Sharing Sensor Network Data[*][†]

Gong Chen Nathan Yau Mark H. Hansen Deborah Estrin
Center for Embedded Networked Sensing
University of California, Los Angeles
Los Angeles, CA 90095

Abstract

Sensor networks generate a variety of data streams in different temporal and spatial resolutions. The data come as numbers text, images, and audio and are dynamically produced periodically and sporadically. How can we organize hundreds of thousands of such data streams? How can we make it easy for scientists and engineers to publish and share such data streams? In this paper, we present our solution, SensorBase.org. It is a web application that not only provides the user with the functionality of a traditional database management system, but also runs under the notion of a Web 2.0 data experience with a responsive user interface design and RSS data feed techniques. SensorBase.org also aims to be a data search engine to promote exploration. Like a web search engine, the user should be able to search for structures, or rather, "signals", in the data using simple language queries. We provide a solution to a specific type of signal search problem and describe a search framework derived from the solution. A visualization component is in development to help people understand the data and to enhance the results provided by a data search engine.

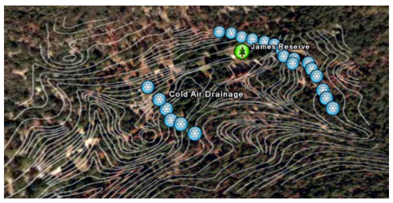
The development of SensorBase.org allows us, the statisticians, to influence "data practices". People share data and add documentation within the guidelines of SensorBase.org which supports good experimental design and helps future modeling efforts. Last but not least, SensorBase.org provides a base for data integrity and data quality.

[*]Gong Chen and Nathan Yau contributed equally to this work.
[†]The authors gratefully acknowledge support from NSF CCF-0120778, NSF CNS-0626702 and NSF CNS-0627084.

1 Introduction

The Center for Embedded Networked Sensing (CENS) at UCLA develops various types of sensors to collect many types of data from natural and built environments. Networks of wireless sensors are set up to take high granularity measurements which allow observers to evaluate surroundings in a different way than they normally do. For example, temperature, humidity, and wind data are collected by motes to monitor environmental changes while images, videos, and audio are collected by mobile phones to monitor urban settings. With such continuous data streams that demand a high level of analysis, a database is absolutely necessary to keep things organized.

We can get a feel for embedded sensing with a few examples. At James Reserve in the San Jacinto mountains, a collection of twenty-five wireless sensors, known as the Cold Air Drainage (CAD) transect, records temperature data every five minutes. In Figure 1 (left), we can see a map of the transect. At the nearby Lake Fulmor, a line of buoys reside with dangling thermistors that collect time series temperature data at different depths. A half meter below each buoy a fluorometer measures chlorophyll concentrations.



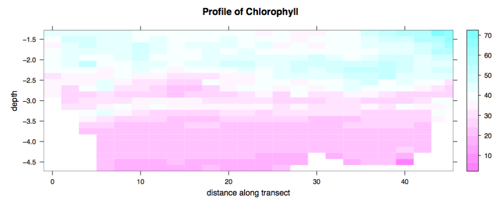


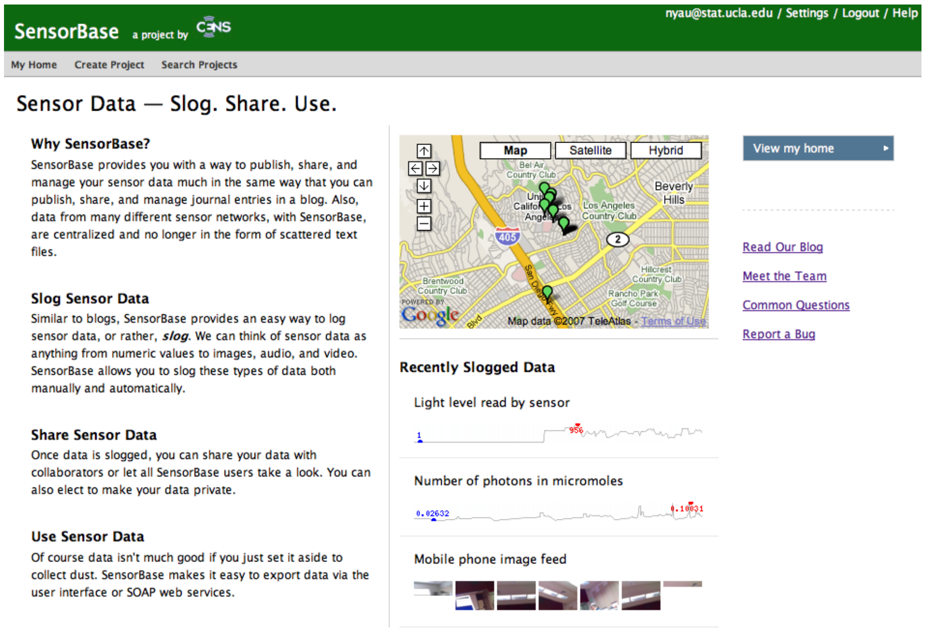
Figure 1: CAD transect at James Reserve (left) and Chlorophyll concentrations at Lake Fulmor (right)

Four times a year, a robotic sensing device is installed, which is a shuttle suspended on a cable that hangs across the lake perpendicular to the buoy line. The shuttle moves along the cable, dips down a sensor node at regular intervals, and then takes measurements at several depths. Temperature and chlorophyll measurements for one pass over the lake can be seen in the grid in Figure 1 (right).

In a completely different setting, where buildings replace trees and people replace animals, sensors, in the form of mobile phones, collect media data. The urban setting brings in different types of data in the sense that much of it is non-numeric and it is much more sporadic, taking the form of periodically uploaded digital recordings, images or video.

This large variety of data and locations highlights a great need for a database in order to make data management easy. However, given the diverse population of end-users for these data (a group which includes subject-area scientists as well as engineers and computer scientists), it is not reasonable to expect everyone to know SQL statements and database structures. Instead, to abstract the database, and to push focus on the data being stored, we have created Sensor-

Base.org (Figure 2). Not only have we made publishing data easy, but once the data is available users can share, manage, and search continuous data streams. In Section 2 we discuss some of the already existing applications to manage generic data and how we improve on these models for embedded sensing applications. In Section 3, we go over SensorBase.org design and implementation and then end with a recipe to search data in Section 4 and the future of our application in Section 5.



Sensor Data — Slog. Share. Use.

Why SensorBase?
SensorBase provides you with a way to publish, share, and manage your sensor data much in the same way that you can publish, share, and manage journal entries in a blog. Also, data from many different sensor networks, with SensorBase, are centralized and no longer in the form of scattered text files.

Slog Sensor Data
Similar to blogs, SensorBase provides an easy way to log sensor data, or rather, ***slog***. We can think of sensor data as anything from numeric values to images, audio, and video. SensorBase allows you to slog these types of data both manually and automatically.

Share Sensor Data
Once data is slogged, you can share your data with collaborators or let all SensorBase users take a look. You can also elect to make your data private.

Use Sensor Data
Of course data isn't much good if you just set it aside to collect dust. SensorBase makes it easy to export data via the user interface or SOAP web services.

Recently Slogged Data

Light level read by sensor

Number of photons in micromoles

Mobile phone image feed

Figure 2: SensorBase.org home page

2 Current Data Sharing Mechanisms

An informal survey of early embedded sensing deployments both at CENS and at other institutions suggests that lots of data exist as very large, unorganized, flat delimited text files. Excel spreadsheets rest a little bit higher on the organizational food chain, but when the number of observations gets higher in the millions, data management can easily get out of hand; and more importantly, sharing data becomes very difficult. If a collaborator needed data, the data owner would have to look for the appropriate log files and have to explain the data's who, what, when, and where to the collaborator. Once the data is explained, the collaborator would still have to sift through millions of observations, and it is highly possible he would not even know where to begin. Suppose the owner has long since left the research group. Unexplained data left behind is virtually useless. Outside the sensor networks community, we see excellent examples of data sharing practices.

Dabble DB (`dabbledb.com`) is an application that helps users share data a bit more easily. The application has an excellent dynamic user interface (UI) and does a great job of abstracting the database. However, Dabble DB is still limited in the type of data that can be stored and how data can be published. It can only store text, numbers, and date and time, and data can only be published via a UI. The nature of CENS data streams requires automatic publishing functionality as well as an ability to store media data, such as images, in addition to text, numbers, and time.

Google Spreadsheets (`docs.google.com`) is another application that makes sharing data a bit easier. Users can share spreadsheets, and edit spreadsheet data with others in real-time. However, like Dabble DB Google Spreadsheets is limited by the types of data that can be stored and how data can be published. Dabble and Spreadsheets functionality stops at the UI and the goal behind the two is mostly to store user input such as names and addresses and less about storing dynamic data coming from non-human sources.

Swivel (`swivel.com`) pushes data sharing in a different direction than the two previous. Topics include politics to weather and sports to technology. There are some pretty graphs sitting on a data exploration platform. Again though, for CENS's purpose, the scope is limited. Users can only upload text-delimited files and data updates stop after the file upload is complete. Published data also completely relies on user input.

While Dabble, Spreadsheets, and Swivel are beautiful applications in their own right, SensorBase.org is our answer to bring media, dynamic data, and more effective data exploration into the world of embedded sensing applications.

3 Design and Implementation

To make it easy for the user to publish and retrieve data we overcome several difficulties to design a database and a closely-coupled UI. We take a bottom-up approach informed by previous experiments [1], the input of engineering and application science faculty, students, and staff, and a core advisory board to take input from the different constituencies at CENS. The database schema is developed to be robust enough to handle the different types of data while at the same time guide the flow of data streams in a way that would make the data most useful during analysis. In addition to storage of the actual data, our schema allows us to manage metadata and documentation to provide context and meaning.

Next, we design the SensorBase.org UI to abstract the database. From the very beginning, our running analogy to aid us in UI design has been the UI for blogs. One of the reasons blogs have become popular is because blog UIs make publishing entries so effortless. With a few mouse clicks, a user can publish, delete, categorize, or set permissions on any given entry. In short, the evolution of blogs has made it easy for anyone to share their thoughts online. Additionally, urban applications that involve images and audio are similar to already popular data sharing platforms like YouTube (`youtube.com`) and Flickr (`flickr.com`). These metaphors guide SensorBase.org UI design and set standards that we have to meet. We strive for a responsive UI implemented with PHP, MySQL, and AJAX that makes it easy for anyone to share data online. We call this idea sensor logging, or rather, *slogging*.

3.1 Slogging Data

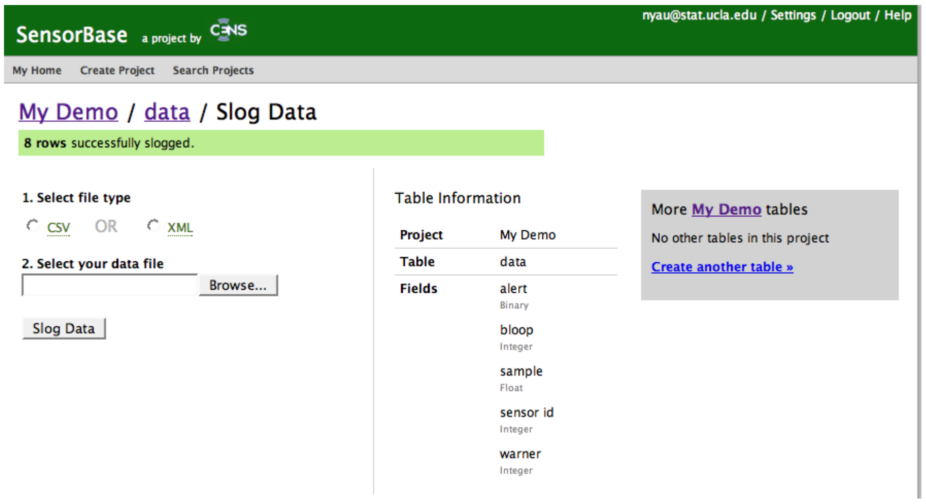
Our UI for slogging (Figure 3) handles different types of data generated via various methods. The user can slog numeric and textual data such as temperatures and sensor names, and just as easily, she can slog media data such as images, videos, and audio. In addition to slogging via the UI, the user can also slog data programmatically (i.e. automatically) via web services or an HTTP POST. The slogging work flow is as follows: (1) create a project; (2) define how data will be stored; and (3) slog data in the form of delimited text, XML, media files, or live data streams.

The idea is that every project has a home page where the user can stay updated on the type of data, from a specified sensor network, getting slogged. On the project page, the user can and will be able to find dynamic plots and maps as well as documentation that explains the data. In addition to the project home page, the user can also subscribe to RSS data feeds to stay updated on what is most recent. If the user wants to be notified, for example, every time a one-hour gap contains no data, she can subscribe to the appropriate feed and stay updated with her favorite RSS reader. It is also possible to subscribe to feeds that update the user whenever values have exceeded or dropped below a specified threshold as well as boolean combinations of threshold crossings. In the future, combining computation (with connections to R and other science-specific computing platforms) to RSS, we will allow users to generate numeric RSS feeds based on generic filters [4].

3.2 Annotating Data

Before SensorBase.org, CENS data rested in the form of flat text files on personal computers and data loggers. Aside from the difficulties of making such data highly available, the data was often difficult to understand not only for outsiders but also for the data owners. The only documentation that existed, if any existed at all, was in a “README” file that often was not updated regularly.

Sensorbase.org provides an easy way to annotate incoming data streams from the project-level down to a single observation. The user can highlight and annotate chunks of data to remind herself and let others know why data might appear strange or to explain anomalies in the data. For instance, a sensor’s battery might have died, a sensor might not have been deployed correctly, or connectivity might have been spotty. This is annotation at the data-level. The user can also describe her projects as well as individual sensors. In a collaborative effort, there might be multiple users who participate in a project. In this


SensorBase a project by CENS
nyau@stat.ucla.edu / Settings / Logout / Help
My Home
Create Project
Search Projects
Browse...
Slog Data

My Demo / data / Slog Data

8 rows successfully slogged.

1. Select file type

CSV OR XML

2. Select your data file

Table Information

Project	My Demo
Table	data
Fields	alert Binary
	bloop Integer
	sample Float
	sensor id Integer
	warner Integer

More My Demo tables

No other tables in this project

Create another table »

Figure 3: Data slogging UI

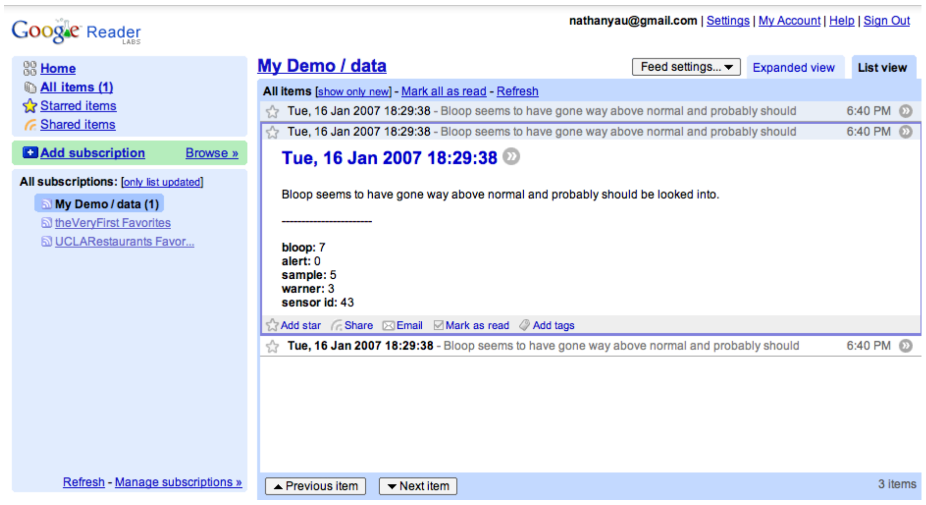
case, users can interact with each other with "project notes" which are similar to comments in blogs and can provide a deeper understanding of slogged data.

3.3 Setting Permissions and Access Levels on Data

It is likely that the user may not always want to share her data with anyone, and so the user can set access levels for who can read and slog data to her projects. The user can set her project either to "public", if she wants to let anyone who has a SensorBase.org account see her data, or to "private", if she wants to set access levels on an individual basis. The user can give permission to other users and give them one of the three levels of access — read only, read and write, or read, write, and build. Read access allows the user to retrieve data; write access allows the user to slog data; and build access allows the user to define how to store incoming data.

3.4 Retrieving Data

Once data comes in to SensorBase.org it is critical that it is easy to retrieve the data. However, in our case, data flow into SensorBase.org continuously, and the number of observations can grow into the millions in less than a month. For example, data from the CAD transect feed into SensorBase.org every five minutes, and a month's worth of data is over six million observations. Most of the time the user will retrieve a subset of a giant data set. Either via the UI or web services, the user can retrieve data within a certain time range, limit the number of returned observations, and set sorting by any specified variable (Figure 5). It is particularly important that we provide data via web services because it allows the user to create applications on top of SensorBase.org and



Tue, 16 Jan 2007 18:29:38 - Bloop seems to have gone way above normal and probably should 6:40 PM

Tue, 16 Jan 2007 18:29:38 - Bloop seems to have gone way above normal and probably should 6:40 PM

Tue, 16 Jan 2007 18:29:38

Bloop seems to have gone way above normal and probably should be looked into.

----

bloop: 7
alert: 0
sample: 5
warner: 3
sensor id: 43

Tue, 16 Jan 2007 18:29:38 - Bloop seems to have gone way above normal and probably should 6:40 PM

Figure 4: SensorBase RSS feed as seen in Google Reader

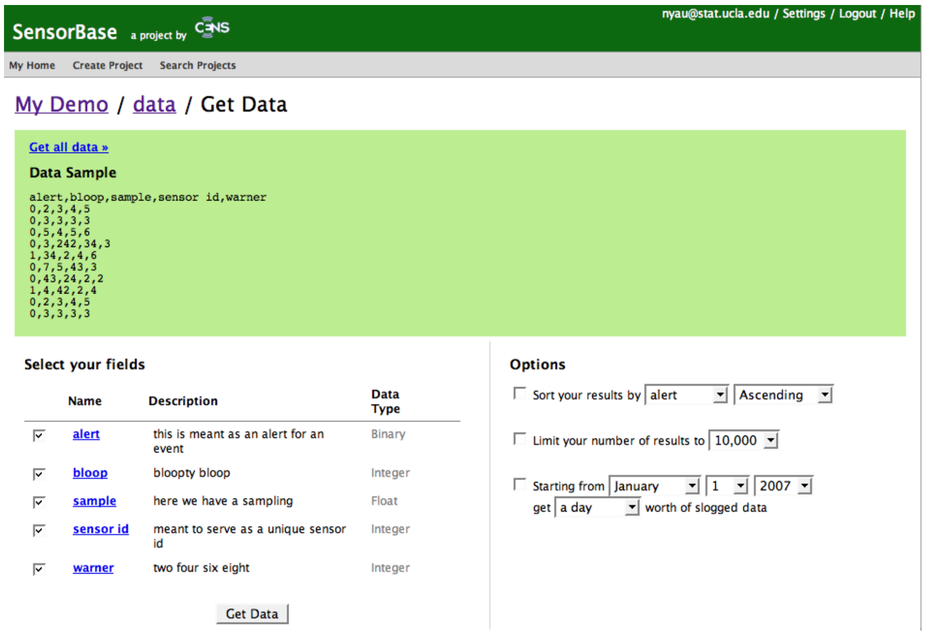
opens the opportunity for the user to create her own exploration tools or to feed data into existing software. Moving forward to more advanced data retrieval, we are working on so-called signal search functionality where the user can search for data based on features.

4 Searching Sensor Data

4.1 Variation Search Problem

To use a search engine such as Google, the user issues a query in the form of a keyword or phrase and receives a list of results related to the query. Similarly, the user should be able to identify structures in the data, in the "signal", using queries expressed in simple language. To clarify the problem, we start by answering a question posed by a CENS biologist and then describe a search framework derived from the solution.

A sensor network, designed to capture so-called Cold Air Drainage (CAD) events, collects temperature data. The biologist wants to study the temperature characteristics when potential CAD events occur. A sharp temperature drop between 4:00am and 8:00am indicates the occurrence of a CAD event, so he wants to find the data from when temperature drops more than 3 degrees Celsius over 1 hour during that time range. The nature of this question is fuzzy. Temperature might drop 10 degrees over half an hour. It might drop 2 degrees over 1 hour and continue to drop another 2 degrees for the next 10 minutes (on average, 3.4 degrees over 1 hour). Both are correct answers to the biologist's question. We must be able to detect both situations while searching the data. We will describe a simple framework for answering these boolean combinations


SensorBase a project by CENS
nyau@stat.ucla.edu / Settings / Logout / Help
My Home
Create Project
Search Projects

My Demo / data / Get Data

Data Sample

```
alert,bloop,sample,sensor id,warner
0,2,3,4,5
0,3,3,3,3
0,5,4,5,6
0,3,242,34,3
1,34,2,4,6
0,7,5,43,3
0,43,24,2,2
1,4,42,2,4
0,2,3,4,5
0,3,3,3,3
```

Select your fields

	Name	Description	Data Type
✓	alert	this is meant as an alert for an event	Binary
✓	bloop	bloopty bloop	Integer
✓	sample	here we have a sampling	Float
✓	sensor id	meant to serve as a unique sensor id	Integer
✓	warner	two four six eight	Integer

Options

☐ Sort your results by alert Ascending

☐ Limit your number of results to 10,000

☐ Starting from January 1 2007 get a day worth of slogged data

Figure 5: Data retrieval UI.

of these easily specified queries efficiently with any database backend.

4.2 Search Framework

We design a three-tier framework to extract features from data so that they can be stored in a database allowing us to transform searches into queries on the features.

1. There is a lot of noise in data collected by sensors because sensors can malfunction and generate anomalies. Data quality problems plague sensor deployments [5]. Local polynomial regression is a robust method that can be used to find the mean curve from noisy data.

2. Using a user-defined error tolerance for approximation and an online-algorithm [3], the curve is represented by piecewise linear segments. This approximation significantly reduces the number of extracted features search has to use, which speeds up feature extraction at the third tier. At this tier, segment, we record its starting time st, duration Δt and slope k for each segment.

3. At the third tier, we impose a sliding window over a certain period of time Δt_w on segments from the second tier as shown in Figure 6. Suppose a sliding window moves to the current time t and $i = 0$ is the index of the first segment in the sliding window from t. The accumulative variation

$v_i = \sum_{j=0}^{i} k_j \Delta t_j$ and the time span $sp_i = \sum_{j=0}^{i} \Delta t_j$ from t for segment i is recorded and stored in database.

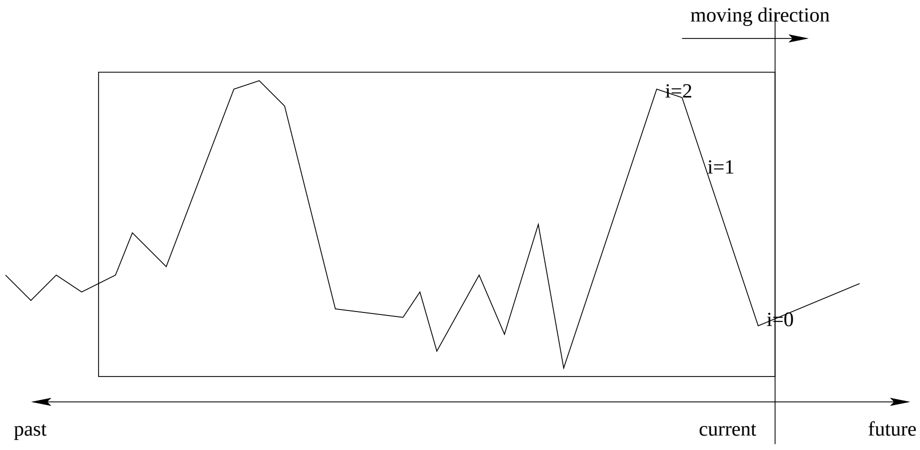


Figure 6: A sliding window at the third tier

We construct variation space to explore the search supported by extracted features. Figure 7 shows part of variation space where any point there is corresponding to some variation over a certain time period. Two features sp_i and v_i for each segment are mapped to this space. The point $(1, -3)$ at the cross of the vertical line $sp = 1$ and the horizontal line $v = -3$ denotes the event of the exact 3 degrees drop over 1 hour. The correct answers for the biologist's question should be in the region where $sp < 1$ and $v < -3$ or in the region where $sp \geq 1$ and $v < -3sp$. The former region is the situation that temperature drops more than 3 degrees over a time period less than 1 hour while the later one is the situation that the accumulative drop over longer time achieves the effect. The line $v = -3sp$ is plotted for reference in Figure 7.

However, we miss some variation by just returning cases with $v < -3$ and $v < -3sp$, the combination of the two regions mentioned above. Since variation that occurs inside a segment is not mapped to the space, it is overlooked due to compression. Nevertheless, two features sp_{i-1} and v_{i-1} from the other end of segment i can be used to find the missing case. For segment i, the point (sp_{i-1}, v_{i-1}) and the point (sp_i, v_i) is connected by a line in Figure 7. The part of that line falls into the answer region even if its two end points do not. To capture such kind of line, we want to form the query conditions as follows: (1) $-3 < k_i < 0$; (2) $sp_{i-1} < 1$ and $v_{i-1} > -3$; (3) $v_i + k_i(1 - sp_i) < -3$ and (4) $v_i > -3sp_i$. k_i is the slope of that line. The third condition is on the position of the point $(1, v_i + k_i(1 - sp_i))$ in the middle part of that line. The case with the features satisfying above four conditions has variation in the answer region and should be returned. Condition (1) and (3) can be used to derive an indexable condition $v_i - k_i sp_i < 0$. Multidimentional-index structures like K-D-B trees [6] can be built on this condition together with condition (1) and (2) to provide fast

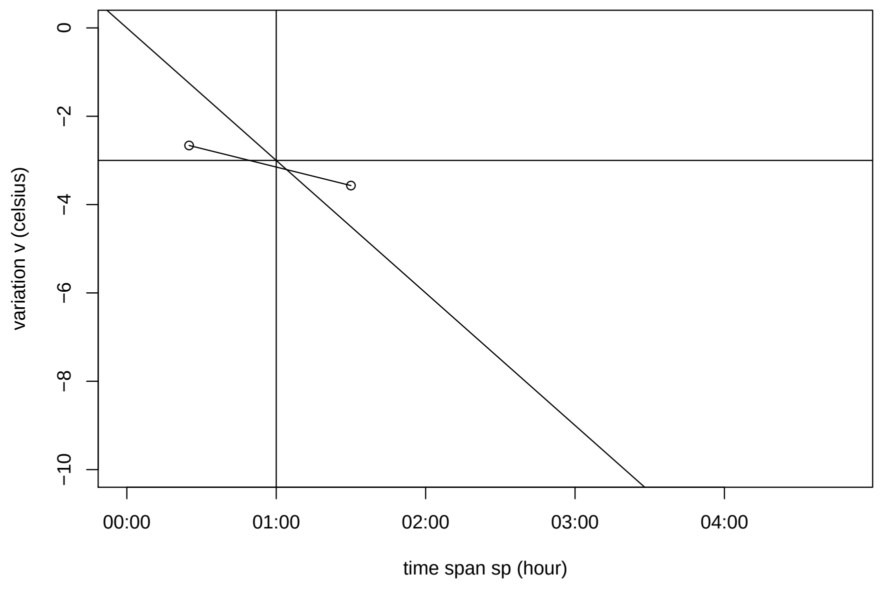


Figure 7: Variation space

access in database. Finally, we can generalize the above example as search for data when more than b units increase/decrease over a time units where $a > 0$ and $b > 0$.

In summary, the three-tier framework takes into account important performance factors such as feature reduction, fast processing, and query efficiency. Furthermore, compared with similarity search in time series databases [2] that requires a query series, human word-able search is suitable for most users who do not have exact ideas about the shape of series they are looking for. In the case of variation search, language description is more appropriate than a query series since there are numerous series that satisfy the description but have very different shapes.

The framework of variation search can be generalized to develop a data search component for SensorBase.org. Statistical computing extracts features from data streams, online algorithms make real-time feature extraction possible, databases store features along with the data, and indexing techniques provide fast access to these features so that search results can be quickly returned. In this way, the data become searchable. By specifying a few words as search criteria, the user can easily find the data she is interested in.

5 Future Work

Future developments for SensorBase.org include visualization of search results, improved searching functionality based on metadata, greater computing power with our existing data search framework, and standardized output based on existing and highly comprehensive data dictionaries.

Similar to a Google search, we see data search as an iterative process where a user will search, and if she does not find what she is looking for the first time, she goes back and changes her query and sees what new results are returned. It is at this stage that we feel it is absolutely essential to have useful visualization of the results. We have the benefit of knowing that most SensorBase.org data streams are time series and linked to location, so we imagine a combination of Sparklines [7] and maps to help users decide what data to explore. Sparklines, small "word-like" time series plots, allow us to display a lot of information in a small amount of space while maps offer a very intuitive interface and context to the data.

In Figure 8 we see four days of data from six sensors represented by six Sparklines. At the beginning of this time frame the sensors appear to be running as expected. After the first hump, is an anomaly followed by a wiggly signal and an occurrence of a CAD event. Around the last third of each Sparkline is a fair amount of noise relative to the rest of the data. At this point, there was a malfunction in the sensor network. The local minimum and maximum (in degrees Celsius), which can be thought of as metadata, are also marked for each frame.

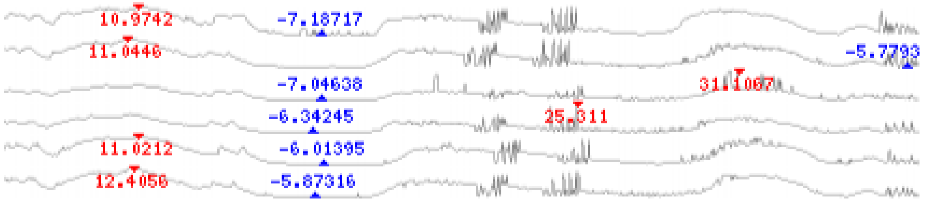


Figure 8: Four days of CAD temperature observations in degrees Celsius from 6 sensors.

As hundreds of thousands of data streams flow into SensorBase.org, it would be desirable to make their metadata searchable. The user may want to find data with the same calibration equation or may want to find out which part of data are collected when sensors malfunction. All such results come from annotations, one type of metadata. Furthermore, the examples from Section 1 indicate many data streams from different projects come from the same geographic area. Through metadata search, the user can find different data resources that monitor the same location. Sharing becomes more powerful when data are linked to provide different temporal-spatial resolutions of the world.

Our search framework sheds light on an architecture component we will need to build for SensorBase.org. Statistical computing processes can happen as data *stream* into the database. Computed results are stored along with data and made ready for search. Different types of data ask for different computing procedures to achieve different goals. We will set up a cluster of computing servers around SensorBase.org to make different computations happen smoothly. This component will couple with notification services from SensorBase.org to provide the user real-time alerts. As soon as data containing a predefined event stream into the database, the user subscribing to the RSS feed on that event will get a notification with computing results from her RSS reader. For example, a signature-based fault detection scheme is being implemented that will make use of the computational RSS idea to generate alerts for bad data.

As more and more users share their data with SensorBase.org, an open standard for data format is needed for transferring data in and out. This will facilitate the exchange of information. When the user slogs her data, she might want to slog metadata such as annotation for calibration equations and units of measurement. When a set of data is pulled out, all of its related metadata will be shipped out with the data package to make the set meaningful. To this end, we will design an XML schema serving as a data format standard. It will borrow descriptive power from complex schemas such as EML and SensorML but aim to own a light-weight specification to make it easy for common users to adopt.

6 Conclusions

Sensor networks are generating a variety of data streams in different temporal and spatial resolutions. We have built an application, SensorBase.org, to manage such streaming data, and more critically, to make data publishing and sharing easy for engineers, scientists and even citizens in urban settings. SensorBase.org not only provides the user with the same functionality of a traditional database management system, but also runs under the notion of a Web 2.0 data experience with a responsive user interface design and RSS data feed techniques. Our data search framework coupled with RSS provides a steady foundation to make seemingly endless data informative and useful. Enhanced with visualization to help people understand the data, SensorBase.org stands as a robust application to slog, edit, search, and retrieve data with ease. Creating such a platform lets us, the statisticians, influence "data practices". People share data and add documentation within the guidelines of SensorBase.org which supports good experimental design and helps future modeling efforts. Last but not least, it provides a base for data integrity and data quality.

References

[1] K. Chang, N. Yau, M. Hansen, and D. Estrin. Sensorbase.org—a centralized repository to slog sensor network data. In *DCOSS/EAWMS*, 2006.

[2] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *ACM SIGMOD '94*, pages 419–429, 1994.

[3] E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani. An online algorithm for segmenting time series. In *ICDM*, pages 289–296, 2001.

[4] NRSS. Numeric really simple syndication. http://www.nrss.org/.

[5] N. Ramanathan, E. Kohler, and D. Estrin. Towards a debugging system for sensor networks. *Int. J. Netw. Manag.*, 15(4):223–234, 2005.

[6] J. T. Robinson. The k-d-b-tree: a search structure for large multidimensional dynamic indexes. In *ACM SIGMOD '81*, pages 10–18, 1981.

[7] E. Tufte. *Beautiful Evidence*. Graphics Press LLC, Cheshire, CT, USA, 2006.