest, thus a time series of data is likely to be requested. For such situations, the ability to sequentially retrieve data would be beneficial. In spatial or spatio-temporal data structures this is not possible and would require complete tree traversal every time data is requested.

It would be best if data could be stored sequentially while maintaining an efficient indexing scheme. This can be achieved by splitting the spatial component and temporal component into two different indexing structures. This essentially creates one spatial index and many temporal indexes: one temporal index for each sensor. This approach has been described by Noël et al. [25], who developed the Po-tree in which both the spatial and temporal components are split up into indexing trees (see figure 4).

The Po-tree addresses many of the issues mentioned in the introduction to this paper and also addresses the requirements for data management. The items which the Po-tree does not address are:

1. Post-processing requires data for different lengths of time
2. Post-processing creates new data
3. Post-processing requires efficient data access
4. Multitude of data storage limitation mitigation techniques

The following section will introduce a new data structure and section 5 will explain how the structure can address these outstanding issues.

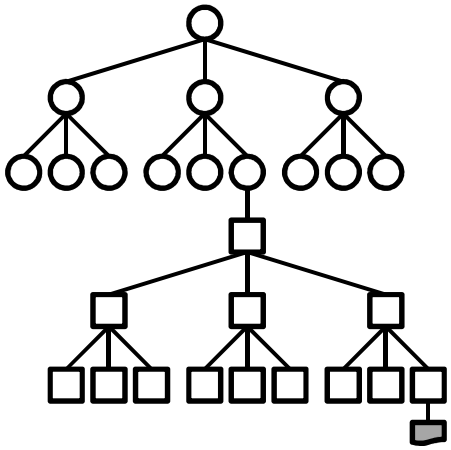


Figure 4: The Po-tree division (redrawn from Noël et al. [25]), where each circle (○) represents a spatial node, each square (□) represents a temporal node and the grey shape (▭) represents data.

4. NEW DATA STRUCTURE

The proposed data structure is inspired by the Po-tree and also consists of two indexing structures. The top data structure can be either spatial or key division, where the Po-tree is defined to use a spatial division only. Figure 5 shows the data structure divided into the top and bottom data structures, and immediately identifies significant differences with the Po-tree in its bottom temporal data structure.

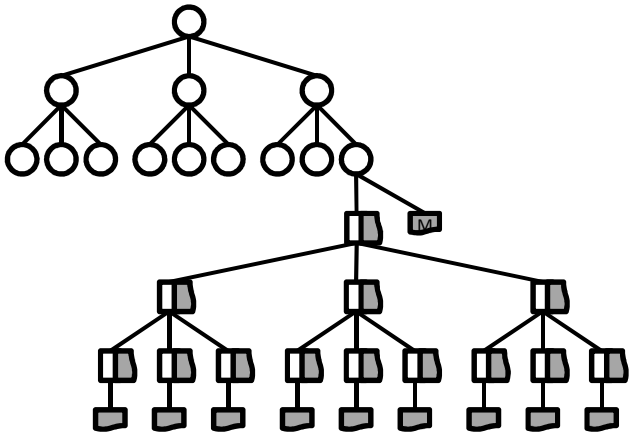


Figure 5: The new proposed data structure, where each circle (○) represents a spatial or key node, each unshaded rectangle (▯) represents a temporal node and a grey shape (▭ and ▱) represents data.

While the spatial division of the Po-tree is limited to that of a kd-tree, the new structure does not require the kd-tree to be used: any spatial data structure can be chosen. In certain cases, and most likely to occur in structural monitoring due to its limited amount of sensors, this data structure does not have to be based on spatial location but may be a sensor identifier (key). However, the top data structure must follow these requirements:

1. The leaf node must only contain a single sensor.
2. The sensors should be spatially quasi-static, thus not change its location in the spatial data structure (when a spatial data structure is used).

The final requirement means that certain spatial movement is allowed, as long as care is taken that the sensor does not modify the spatial tree structure. A modification in the tree structure would require the complete restructuring of the underlying temporal data structure, which results in inefficient processing. The same however is valid for the original Po-tree.

For the benefit of the temporal division, each leaf node of the top data structure will link to important metadata about the sensor and to the temporal data structure (the polygon marked M in figure 5). This allows for complete flexibility in the temporal data structure design. The contents of this metadata file are described after the explanation of the data structure in the next section.

The idea for the new temporal data structure comes from combining the notion of temporal division (similar to a kd-tree) with the consecutive blurring of images (as used in image processing for the scale-invariant feature transform (SIFT) [26]). The division of the temporal data structure means the splitting of data files into manageable chunks, while blurring equates to the averaging of data to obtain representative statistical information. As blurring is the main feature of the new technique, the resultant data structure will be named the *temporally blurred tree*, or TB-tree.

The tree structure is based on the B+-tree, where the leaf nodes refer to the original data files and are in sequential order. The data files are stored with certain measurement duration, and for each file averages and other statistical values are calculated, and stored into the parent node. This is similar to the procedure performed for the new Svinesund Bridge [13], with the exception that this averaging spans multiple tree levels and that timestamps are used as database indexes.

Each blur level of the TB-tree (with level 0 being the original data, level 1 being the first blur, and so on) has its own defined time span, depending on the sensor measurement frequency and required post-processing analysis. Figure 6 shows an example of how data is being blurred and how timestamps are

used as indexes. Ten original data points are blurred into averages and other statistical information at level 1. The average values of level 1 are used to blur at level 2, and so on.

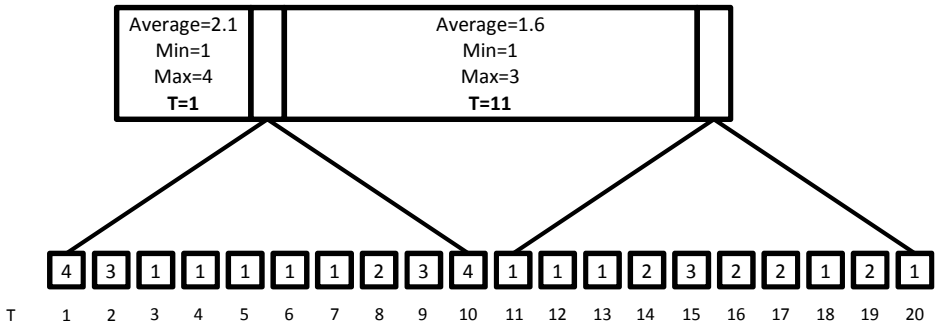


Figure 6: Example of blurring data for T=1 to T=20.

5. ADDRESSING SENSOR NETWORK ISSUES

In the introduction the following issues were identified for sensor networks:

1. *Data from sensors use different storage systems (files, database)*

Although the proposed data structure does not directly address this issue, there is no other system which deals with this problem immediately. For any sensor network implementation, software has to be developed to deal with this item, even if it were to use only one storage system. If this software employs the new data structure, it will be able to deal with the other issues.

2. *Sensor data is streamed at different frequencies*

Due to the split between a spatial/key data structure and temporal data structure, each TB-tree can be specifically designed for the measurement frequency of the sensor data.

3. *Sensor data representations are different (e.g. temperatures, tri-axis acceleration, strain)*

Similar as item 2, the TB-tree can be specifically designed to deal with the measurement variables.

4. *Post-processing requires data from multiple sensors*

Multiple sensors can easily be selected with the spatial/key data structure, after which sensor data is retrieved through the TB-tree.

5. *Post-processing requires data for different lengths of time*

This is the first issue not addressed by the Po-tree, but is addressed by the TB-tree. Each blur level represents periodic averages, allowing the selection of longer data the higher the blur level. In SHM, trends over very long periods, possibly months or years, are to be identified, while the sensor data is retrieved at (for example) 200 Hz. It is very inefficient to have to access and process each and every data point. Instead, one can select a higher blur level (possibly containing daily averages) and use these values for trend detection.

It must be noted that the amount of blurring for each level should be identified before sensor network operation, just like a traditional database has to be set up. A modification in the blur levels after data has already been gathered requires the re-processing of all data to generate the new averages and statistical information. This however has to be done only once, after which the average values can be used in multiple post-processing algorithms.

6. *Post-processing creates new data*

When certain post-processing algorithms require data from multiple sensors, it is highly likely it will output new information. Additionally, many post-processing techniques reduce multiple data points to a single value (such as frequency processing). To accommodate this additional data, a spatial/key leaf

node split can be performed and a new TB-tree is formed for the processed data. The metafile associated with the sensor and the metafile associated with the new spatial/key leaf node can then contain the information about the additional processed data.

7. *Post-processing requires efficient data access*

Although access to the data itself is already efficient through the tree indexing system, not all data access queries are efficient. A traditional database is efficient in obtaining data given a certain sensor identifier and less efficient when a sensor identifier and time span are given. The Po-tree solves this issue through the split between a spatial and temporal data structure, and this efficiency is kept with the new data structure employing the TB-tree.

In sensor networks this however is not the only query. More common queries are threshold searches for event occurrences. To find data where thresholds are exceeded, in current data structures all data has to be accessed and identified to be associated with the event or not. With the statistical information in multiple levels of the TB-tree, search algorithms can immediately discard TB-tree branches where the threshold has not been reached, making threshold searches much more efficient.

As a bonus of the statistical values stored through the blurring algorithm, standard deviation can indicate a possible failing sensor. If the deviation suddenly increases while no events occurred, sensor failure can be detected. Other calculations for different statistical values can be added for the benefit of other processing algorithms.

8. *Multitude of data storage limitation mitigation techniques*

It is counterintuitive that a data structure that adds statistical information at each level can decrease the required storage. However, the data storage limitation techniques mentioned in the introduction can all be applied in this data structure:

- Move data to an archive
- Delete data after a certain time period
- Delete data not associated with events
- Only store processed data, thus omit all raw data
- Omit all data except for predefined periods

Although an indexing structure is common to databases, original data is still stored in files. The files can thus be moved to an archive. Even when the original data is archived, the average and statistical information can still be maintained. The same accounts for the removal of old files, and for time periods where event detection algorithms have identified no event took place. It is then also clear to see how only processed data can be kept in level 1 of the TB-tree, and how all original data files, except for certain time periods, can be deleted.

The combination of the spatial/key data structure and TB-tree allows for storage reduction through distribution of sensor data. Sensors have both a set of static data and dynamic data, as shown in table 1. Storing the static data at each measurement will generate an unnecessarily large overhead and it can therefore be stored only once per sensor, in the metafile of the spatial/key leaf node.

Static data	**Dynamic data**
Metadata (accuracy, sensor type, …)	Time stamp
Approximate location	Measurement value(s)

Table 1: Static and dynamic data of sensors

Data reduction continues in the TB-tree itself. In traditional databases each data point is stored together with the timestamp of this data point. If the data stream is sequential and the sensor measures at a fixed frequency, the timestamp is not necessary. Instead of storing a time stamp at every measurement value, the measurement frequency has to be stored only once: in the metafile. The

indexing system of the TB-tree will always contain the timestamp of the first data point of the file as needed for tree indexing.

All of the contents of the metafile can now be compiled. Table 2 shows the information contained in the metafile.

Data	**Value**
Metadata	Sensor id, type, accuracy, etc.
Approximate location	Coordinates
Tree identifier	Id value
Measurement frequency	TB-tree level 1 TB-tree level 2 TB-tree level *n*
Calculation results 1	Link to identifier of tree
Calculation results 2	Link to identifier of tree
Calculation results *n*	Link to identifier of tree

Table 2: Summary of metadata file contents at each spatial/key leaf node

6. THEORETICAL EXAMPLE

One of the most important measures in SHM is the frequency of a structure – measured through accelerometers and time series data analysis. This measure however has a dual-temporal nature, as seen in the following example.

The natural frequency of a high-rise building is in the order of 0.1 to 2 Hz [27] while earthquakes vibrate with a frequency range of 0.001 Hz to more than 10 Hz [28]. To measure earthquakes successfully, a measurement frequency of 100 Hz is recommended. However for natural frequencies a measurement of 10 Hz is sufficient.

To be able to cover both earthquake and natural frequency monitoring, an accelerometer is therefore measuring a building at a frequency of 100 Hz. This 100 Hz data will be stored as raw data in level 0 of the TB-tree. To obtain the 10 Hz data sufficient for natural frequency calculation, every 10 samples of the 100 Hz data is averaged and stored in level 1 of the TB-tree. Level 2 can be an even further reduction of data to facilitate other information conversion algorithms, such as keeping track of the frequency every 10 minutes.

In this example it is clear that the raw data (level 0) can be omitted if no earthquake event was detected. Without the ability of the TB-tree and if all data was to be kept for post-processing, the 100 Hz data would use 10 times more space than the actual required 10 Hz data. With the employment of earthquake detection algorithms the raw data can be removed when not needed. The algorithms will have to indicate how much of the 100 Hz data is to be kept for future post-processing in case an earthquake did occur.

7. CONCLUSION

The following 8 data management issues for sensor networks were identified:

1. Sensor data from sensors use different storage systems (files, database)
2. Sensor data is streamed at different frequencies
3. Sensor data representations are different (e.g. temperatures, tri-axis acceleration, strain)
4. Post-processing requires data from multiple sensors
5. Post-processing requires data for different lengths of time
6. Post-processing creates new data

7. Post-processing requires efficient data access
8. Multitude of data storage limitation mitigation techniques

A new data structure is proposed based on the initial spatial or key division, after which the temporal domain is indexed. This data structure is particularly well suited for time-sequential data as commonly occurring in sensor networks, and addresses the above 8 data management issues.

The data structure focuses on the development and benefits of the temporally blurred tree, or TB-tree, for the temporal division. The TB-tree stores raw data in its lowest level and blurs (averages) this data into higher levels, allowing easier trend processing and event threshold detection. Due to its flexibility, additional algorithms are able to integrate into the data structure.

Additionally the paper identified how data reduction can be applied to this data structure by distributing the sensor static and dynamic data, and how currently employed data reduction techniques are applicable to this structure. A theoretical example was presented to show its usage.

This paper revisits the IT needs for successful and future-proof structural health monitoring systems. It is the hope of the authors that the new data structure will be a stepping stone for further IT development in structural monitoring systems.

8. REFERENCES

[1] Yinghong Cao, Yi Zhang, Yang Zhao, and Ming L. Wang. Distributed Health Monitoring System for Zhanjiang Bay Bridge. *Advances in Structural Engineering*, 14:1–12, 2011.

[2] Hui Li, Jinping Ou, Xuefeng Zhao, Wensong Zhou, Hongwei Li, Zhi Zhou, and Yongshun Yang. Structural Health Monitoring System for the Shandong Binzhou Yellow River Highway Bridge. *Computer-Aided Civil & Infrastructure Engineering*, 21:306 – 317, 2006

[3] D. Kwon, T. Kijewski-Correa, and A. Kareem. SmartSync: An Integrated Real-Time Monitoring and SI System for Tall Buildings. In *2010 Structure Congress*, pages 3176–3185, Orlando, USA, May 2010.

[4] A.E. Aktan, F.N. Catbas, M. Pervizpour, E. Kulcu, K. Grimmelsman, R. Barrish, and X. Qin. Real-Time Bridge Health-Monitoring for Management. In *2nd Workshop on Advanced Technologies in Urban Earthquake Disaster Mitigation*, July 2000.

[5] Chung-Bang Yun, Jong-Jae Lee, Sung-Kon Kim, and Ji-Won Kim. Recent R&D Activities on Structural Health Monitoring for Civil Infra-Structures in Korea. *KSCE Journal of Civil Engineering*, 7:637–651, 2003.

[6] Helmut Wenzel. Health Monitoring of Bridges. John Wiley & Sons, Ltd, 2009.

[7] Andrey Shakhramanyan, Joost Kuckartz, and Yuri Kolotovichev. Modern Structural Monitoring Systems for High-Rise and Unique Buildings. In *4th International Workshop on Civil Structural Health Monitoring (CSHM-4)*, Berlin, Germany, November 2012.

[8] James Mark William Brownjohn and Tso Chien Pan. Response of Tall Buildings to Weak Long Distance Earthquakes. *Earthquake Engineering & Structural Dynamics*, 30:709–729, 2001.

[9] Noemi Emanuela Cazzaniga, Livio Pinto, Franco Bettinali, and Antonella Frigerio. Structural Monitoring with GPS and Accelerometers: The Chimney of the Power Plant in Piacenza, Italy. In *3rd IAG, 12th FIG Symposium*, May 2006.

[10] S. Sumitro and M.L. Wang. Sustainable Structural Health Monitoring System. *Structural Control and Health Monitoring*, 12:445–467, 2005.

[11] Daniele Inaudi, Andrea Del Grosso, Thomas Messervey, Roberto Walder, and Clemente Fuggini. Long-Term Structural Health Monitoring of the Royal Villa of Monza. In *3rd International Workshop on Civil Structural Health Monitoring (CSHM-3)*, August 2010.

[12] Y.Q. Ni and H.F. Zhou. Guangzhou New TV Tower: Integrated Structural Health Monitoring and Vibration Control. In *2010 Structure Congress*, pages 3155–3164, Orlando, USA, May 2010.

[13] R. Karoumi, G. James, F. Myrvoll, L. Lundh, and R. Ronnebrant. Monitoring the Structural Behaviour of the New Svinesund Bridge. In *6th European Conference on Structural Dynamics (Eurodyn)*, September 2005.

[14] Charles R. Farrar and Nick A.J. Lieven. Damage Prognosis: the Future of Structural Health Monitoring. *Philosophical Transactions of the Royal Society – Series A: Mathematical, Physical and Engineering Sciences*, 365:623–632, 2007.

[15] E. Aktan, S. Chase, D. Inman, and D. Pines. Monitoring and Managing the Health of Infrastructure Systems. In *2001 SPIE Conference on Health Monitoring of Highway Transportation Infrastructure*, March 2001.

[16] Jinbao Li, Zhipeng Cai, and Jianzhong Li. Data Management in Sensor Networks. In *Wireless Sensor Networks and Applications, Signals and Communication Technology*, pages 287–330. Springer US, 2008.

[17] Paul E. Black. Dictionary of Algorithms and Data Structures. National Institute of Standards and Technology, 2013.

[18] Douglas Comer. The Ubiquitous B-Tree. *ACM Computing Surveys (CSUR)*, 11(2):121–137, 1979.

[19] David Lomet and Betty Salzberg. Access Methods for Multiversion Data. In *1989 ACM SIGMOD International Conference on Management of Data*, pages 315–324, Portland, USA, June 1989

[20] Rui Zhang and Martin Stradling. The HV-tree: a Memory Hierarchy Aware Version Index. *Proceedings of the VLDB Endowment*, 3(1-2):397–408, 2010.

[21] Jon Louis Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9):509–517, 1975.

[22] Antonin Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. In *1984 ACM SIGMOD International Conference on Management of Data*, pages 47–57, Boston, USA, June 1984.

[23] Mario A. Nascimento and Jefferson R.O. Silva. Towards Historical R-trees. In *1998 ACM Symposium on Applied Computing*, pages 235–240, Atlanta, USA, February 1998.

[24] Simonas Šaltenis, Christian S. Jensen, Scott T. Leutenegger, and Mario A. Lopez. Indexing the Positions of Continuously Moving Objects. In *2000 ACM SIGMOD International Conference on Management of Data*, volume 29, pages 331–342, Dallas, USA, May 2000.

[25] Guillaume Noël, Sylvie Servigne, and Robert Laurini. The Po-tree: a Real-Time Spatiotemporal Data Indexing Structure. In *11th International Symposium on Spatial Data Handling*, pages 259–270, 2005.

[26] David G. Lowe. Object Recognition from Local Scale-Invariant Features. In *Seventh IEEE International Conference on Computer Vision*, pages 1150–1157, Kerkyra, Greece, September 1999.

[27] Q.S. Li, Y.Q. Xiao, J.Y. Fu, and Z.N. Li. Full-Scale Measurements of Wind Effects on the Jin Mao Building. *Journal of Wind Engineering and Industrial Aerodynamics*, 95:445–466, 2007.

[28] J. Douglas. Earthquake Ground Motion Estimation Using Strong-Motion Records: a Review of Equations for the Estimation of Peak Ground Acceleration and Response Spectral Ordinates. *Earth-Science Reviews*, 61(1):43–104, 2003.

9. ACKNOWLEDGEMENTS

This research is supported by the Australian Research Council under a linkage project between the University of Melbourne and the University of New South Wales and industry partners C.R. Kennedy Pty. Ltd., Leica Geosystems, Vekta Pty. Ltd and NPO SODIS LLC. Thanks to Dr. Rui Zhang of the Department of Computing and Information Systems of the University of Melbourne for his guidance in the data structure proposal during the Advanced Studies in Computing subject.