Dissemination and Presentation of High Resolution Air Pollution Data from Mobile Sensor Nodes

Will Hedgecock, Peter Volgyesi, Akos Ledeczi, and Xenofon Koutsoukos
Institute for Software Integrated Systems - Vanderbilt University
Nashville, TN, USA 37203
{will.hedgecock; peter.volgyesi; akos.ledeczi; xenofon.koutsoukos}@vanderbilt.edu

ABSTRACT

This paper presents the framework of a mobile air quality monitoring network, with an in-depth discussion of several new innovative techniques for web-based visualization. These techniques allow typical web users to access high-resolution pollution data gathered from a large number of vehicle-mounted mobile sensing devices coupled with highly-accurate static sensor data in an easy-to-use, intuitive interface. Additionally, this interface offers users a set of novel applications to promote health and pollution awareness, including a green trip planner, whereby users can plot routes between two locations based on a path of least exposure to specified pollutants, and an exposure estimator, which allows users to calculate previous levels of exposure to harmful pollutants based only on a single timed GPS track.

Categories and Subject Descriptors

H.5 [**Information Systems**]: Information Interfaces and Presentation

General Terms

Web Based Visualization, Mobile Data Networks

Keywords

Data visualization, pollution monitoring, wireless sensor networks

1. INTRODUCTION

In 2007, a collaborative research project was commissioned by Microsoft to design a system capable of providing real-time air quality information to the general public via its SensorMap online visualization interface. Work was begun on such a system with the initial steps toward its implementation being completed by late 2007 [14]. The reason behind the need for such a system comes from a serious lack of real-time pollution indicators in both the United States and worldwide.

The current method of air pollution monitoring in the United States includes sampling airborne pollutants hourly, averaging these pollutant concentrations together over a 24-hour period, and publishing this data as a single value known as the day's Air Quality Index (AQI) [5]. For people who desire a higher level of resolution, the Environmental Protection Agency (EPA), along with several other government agencies, runs a service called AIRNow which publishes the available hourly pollution data on a web site in both graphical and downloadable form. It should be noted, however, that there are only about 5,000 pollution monitoring stations throughout the country. This minimal number of sensors coupled with a sparse sensing schedule means that the AIRNow interface only provides an extremely low-resolution image of air quality. Since pollution is highly location dependent, there is insufficient data to accurately evaluate air quality within specific neighborhoods or at precise locations.

Implementing a mobile air quality monitoring network enables a detailed picture of air pollution to be constructed based on real-time data from mobile sensors over an entire populated area. The work carried out under the Microsoft Research Grant enabled us to demonstrate the feasibility of this approach and build five car-mounted pollution sensor prototypes, each with an onboard GPS receiver and gas sensors measuring O3, NO2, and CO [14]. The recorded data, however, is only as useful as the manner in which it is presented. This paper discusses a novel method for accessing large-scale pollution data using web-based visualization techniques and a set of innovative web-based applications to provide additional health-related services to the public.

2. SYSTEM ARCHITECTURE

2.1 Sensor Node Overview

In order to discuss current research, we must first introduce the building blocks of the mobile monitoring network, namely the vehicle-mounted sensing nodes, as shown in Figure 1 on the next page. Each of these nodes is able to collect pollution data corresponding to atmospheric O3, NO2, and CO levels in an autonomous fashion, as well as store this data offline or stream the data to a base station in real time. Location and time information is provided by an on-board 20-channel SiRF-III-based GPS module at a sampling rate of 1 Hz. Gas concentration levels are measured by three analog sensors whose readings, along with the temperature, relative humidity, a time stamp, and GPS data, are stored in a serial flash device, capable of holding 6 hours and 50 minutes of data without offloading the information.

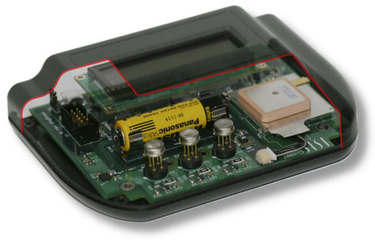


Figure 1: Sensor Node Prototype

The entire node runs at a software-defined frequency of 0.833 Hz, equivalent to 1 clock cycle every 1.2 seconds. At every cycle, the node reads the ambient temperature and relative humidity, polls the gas sensors, stores this information into flash memory, and offloads the data if possible. Each sample includes a maximum of 82 bytes of GPS data, all of the status information regarding the unit, as well as the actual atmospheric and pollution conditions, totaling 98 bytes of data. Thus, the total amount of pollution data to be stored, processed, retrieved, and visualized is equal to $4900 * NumActiveNodes$ bytes per minute.

2.2 System Overview

Although the sensor nodes are the physical building blocks of the mobile air quality monitoring network, many other components contribute data necessary to implement this architecture on a global system scale. The most notable of these components are the servers which are responsible for the processing, filtering, storage, and reconstruction of the information, as well as the clients who are interested in accessing the data. In addition to these components, the system uses external sources of information to add to its accuracy and robustness, such as area maps, local weather and traffic information, and measurements from the 5,000 EPA sensors [5]. The data from all of these components are used to reconstruct an overall pollution function for a given location over a given time window. This massive amount of data is then used by our servers to provide web clients with data, which is then displayed in an easy-to-use and intuitive interface, as described in the following sections.

It is important to note that the servers in this system constitute a server cloud, whereby server processes run on an interconnected distributed serving system. This increases the system's robustness and fault-tolerance while decreasing the bottleneck between potentially thousands to millions of clients accessing system resources through only one server or server bank. The server system is distributed in the sense that each server array is connected, directly or indirectly, to every other server array, such that data can be passed between servers in a manner similar to Internet routing [13].

The server system processes incoming measurements as atomic tasks through a series of phases. The inputs and outputs of each stage are stored so that failed steps can be rolled back. Only data tasks that successfully complete all phases are stored in each server's database. This approach avoids the cumbersome and resource-intensive process of removing faulty data. More abstract processing stages, such as context binding and interpolation, happen only after all relevant data are available. Waiting for complete data sets before processing reduces the cost of sophisticated stages and improves the overall quality of the generated data.

All processing stages are implemented as user-defined functions and stored procedures, developed in the C programming language. The overall data management system uses Microsoft SQL Server 2008, including its OpenGIS implementation. SQL Server is a good choice of data management software because it is a production-level Database Management System that offers high performance through clustering (the distribution of database information over several servers for redundancy and increased efficiency) [6]. Likewise, OpenGIS is an important database component, as it allows other developers and programmers to access the stored information in a widely-available, open-source format [9].

Finally, the clustering of data between servers occurs naturally based on geographic location. Several servers located throughout the country should primarily service the nodes and clients in their own sphere of influence. For example, all sensing units in Chicago transmit (using GSM modems) to the closest server bank to Chicago. This is possible since each device's on-board GPS unit is able to pinpoint its position and choose the correct server address from an internal database. Likewise, web clients accessing the project web site from Chicago are routed to the closest server bank. Finding a user's location based on IP address is a well-defined process with many implementations already available, such as MaxMind's GeoLite City [10]. Thus, the traffic to and from each server will be balanced according to geographic location, lowering access and processing latencies. If, however, a user in Chicago wants to view sensing data from another region (New York, for example), the local server acts as a cluster of the distributed whole; thus, it can fetch the required information from the New York server bank (over the Internet or dedicated lines) and store it in a local cache in case it is needed again or until it expires.

2.3 Communication & Information Interfaces

The transfer of data between server and clients uses an open-source platform called APE, short for "AJAX Push Engine." The engine makes use of an epoll-driven HTTP server written in C to allow data exchange between over 100,000 users per server via a web browser, without reloading or relying on any external plug-ins other than Javascript, which comes standard in almost all graphical web browsers [1]. The advantage of using APE in this system is that it uses push technology instead of pull to deliver updates to subscribed clients as they appear on the server in real-time. This not only reduces the massive amount of network traffic required to simply request information every few seconds, but also ensures that data is only transmitted when necessary.

The client side of APE uses a Javascript framework to hook into its communications capabilities. It allows for new plug-in modules to be added to extend server capability should any unforeseen needs arise. In the context of the pollution monitoring network, a single persistent connection is made over APE to the geographically-closest server when a user logs onto the project web site. The user is then presented with a Google map to display sensing nodes and pollution data. The GPS coordinate bounds of the map's viewport are communicated to the APE server, where they are used to subscribe the user to any nodes or pollution information falling within those bounds. Once this connection is established, data only flows between server and client when a node's data has been updated or when the user changes the viewport and must subscribe to a new set of locations.

Since the APE framework uses a subscription method for client communications, a single node update could result in the update being propagated from a single server to numerous clients, similar to a data broadcast. Also, since SQL Server is being used as the database system to drive APE, users may request (query) subsets of all the available information. In this way, a user viewing a simple sensor map is able to subscribe exclusively to sensor node location information without any accompanying pollution data. This greatly decreases the amount of traffic being sent over the network. If the user wants to access the raw data coming from a sensor on the map, then our visualization tools allow them to select that sensor, thereby subscribing to its total data package. In this way, the only data that traverses the network is the data the user has an interest in viewing, making the client-server communications as efficient and resource-friendly as possible.

3. WEB-BASED VISUALIZATION

The most visible and integral part of the monitoring network for everyday users is the web interface used to access the vast quantity of data provided by the system in an intuitive and easy-to-use fashion. The main purpose of the web interface is to provide access to the reconstructed pollution signal. At the most basic level, the system is able to calculate a vector of contour points or a two dimensional array of data points for pollutants in an area at a specified resolution and time interval, calculate a vector of time instants when pollutants cross specified thresholds at some location, or return the time series of pollutants at a specified time resolution and location. On top of this basic set of queries, several web applications provide useful services to the end user, most notably web-based visualization.

3.1 GPS Smoothing

While GPS systems are extremely accurate when used in situations where the GPS receiver is moving at a high velocity, accuracy decreases substantially at slower receiver speeds, due to a lack of velocity information to aid in modeling and estimating the receiver position [3]. We have witnessed location measurements of 150 meters error when the receiver is stationary. Due to these inherent shortcomings, especially when stationary, coordinate readings are generated with highly varying degrees of accuracy. Since precise estimation of sensor location is essential in this system, it is important that these inconsistencies be handled gracefully to obtain the most accurate location estimation possible.

We have developed a new method of filtering and smoothing recorded GPS coordinates, especially in stationary and low-speed situations, based on a modified Cumulative Displacement Filter [4]. This filter relies on the assumption that the relative displacement between sets of coordinates are more accurate than the coordinates themselves in low-speed situations. Our modifications take advantage of the knowledge that GPS accuracy increases with speed, and abrupt changes in location, azimuth, and speed are unlikely.

In our implementation, a variable-length history of GPS data is stored, such that the actual (albeit slightly time-delayed) position can be extrapolated from current, previous, and future measurements when the speed of the device decreases past some threshold or upon detection of outliers. Essentially, the filter works by projecting positions onto a linear regression built from the normalized sum of displacements from each position in the history to the next. It should be noted that this filter cannot be used in real time, but instead requires a delay equal to the size of the filter history. This is acceptable since data processing is done offline on a server, and the history only requires three data points to achieve maximum smoothing results. In addition to estimating positions at low speeds or with noisy signals, we have added functionality to disallow changes in location when the device is likely stopped, judging by the fusion of inputs from an on-board MEMS accelerometer, the speed estimation of the node, and sporadic readings from the GPS unit. This greatly cleans up the signal at events such as traffic lights, stop signs, or weak signals when the unit reports extremely inaccurate positions. Figure 2 shows the result (blue) of applying this filter to a noisy GPS signal (green).

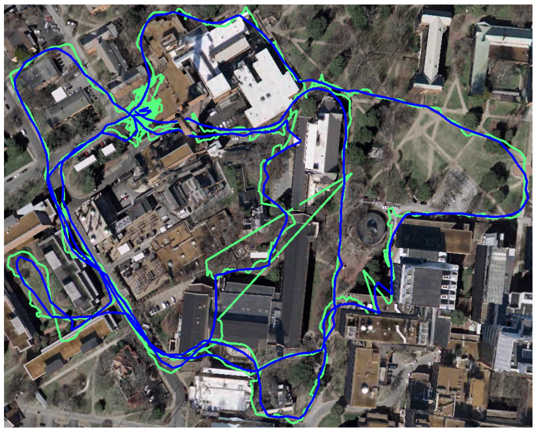
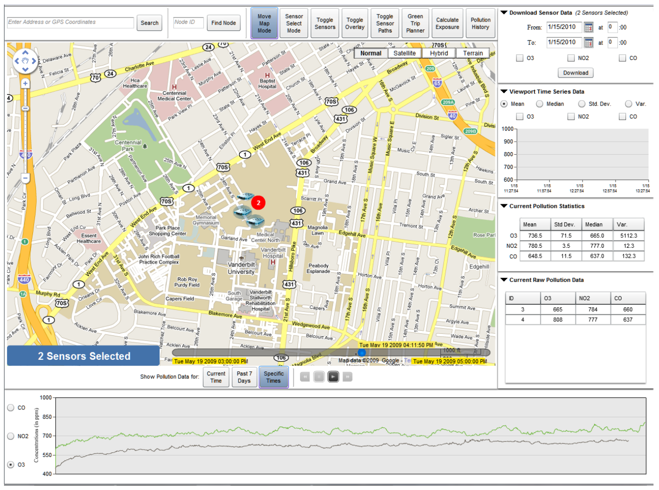


Figure 2: GPS Smoothing Results

Initially, a simple low-pass filter was tested to generate the aforementioned results. It was found, however, that a low-pass filter does not take into account typical GPS receiver behaviors, such as the possibility of losing satellite locks, having highly erroneous measurements followed instantaneously by accurate measurements, or even stopping and going in the reverse direction. As such, the smoothed results did not accurately reflect the ground truth, and oftentimes led to tracks that went off of roads and sometimes even into buildings. Our implementation has been found to give extremely accurate results with respect to a measured ground truth. In fact, the results as shown in Figure 2 so closely match the actual ground truth for that track that a graphical representation primarily shows only a single track.

3.2 Flash-Based Web Client

Access to the pollution information in our databases occurs mainly via an innovative new flash-based web client, which enables users to not only access raw sensor data, but also visualize relative pollution functions in a manner similar to Weather Channel radar maps [15], interact with the map in an intuitive fashion to retrieve desired data, use the data for several practical health-related applications, and gather targeted information about specific nodes or geographic areas over a variable time history. Our implementation of this client takes great pains to take into account the specific nature of the system to provide the most efficient, reliable, and robust interface possible. This section describes the various aspects of the web client, including its capabilities as well as the means by which it carries out its numerous tasks.



Download Sensor Data (2 Sensors Selected)

From: 1/15/2010 at 0 :00

To: 1/15/2010 at 0 :00

O3 NO2 CO

Download

Viewport Time Series Data

[✓ Mean] Median Std. Dev. Var.

O3 NO2 CO

Current Pollution Statistics

	Mean	Std Dev.	Median	Var.
O3	736.5	71.5	665.0	5112.3
NO2	780.5	3.5	777.0	12.3
CO	648.5	11.5	637.0	132.3

Current Raw Pollution Data

ID	O3	NO2	CO
3	665	784	660
4	808	777	637

Figure 3: Flash-Based Web Client

The main and most important aspect of the web-based client is the Google Map used to display sensor and pollutant information. Figure 3 shows the client as it appears when you first log onto the project web site. All of the information that can be displayed by the client is highly customizable, down to the type and resolution of map used for displaying the sensors and pollution information.

3.2.1 Marker Clustering

Once this system is deployed, it is likely that there will be many sensors located in close proximity to one another. Showing so many sensor icons at once will cause delays in the visualization system and look cluttered, making it difficult to see what is happening, let alone to select individual sensors. Likewise, when a user zooms out, the scale of the map increases and the sensors move closer and closer together. To remedy this problem, we use a marker clustering approach, embodied in a "Marker Clusterer" component that analyzes the positions of all sensors in the viewport, and if any markers are within 20 pixels of any number of other markers, they are combined into one "cluster marker" with the number of sensors contained in the cluster being displayed. To see the nodes contained in the cluster, the user can either move the cursor over the node where a text box will show the cluster contents, or zoom in such that the markers are sufficiently far apart to be displayed individually.

All sensor markers are completely managed by the Marker Clusterer in this system, including the markers for single sensors. If the clustering algorithm deems the marker to be sufficiently far away from other markers, it takes care of adding the individual icon to the map. In addition to adding to the user-friendliness and visual aesthetics of the client, the marker clustering implementation also provides a better, more efficient way to manage the potentially vast number of markers present on the entire map. Instead of simply adding a sensor marker everywhere a physical sensor is located, the Marker Clusterer keeps track of all of the markers available to be drawn, but only adds the ones to the map that would currently be visible given the geographic bounds of the viewing region. This keeps the Google Maps system from being bogged down by too much data, and also allows for 1000s of markers within the viewing area to be displayed with little to no noticeable delay to the user.

Users can select sensors by clicking the individual markers, the cluster markers, or in a special "Marker Select Mode" by selecting all markers in a region indicated by a cursor-drawn box. Selecting a cluster marker effectively selects all the individual nodes within the cluster. When one or more nodes are selected, a blue box appears at the bottom left corner of the map indicating the number of selected sensors.

3.2.2 Sensor Data

Data from the viewport and the selected sensors is visible in the sidebar on the right side of the client. This is the most important part of the client for viewing and manipulating data and statistics. Each sub-panel in the sidebar is collapsible and expandable such that the user must only view interesting or relevant data. The top-most panel is used to download raw data from the selected sensors. All that is required to use this function is a selection of some number of sensors, the sensing modalities for which the user would like data, and the starting and ending times and dates of data to retrieve. A database request query is sent to a server over the persistent APE connection, asking for data from the selected nodes (using their node IDs) with additional filter parameters being the start and end dates specified. The data is returned in the form of a SQL query response and downloaded to separate XML files for each selected node.

The next sub-panel is dependent upon the current viewing area, not the selected sensors. It displays statistical time series data over the past two hours for the entire viewing area. The user selects which statistic and modalities they are interested in, and a SQL query is sent over the APE connection, requesting sensor information for the past two hours, filtered by the geographic bounds of the current viewport. The requested statistic is calculated from the returned data, and the graph displays the information, updating itself automatically whenever the viewport changes. So that the data is kept fresh, but network traffic is minimized, the graph caches its current data and updates it every 30 minutes.

The third sub-panel shows the current statistical information for all sensing modalities for the range of selected sensors. This includes the mean, median, variance, and standard deviation for each pollutant. This is an extremely informational yet resource-economical function as it only requires the most recent sensor data from each selected node.

Finally, the last sub-panel shows the current raw data being reported by each selected sensor. When a user selects a sensor, it is added to this list, and the raw O3, NO2, and CO levels are displayed in the table. Likewise, when a user deselects a sensor, it is removed from this list. It should be noted that all of these sidebar sub-panels are updated dynamically in realtime as new data arrives. Thus, the user can be assured that any of the data or statistics being shown represent the most current data available to the system.

3.2.3 Sensor Node Visibility and History

The actual map area is able to show a variety of overlays depending on the interests of the user. By default, all sensor node locations are displayed upon loading of the web client; however, other overlays require manual instantiation. All available overlays may be toggled on or off via their respective buttons on the top bar of the client. Currently, two other overlays are available: a contour pollution overlay and a node path history overlay. The contour pollution overlay is a visualization of the pollution function within the viewport, which when fully implemented, will look similar to a Weather Channel radar map [15]. When viewing this type of overlay, only one modality can be shown at a time, due to the colorful, contour nature of the overlay which makes it impossible to differentiate between multiple modalities. These pollution maps will be generated locally by each individual web client using sparse data from the servers to feed a pollution model reconstruction function.

The node path history overlay provides a way to visualize where a sensor has traveled over the past two hours. It can show history information for all sensors in the viewport or only for selected sensors. In either case, the precise locations of the chosen sensors will be shown for the past two hours as different color line traces on the map. This history data is loaded via a simple SQL query to the web server, and the resulting coordinates are fed into a "Polyline" interface in the Google Maps API to produce the desired tracks.

3.2.4 Realtime Raw Data Graph

The very bottom area of the web client displays a graph which is used to show the raw data for selected sensors in graphical form. The user selects a desired sensing modality, and the past two hours of history for each of the selected sensors will appear on the graph. As time progresses and newer information becomes available, the graph continually updates itself, scrolling to the left such that there are always two hours of data showing with the right-most data point being the most current for each of the selected sensors.

It is clear that each of the components in the web client reuses information from other areas, manipulating the data as necessary for the specified functionality. A great strength of the client is its ability to send large requests for data once, then visualize the data in numerous ways from graphical to historical to statistical, while still remaining responsive and efficient. This is a primary goal of such a visualization framework, to present the available data in a variety of useful formats without overwhelming the user, the system, or the resources required for such data-intensive processing.

3.2.5 Data Retrieval

The greatest power and flexibility of this web client comes from the fact that all of the above functions can be used not only for current, realtime data, but also for the most recent 7-day span of history or any user-defined span of times and dates in the past. History functionality works similarly to a video player. When the user selects any mode other than "Current Time," a set of player controls appears at the bottom of the map. These controls work like any standard video player controls. The labels at the bottom left and right of the timeline show the start and end times and dates, and the label above the tracking bead shows the current time and date being displayed by the web client. Pressing the play button causes the tracking bead to move to the right, playing the history in real time. If the user wants to see the history played in an accelerated fashion, they can press the "faster" button to increase playback speed an unlimited number of times. Conversely, if they want to slow the play speed back to normal, they can press the "slower" button until the play speed has returned to realtime. It should be noted that playback cannot occur at speeds slower than realtime. If the user wants to directly skip ahead to a time, they can drag the bead forward or backward to reach the desired time and playback will resume from there. Finally, to stop playback, the "stop" button can be pressed. Pressing play after stopping playback continues to play the history from the time when the stop button was pressed.

The player works by making the web client think that the current time is something other than it is. In other words, the entire web-client system thinks it is actually operating at some user-defined date and time, and carries out normal functionality under that assumption. As such, playback is just as efficient as realtime streaming of data since it does not rely on changes to the underlying code. The only thing different from realtime streaming is that the sensor locations for all visible nodes over the entire history are loaded such that the user can skip to different times on the timeline without having to wait for data to buffer. Actual pollution concentration data is streamed in realtime as usual because the latency required for this is minimal, and it can be done without creating lags or delays in system playback.

Whenever the viewing area is changed by the user, the history data reloads based on the new bounds and playback continues; thus, the amount of memory resources required is minimized by only storing the data currently visible in the viewport (exactly the same way that the Marker Clusterer only displays markers currently in the viewport). All client functionality is retained in any of these modes, including sensor node path histories and sidebar functions such as raw sensor data and current pollution statistics.

3.2.6 Additional Applications

In addition to the functionality described previously, the client provides several innovative applications for use by the general public and the health-conscious. The framework for these applications has been created, but their implementation depends on the pollution model reconstruction algorithm which has yet to be finalized. Thus, while the following applications are implemented, they assume the presence of such reconstruction techniques, and are not yet functional.

The first additional application available to users in this web-based client is a "Green Trip Planner." This application provides the ability to plan driving routes between points based on a path of least exposure to pollutants. It has been designed to be extremely flexible, allowing users to compromise between pollution exposure and travel time for a happy medium. The user is prompted to enter a starting and ending address and the pollutants they want to avoid. The planner uses the current pollution model to define areas to avoid between the two locations, leaving available only those areas with acceptable pollutant levels. It then uses the Google Maps API "driving directions" function to find a path though those acceptable areas. The directions are displayed to the user along with a track on the map. If the user decides that the route is too long or inconvenient, they can click at various places on the map to add locations they explicitly want to drive through (i.e. an interstate between two points). The planner will then amend the directions to include the new points while retaining routes through as many of the pollution-safe zones as possible. The user can iterate through this process repeatedly until they are satisfied with the results, at which point, they will be given the option to print the directions or save them to a file.

The second additional application is a "Past Exposure Estimator", which enables users to estimate past exposure to pollutants given a timed GPS track as input. The estimator application maps each of the input GPS coordinates to a location in the pollution function for the given time. The output will be an XML file, whose contents are defined by the user and can contain pollution concentration information for each coordinate, a list of times when the user experienced pollution levels over a self-defined threshold level, or average concentration levels over specified time ranges. Finally, our server databases have hooks programmed into them to allow other users and developers to access our data at a low-level for use in their own applications and extensions.

4. RELATED WORK

Pollution awareness has increased substantially in recent years, motivating several projects that use mobile platforms to collect air quality measurements [7, 11, 2, 12]. One of the primary differences between these earlier works and ours is that the earlier mobile sensors were not networked, but rather relied on the manual download of measurements [7, 11]. The PEIR project from UCLA uses cell phones to collect air quality data. Our system places air pollution sensors on vehicles. This decision allows us to not only use more accurate sensors, but also to measure a wider spectrum of air quality parameters such as particulate matter, without having to worry about the resource constraints that make cell phone based approaches impractical. Additionally, our system is better-suited for pollution estimation since vehicles spend their time outdoors covering a wide geographic area. Cell phones tend to stay in a person's pocket indoors where pollution levels may deviate greatly from ambient levels.

Both the Common Sense project from Intel Research and a project by a conglomeration of universities in the United Kingdom are working on a similar platform to ours, creating a pollution sensing network using a variety of mobile sensing platforms including "smart-dust," mobile motes, static sensors, and cell phones [8]. The system outlined in this document, however, takes the approach to a higher level. We are not only developing algorithms for the reconstruction of air pollution information from irregular and sparse measurements, but also devising methods of visualization such as contour maps as opposed to simple discrete observations. In addition, our system implements practical applications which provide typical system users with valuable and accessible data that promises to impact everyday life, not only the needs of air pollution specialists or experts.

5. CONCLUSIONS

The overarching goal of this project is to dramatically increase the resolution of air pollution information and maximize its impact on public life. We have designed mobile nodes to sense three known air pollutants, O3, NO2, and CO, as well as ambient environmental conditions and communicate this data to a central server for the purpose of providing continuous realtime data feeds over a web interface. The system provides an intuitive method of data retrieval using web-based visualization with a number of novel applications making use of the high-resolution data. Users have the ability to not only download raw sensor data from any number of mobile sensing devices, but also to stream pollution information in real time, visualize this data in easy-to-understand contour-like pollution maps, and gather statistical information about pollutants over a specified spatiotemporal region. In addition to being able to do this with current realtime data, users can also access historical pollution information using the same interface in a manner similar to a standard video player, allowing researchers to perform historical analyses using our data.

The highest level of access to our pollution information comprises two novel applications, one of which can be used to estimate an individual's past exposure to a pollutant using only a single timed GPS track, and another which provides a route planning service to minimize a person's exposure to a given set of pollutants. Although some aspects of this project are still in research and development, we have outlined a solid framework for implementing a mobile air pollution monitoring network with concrete applications in the real world. It is now up to the public to utilize the resources we have provided to not only benefit their own personal level of health, but also to increase overall awareness of the societal impact of air pollution, so that together, we can work to promote a cleaner, safer environment.

6. ACKNOWLEDGMENTS

This work was made possible by a generous grant from Microsoft, NSF award 0807464, and the continuing financial support of Vanderbilt University.

7. REFERENCES

[1] APE: AJAX Push Engine. Available at `http://www.ape-project.org/`, 2009.

[2] Aoki, P.M. et al. Common sense: Mobile environmental sensing platforms. In *Adjunct Proceedings, Ubicomp 2008*, September 2008.

[3] Arden, D. MEMS/GPS Kalman Filter. Defense Research and Development Canada. May 2007.

[4] Cayzac, J. The cumulative displacement filter. Available at `http://julien.cayzac.name/code/gps/`, 2006.

[5] EPA. Air pollution data sources. Available at `http://www.epa.gov/air/airpolldata.html`, May 2009.

[6] Ester, M. et al. A database interface for clustering. In *Proceedings of the AAAI's KDD 1995*, 1995.

[7] Ghanem, M. et al. Sensor grids for air pollution monitoring. In *Proc. of 3rd UK e-Science All Hands Meeting*, 2004.

[8] Imperial College et al. Message project: Mobile environmental sensing system. Available at `http://bioinf.ncl.ac.uk/message/`, 2008.

[9] Intl. Open Geospatial Consortium. Opengis standards. Available at `http://www.opengeospatial.org/standards`, 2009.

[10] MaxMind. Geolite city. Available at `http://www.maxmind.com/app/geolitecity`, 2009.

[11] Rudman, P. et al. In *Proc. of UK-UbiNet workshop on eScience and ubicomp*, 2005.

[12] UCLA CENS. Personal environmental impact report. Available at `http://peir.cens.ucla.edu/`, 2009.

[13] Vaquero, L.M. et al. A break in the clouds. *ACM SIGCOMM CCR*, 39(1):50–Ű55, Jan.

[14] Volgyesi, P. et al. Air quality monitoring with sensormap. *IPSN*, pages 529–530, 2008.

[15] The Weather Channel, Interactive Map. Available at `http://www.weather.com/weather/map/interactive/`, 1995-2009.