NoizCrowd: A Crowd-Based Data Gathering and Management System for Noise Level Data

Mariusz Wisniewski[1], Gianluca Demartini[1], Apostolos Malatras[2], and Philippe Cudré-Mauroux[1]

[1] eXascale Infolab, University of Fribourg—Switzerland

[2] Pervasive and Artificial Intelligence Group, University of Fribourg—Switzerland

{`firstname.lastname`}`@unifr.ch`

Abstract. Many systems require access to very large amounts of data to properly function, like systems allowing to visualize or predict meteorological changes in a country over a given period of time, or any other system holding, processing and displaying scientific or sensor data. However, filling out a database with large amounts of valuable data can be a difficult, costly and time-consuming task. In this paper, we present techniques to create large amounts of data by combining crowdsourcing, data generation models, mobile computing, and big data analytics. We have implemented our methods in a system, NoizCrowd, allowing to crowdsource noise levels in a given region and to generate noise models by using state-of-the-art noise propagation models and array data management techniques. The resulting models and data can then be accessed using a visual interface.

1 Introduction

Big data has become a key element to support decision making at different levels of granularity, for example on continuous streams of data generated by sensors spread around a given environment (e.g., meteorological stations in cities). Analytics on sensor data is of high importance to support many critical decisions, such as when stopping the car traffic on certain days to limit pollution. However, sensors are expensive to install and maintain and need to be placed at specific geographical locations, which are typically application and context-dependent. Acquiring, deploying, and maintaining a large-scale sensing infrastructure for a given problem is hence difficult, costly and time-consuming.

In this paper we propose NoizCrowd: a crowd-sensing approach to big data generation using commodity sensors. The proposed system, built on top of a database management system for data-intensive applications, is able to collect streams of data from participating users' mobile devices and generate additional data thanks to statistical models based on the crowd input. NoizCrowd also provides dynamic visualizations of both sensor and model data to support user decision queries, for instance to determine regions with low noise levels in a given neighborhood to build a house or buy an apartment.

In more detail, NoizCrowd consists of four different components: i) a mobile application allowing the crowd to measure noise levels accurately using commodity smartphones ii) a scalable array storage layer to manage all the pieces of data gathered from the smartphones iii) a higher-level modeling layer capable of generating continuous models from the (potentially) sparse data gathered by the crowd and iv) a data export and visualization layer to interface with the end-users and support interactive decision making.

The proposed approach has the following advantages: i) it allows any mobile user to report the noise level at any place and time ii) it generates and manages large-scale data about noise levels in order to cover different regions and time intervals and iii) it supports users by answering data analytics requests on noise levels.

The rest of the paper is structured as follows. We start below in Section 2 by comparing NoizCrowd with previous work including sensor data generation applications running smartphones and noise mapping applications. In Section 3, we describe the architecture components of NoizCrowd in more detail, including the mobile application used to gather sensor data as well as the scalable storage system. We present in Section 4 the crowdsourcing model used to obtain noise-level data from geographically distributed users, as well as the data generation models used to integrate and extend potentially sparse user data. Section 5 presents the results of an experimental evaluation of the proposed system based on several live deployments. Finally, Section 6 concludes the paper by highlighting the main findings of our project.

2 Related Work

The notion of participatory sensing [2], namely user-centric monitoring and sensing of environmental conditions by means of high-end mobile phones, has recently emerged as a promising, low-cost alternative to traditional large-scale, costly and difficult to manage sensing infrastructures based on sensor networks [6]. Whereas there are several potential shortcomings for such an approach, especially in terms of privacy [4] and quality of data collection [11], its benefits nonetheless are far from negligible. In particular, attributed to the ubiquity of users with mobile devices equipped with sensors, participatory sensing applications can provide great data collection services at high granularity (spatial and temporal) and with a low cost [14]. Such applications, spurred by active research and development efforts in the domains of pervasive computing and the Internet of Things, are inherently distributed and lacking centralized infrastructures and therefore ensure robust operation and minimal management needs [3].

Participatory sensing applications have been considered for a wide range of sensed information, with typical examples including environmental impact [19], green vehicle routing [10], bargain shopping [7], etc. NoizCrowd focuses instead on the collection of urban noise levels using participatory sensing applications and their spatio-temporal representation on maps. This has generated great interest in the research community for a variety of reasons, the most important of which is the ubiquity of sound sensors, i.e. microphones, in current mobile devices. Moreover, urban noise levels are becoming increasingly important due to the related health concerns [18], as well as the associated regulatory frameworks and citizen concerns [1]. It is noteworthy that during an experiment regarding citizen engagement based on participatory sensing applications [8], people identified noise pollution as one of the most prominent information that should be monitored using applications like NoizCrowd.

NoiseTube [9] is one of the most interesting participatory noise mapping applications. It has been made available to the public through an open-source license and this has led to its widespread usage leading to contributions regarding noise levels for more than 250 locations around the world. NoiseTube allows users to annotate their data readings with social tags, thus allowing for semantic analysis of the collected data, while it additionally provides powerful signal processing techniques on the mobile phones to process received sound levels with a high degree of accuracy. NoiseMap [24] is an application built on the same principles as NoiseTube, with the distinctive characteristics of allowing users control over the collected data, while in parallel supporting real-time representation of user submitted data. Another application that maps noise levels in a city exploiting participatory sensing readings is SoundOfTheCity [23], in which semantic tags are also used to annotate noise readings. The SoundSense framework [15] also collects audio data from the user's vicinity, but instead of submitting raw data to the centralized participatory application it instead utilizes lightweight yet powerful machine learning algorithms to classify sound-related events that it then depicts on a map. Amongst other similar systems, we distinguish the NoiseSPY [12] application that maps noise levels using users' smartphones as sensors; the latter pre-processes the data prior to making it available, by calculating noise levels according to specific guidelines. This ensures more homogeneous data being represented on the map, namely classes of different noise levels, as well as reduced communication overhead in terms of information reporting.

As compared to existing systems, the advantages of NoizCrowd are twofold: NoizCrowd is capable of producing noise data in RDF format. This enables new kind of semantic applications by linking the generated data with other datasets in the Linked Open Data cloud[3]. Furthermore, the use of a scalable storage system together with noise propagation models to generate missing data guarantee better coverage of user requests both in time and space.

Regarding the collected data, there are a lot of issues concerning their granularity, i.e., lack of data for particular locations, which can compromise the operation of such participatory sensing applications. The Ear-Phone participatory noise mapping system [21] has a functional behavior similar to that of the previous applications and aims to address the problem of incomplete data in the context of noise collection. In this respect, it employs compressive sensing techniques and data projection models to enhance and account for the aforementioned problem. The simulation and real-world deployment evaluation results regarding data optimization are quite promising. In the same line of thinking,

[3] `http://linkeddata.org`

Mendez *et al.* present several data interpolation methods to improve the quality of incomplete and random data in participatory sensing applications targeting environmental monitoring [17]. With the same optimization goal, the DrOPS system that was recently presented in [20] utilizes model-driven approaches and online learning mechanisms to predict missing data readings from existing ones. We diverge from this work by focusing on noise data collection and their particularities, i.e. sound dissipation. Lastly, one of the biggest concerns in participatory sensing is ensuring that users are actively contributing and sharing their data [22], since it could eventually lead to poor performance due to the lack of accurate and informative representations. This problem has also been considered in the context of noise mapping, where a persuasive, motivating game was considered in [16] to stimulate user data collection and sharing.

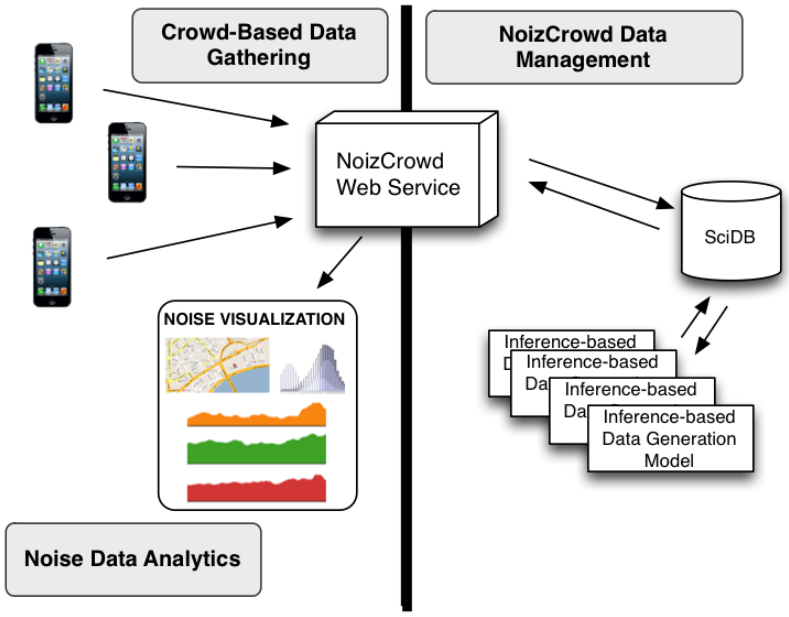
3 Architecture

As stated above, NoizCrowd consists of a scalable system allowing the crowd to accurately measure noise levels using their mobile phones, an array-based storage component for data-intensive applications, a model generation and management layer, as well as a data export and a visualization interface to support decision making. The overall system architecture is depicted in Figure 1. The rest of this section presents each of those four components in more detail.

3.1 Data Gathering

For a noise level mapping system such as NoizCrowd, gathering vast amounts of data over wide areas is key. Deploying noise sensors over a large geographical area would introduce substantial deployment and maintenance costs. Our approach to solve this problems is to use crowdsourcing. Indeed, nowadays many people use smartphones with an integrated global positioning system (GPS), a microphone as well as an Internet connection. These are the elements needed to record noise levels, measure the coordinates of the location where it was measured, and send such data to the NoizCrowd back-end server.

Since smartphones were typically not designed as noise meters, the first problem we face is to map the sound levels one can capture with his or her smartphones to standards decibels. This requires a third-party sound level meter to calibrate the microphone of the smartphone. Since such devices are relatively uncommon, we decided to crowdsource this problem too; For each smartphone model, any user from the crowd can enter through our application a conversion table linking the noise levels as recorded by his or her device to real decibels as recorded by a sound level meter. Such conversion tables are then shared to all other participants having a similar smartphone through our application. We apply a majority-vote algorithm to select the most popular values in those conversion tables when several participants register different values for a given sound level.



Crowd-Based Data Gathering
NoizCrowd Data Management
NoizCrowd Web Service
SciDB
NOISE VISUALIZATION
Inference-based Data Generation Model
Noise Data Analytics

Fig. 1. NoizCrowd architecture diagram: participants can share noise level data in their surroundings using a smartphone application; All measurements are then durably stored in an array database back-end; users can query the system using a visual interface, which triggers the generation of high-level interpolation models based on the raw data.

The main goal of our smartphone application is to allow participants to determine and share the sound levels in their surroundings. Our application, which is currently available for the iOS platform, allows the users to record the noise level at their current time and location by leveraging the embedded microphone and the iOS SDK audio recording functionalities. It returns the average noise level in decibels thanks to the crowdsourced conversion tables described above. Next, the application connects to the data back-end (see below) through a Web service and transmits the median and peaks of the measured values over few seconds.

3.2 Data Storage

We designed a new scalable back-end to be able to durably store and efficiently process all data points shared by the users. Since the data we are working with is highly non-relational (we are dealing with multidimensional data in space and time mostly in the context of this project), we decided to base our storage components on SciDB [5], a new array-based open-source database system for data-intensive applications. The back-end of NoizCrowd is responsible for three main tasks: receiving and durably storing all the data points shared by the users, providing all the required data to build higher-level models, and transforming and exporting relevant data for visualization purposes.

We store all value points shared by the user in multidimensional arrays. Two of the array dimensions represent spatial information (latitude and longitude), discretized using a fixed grid size[4]. The third array dimension represents an unbounded temporal axis, also discretized every hour in order to regroup temporally close measurements into one single value. All array values received by the crowd are durably stored in the database; the cell values are versioned [25] such that if a new value is received for a given array cell within one hour, it is stored as new versions of the older value.

This potentially extremely large array is stored as a compressed and sparse matrix on disk (i.e., only those cells that contain an actual value are materialized on disk). The array is chunked (i.e., split) both in time and space, and the resulting array chunks are stored on clusters of commodity machines if very large amounts of data are received from the crowd (see [5] for details on chunking and distributed storage in SciDB).

This array database layer is used to efficiently build higher-level data models (see below Section 3.3) by selecting slices of values in space and time. It is used in a similar manner by the data export and visualization component to extract relevant data and present them to the end-user to support aggregated information visualization and decision making.

3.3 Noise Modeling

The noise level values captured by the crowd are intrinsically sparse and noisy. We cannot expect all areas and time periods to be covered by end-users. Realistically, the data gathered will hence be highly skewed, with urban areas receiving way more data points than remote areas for instance. Also, the values collected by the users are inherently noisy, because of potential hardware differences between the smartphones (e.g., slightly different microphones used for a given smartphone model), and the high variability of ambient noise levels in urban areas.

To tackle both data sparsity and data fluctuations, NoizCrowd does not provide raw measurements to the end-users but builds instead higher-layer models from the data gathered. The modeling component that integrates overlapping values and generates missing data by interpolating crowdsourced observations is described in more detail in Section 4, while our interpolation models are experimentally validated in 5.

[4] Typically, we use a grid size of 10m given the accuracy of current smartphone positioning systems.

3.4 Data Export and Visualization

The data gathered from the crowd as well as the data generated by means of interpolation can then be used by end-users, either as an RDF export, or through a data visualization layer.

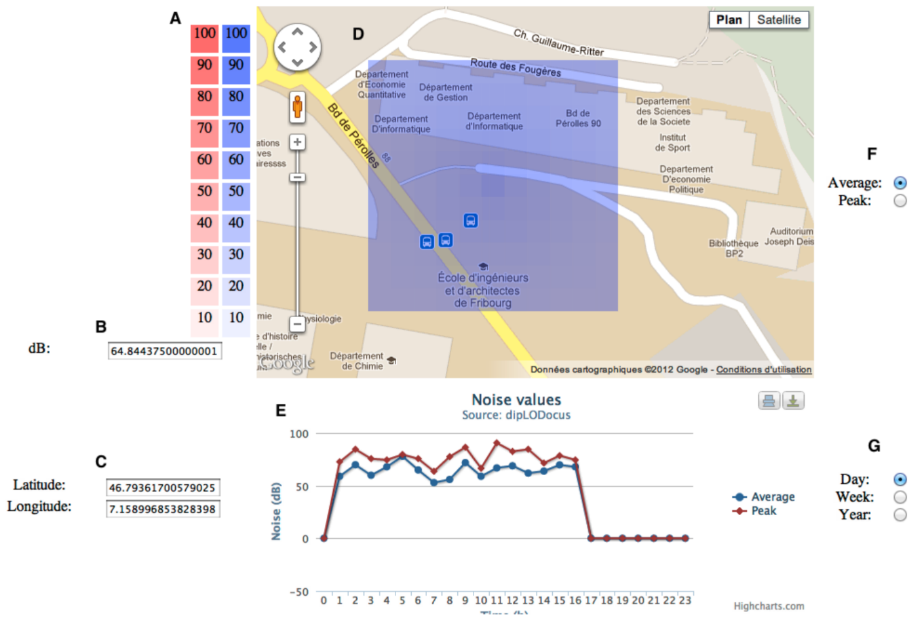
The export component regularly pulls data from the models and converts them into RDF triples for export. We use our own dipLODocus[RDF] system [26] to compactly store all data and to expose a SPARQL interface to whomever is interested in querying some of the exported data.

The visualization interface allows users to retrieve a representation of the noise levels of a region over a map. In the visualization component, the region we display is represented as a grid of squares with sides of 10 meters, hence directly matching the data stored in the array back-end and processed by our models. The temporal information (i.e., different noise levels at different points in time) is shown as an additional chart, as illustrated in Figure 2). An alternative visualization of NoizCrowd data may be obtained by adding a third axis for time and have our data model represented as a cube with latitude, longitude and time.

We use the Google Map API to display additional information at the location where the noise levels are visualized. This allows us to overlay noise levels on top of the map of a given region. Each time the user uses the zoom or moves the map, the model component is queried for new values corresponding to the displayed area. The opacity of our noise level overlay indicates the noise level, while the color indicates if the displayed values are the average sound level or the peak sound level, since the interface gives the users the choice of which value should be displayed. Right now each square is represented on the map, and if we zoom out too far, the overlays stop being displayed. We chose not to display overlays in that case because of the important load on the database when all squares of a big spatial region are queried and because of the prohibitive cost of rendering a multitude of small squares on the client side.

We use Highcharts[5] to generate the graphs. Highcharts is a charting library written in JavaScript, offering an easy way of adding interactive charts to Web sites or Web applications. The graph is used to display statistics of a given area. The user can select what type of statistics he is interested in and then retrieves the statistics of an area by clicking on it on the map. Right now the interface allows the users to get the average of all the measurements of a given area for every hour of a chosen day, to show the average of all the measurements of the area for every day of the week, and to show the average of all the measurements of the area for every month of the year.

Both average and peak values are always displayed on the graph simultaneously using the same colors as for the map (red for peak values and blue for average values). The axes of the chart change according to the options chosen and the values returned by the database management system. A future improve-



dB: 64.84437500000001

Latitude: 46.79361700579025
Longitude: 7.158996853828398

Average: [✓] Peak:

Day: [✓] Week: Year:

Fig. 2. NoizCrowd Visualization Interface.

ment would be to allow the users to select a whole region of the map and to get detailed statistics over it.

4 Models

As compared to well positioned sensors, the use of crowdsourcing does not guarantee an even coverage of the monitored geographical area over time. Thus, there will be missing data points both on the time and space dimensions. Since NoizCrowd needs to be able to answer queries about any location and time interval, our system adopts data generation methods by means of interpolation over different dimensions. In addition, the use of crowdsourcing can add fluctuations to the measurements as described above, and hence require models to integrate the measurements taken by different participants. In the following, we present the various models we have developed to tackle these issues in the context of NoizCrowd.

4.1 Spatial Interpolation Model

NoizCrowd uses an interpolation model to generate missing data between two measurements that have been provided by the crowd and stored in the database

[5] `http://www.highcharts.com/`

over the same time interval. In order to obtain valuable interpolations we need to carefully select different measurements based on their geographical distance. Note that we do not consider temporal interpolation at this point and focus on space solely in the following. To restrict the inference to adjacent measurements only, NoizCrowd defines a maximal distance threshold on the distance between two measurements used for inference[6]. In order to maintain the computational complexity of the inference tractable given the large-scale data NoizCrowd potentially needs to handle, we defined computationally simple interpolation models specifically tailored for our array database back-end. User queries can be generalized in our case as *slab* queries, i.e., bi-dimensional range queries in space: $q = \{x_i, x_j, y_i, y_j\}$. Once such a query is received, the back-end proceeds to a full-scan of the array values overlapping the slab (such slab scans are extremely optimized in SciDB, since the data is store in a compressed, vertical and sparse format using multidimensional chunks). Once all the values stored in the database for the given query are selected, we interpolate the data and compute the noise level for every cell value v in the slab by using a k-nearest neighbor interpolation based on the Manhattan distance:

$$v_{i,j} = MD \sum_{k=1}^{n} v_k md^{-1}(v_{i,j}, v_k)$$

where md represents the Manhattan distance between two cells and MD is a normalization factor ($MD = 1/\sum_{k=1}^{n} md^{-1}(v_{i,j}, v_k)$). Figure 3 shows an example of our spatial interpolation with $k = 2$, i.e, interpolating values given their two nearest-neighbors only.

The above interpolation can be executed very efficiently and in a scalable manner in our array backend, by keeping a small index on the available values in main memory and by executing a parallel swipe over all the missing values and computing each new value in parallel.

4.2 Temporal Interpolation Models

We experimented with two different time interpolation models. For short time ranges (e.g., time intervals lasting a few minutes), we extend the spatial model above to take into account spatiotemporal slab queries and Manhattan distances in three dimensions. In order to infer data over longer unobserved time intervals (e.g., hours or days), NoizCrowd adopts inference models that look for common patterns in the available data. We focus on finding repetitive patterns based on hour and day intervals in that context. For example, if every Monday at 11 a.m. the noise level for a given area is around 50dB and if we did not get any measurement last Monday, then we can assume that there is a high probability that the missing value is also 50dB. As another example, if we had a value for a given area two hours ago and another similar one for the same area right now,

[6] In the current version of NoizCrowd the maximum distance has been set to 50 meters.

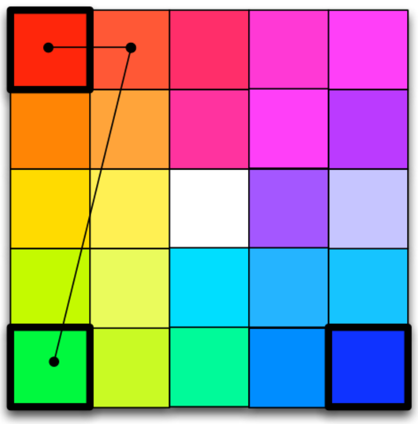


Fig. 3. Noise signal interpolation using two nearest-neighbors and three noise sources. The noise value of each region is computed as the average noise level of the two nearest noise sources.

then we can assume that doing an interpolation over those two values might lead to reasonable results. On the other hand, NoizCrowd does not perform interpolation for such cases when the values are too dissimilar (i.e., when for longer time intervals $|v_{t1} - v_{t2}| > \tau$, where τ is a system constant).

4.3 Noise Propagation Models

In addition to the interpolation models described above, we have created a model implementing the formula for the propagation of sound in real atmosphere based on Lamancusa's model [13]. Such complex models can be particularly useful in case we know the location of a sound source or want to locate a particular source given some measurement. The sound pressure level Lp is derived in this model as follows:

$$Lp = Lw - 20log(r) - 11 + DI - Aabs - Ae$$

Where Lw stands for the sound power level, r is the distance from the source in meters, DI is a directivity index, $Aabs$ is the atmospheric absorption, and Ae is the excess attenuation.

In free space, the directivity index value is 0dB while it is 3dB on a perfectly reflecting surface. Our models take the case of a perfectly reflecting surface into consideration. The atmospheric absorption is the energy dissipated in the air by viscous loss and relaxational processes over some distance. While the concept is easy to understand, it is difficult to measure such parameters using smartphones. However, we can set this variable as a constant if the region we take our measures in is known a priori.

The excess attenuation variable is defined as follows:

$$Ae = Aweather + Aground + Aturbulence + Abarrier + Avegetation$$

where *Aweather* represents the meteorological conditions including the effects of wind and temperature. In our case, we only take the temperature into account (since we can get it directly using smartphone sensors or by looking up values given the GPS coordinates and time). *Aground* represents the ground interaction and *Aturbulence* the atmospheric turbulence. We take standard values for both parameters due to the difficulty of measuring/looking up such parameters in our context. *Abarrier* and *Avegetation* represent obstructions and vegetation. We do not take these parameters into account.

By using the formula above, our model can generates accurate data values for the area surrounding a noise source. Locating the source of some sound by using three or more measurements in decibels can also be performed using the sound propagation in atmosphere formula. For each of the measurement we give to the model, we check each nearby cell and measure the value it would have if it were the source. The square that has the closest resulting values for each measurement is selected as the most likely source. We experiment with such models in Section 5.

4.4 Models & Late Materialization

As described above, we store data in two different ways: We store raw measurements in our SciDB backend, and higher-level, cleaned and interpolated data using higher-level models. As for traditional view mechanisms in database systems, we thus have to choose how we materialize (i.e., precompute and store) the model data. Since some of the models described above can be computationally intensive, and since the total time and geographical areas covered by the system can be very large, we decided to opt for late materialization strategies. The materialization of the models is thus performed at query-time: NoizCrowd only generates model data when a request is sent by the visualization interface about a specific spatiotemporal range. The resulting data is then cached and indexed as a new array in our backend, and can also be cached for future requests.

The model views are stored using a dedicated storage space in SciDB and are replaced using a scalable clock-replacement policy. When new data is inserted into the system, all views overlapping with the new data are selected. Two outcomes can then occur depending on the view: If the view can be updated (i.e., for the spatial and temporal models described above), then our backend updates the view incrementally by inserting the new data only and recomputes parts of the values only. It the view cannot be updated incrementally (i.e., for the noise propagation model), then the view is dropped and will have to be materialized again for future queries.

5 Performance Evaluation

In order to test the validity of our models, we performed series of live deployments using smartphones and mobile sound sources. We report below on a few of such experiments.

5.1 Spatial Interpolation

We tested our interpolation model through 30 outdoor deployments, 10 times using 2 smartphones simultaneously, 10 times with 3 smartphones, and 10 times with 4 smartphones. The location of the smartphones in the deployments were randomly selected in a given flat region of 50 by 50 meters. We chose a relatively busy neighborhood in an urban setting with relatively constant noise levels. We took a professional-grade noise level meter to measure the real values of the sound levels we were trying to interpolate using the smartphones available and our model.

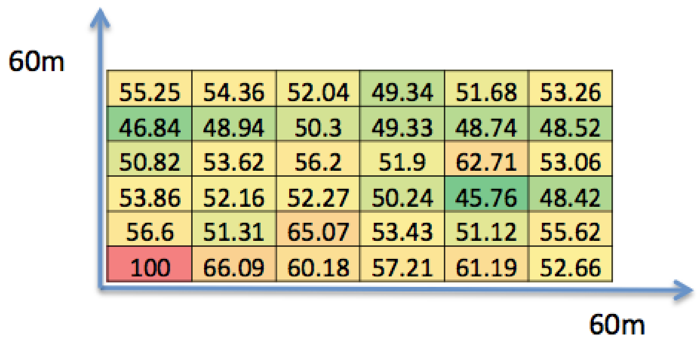
In summary, the results we obtained were as follows: 85% of the resulting interpolated data was with an error of less than 6dB to the real values, with 63% of those values within 4dB or less. We find those results very encouraging given the setting chosen and the few values recorded.

5.2 Sound Dissipation and Source Location

We used a controlled setting to test our sound dissipation and source localization approaches. We picked a 100dB sound source on a 60 by 60 meter baseball field, covered in snow to avoid reverberation. We placed the source in one corner of the field, and then measured the sound levels in the neighboring area using our application and three smartphones. Figure 4 shows the results. Normally, the values should steadily decrease as we get further and further from the source. However, as we can observe on the figure, this is not strictly the case in practice, given that the measurements were taken over several minutes in an open area using different smartphones.

We then tested our sound propagation and source localization models given this relatively noisy input (which is in our opinion close to what the crowd would give us in a larger-scale deployment). We performed 10 tests using 3 random measurements picked from our live deployment, 10 using 4 measurements, and 10 others using 5 measurements.

In summary, the results we obtained are as follows: the error on the sound level value of the source determined by sound dissipation and our smartphone measurements decreases steadily with the number of measurements available: on average, the error is 16% for 3 available measurements, 10% with 4 measurements, and 9% with 5 measurements. The sound localization performed well too, as we were able to locate the source within a 3 meter radius on average.



55.25	54.36	52.04	49.34	51.68	53.26
46.84	48.94	50.3	49.33	48.74	48.52
50.82	53.62	56.2	51.9	62.71	53.06
53.86	52.16	52.27	50.24	45.76	48.42
56.6	51.31	65.07	53.43	51.12	55.62
100	66.09	60.18	57.21	61.19	52.66

Fig. 4. NoizCrowd live measurements in space using three smartphones and a noise source of 100 db (lower left of the figure).

6 Conclusions

Generation and management of large-scale data is key for data exploration and decision making. In this paper, we have proposed NoizCrowd: a system to crowd-source noise level data using smartphones. Our system is able to scale out the gathering and management of noise level signals by means of a mobile application and a scalable array back-end. In addition, NoizCrowd autonomously generates missing data by means of interpolation over time and space. Finally, a visual dashboard allows users to query for the noise levels in a specific time and space interval. Our experimentations using real-world deployments of our system have shown that the proposed interpolation models can accurately generate noise level data with noisy input values.

As future work, in addition to porting our application to different mobile platforms (e.g., Android), we plan to extend it with additional functionalities to incentive people to use it. Indeed, using an application just to measure and share the surrounding noise brings little incentives on its own. Therefore, the data recording should be processed in the background while the participant uses the application for other purposes.

Finally, since computing models dynamically at query-time for any user query can be expensive, we plan to explore materialization strategies in more detail in the future. More specifically, we plan to materialize models at various granularities in an offline manner in order to let the users freely zoom in or out on a maps in real-time.

7 Acknowledgments

This work was supported (in part) by the Swiss National Science Foundation under grant numbers PP00P2_128459 (Infrastructures for Community-Based Data Management) and 200021_130132 (BioMPE).

References

1. Jess Barreiro-Hurl, Mercedes Sanchez, and Montserrat Viladrich-Grau. How much are people willing to pay for silence? a contingent valuation study. *Applied Economics*, 37(11):1233–1246, 2005.
2. J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *In: Workshop on World-Sensor-Web (WSW06): Mobile Device Centric Sensor Networks and Applications*, pages 117–134, 2006.
3. A.T. Campbell, S.B. Eisenman, N.D. Lane, E. Miluzzo, R.A. Peterson, Hong Lu, Xiao Zheng, M. Musolesi, K. Fodor, and Gahng-Seop Ahn. The rise of people-centric sensing. *Internet Computing, IEEE*, 12(4):12–21, 2008.
4. Delphine Christin, Andreas Reinhardt, Salil S. Kanhere, and Matthias Hollick. A survey on privacy in mobile participatory sensing applications. *Journal of Systems and Software*, 84(11):1928 – 1946, 2011.
5. Philippe Cudré-Mauroux, Hideaki Kimura, Kian-Tat Lim, Jennie Rogers, Roman Simakov, Emad Soroush, Pavel Velikhov, Daniel L. Wang, Magdalena Balazinska, Jacek Becla, David J. DeWitt, Bobbi Heath, David Maier, Samuel Madden, Jignesh M. Patel, Michael Stonebraker, and Stanley B. Zdonik. A Demonstration of SciDB: A Science-Oriented DBMS. *PVLDB*, 2(2):1534–1537, 2009.
6. Dana Cuff, Mark Hansen, and Jerry Kang. Urban sensing: out of the woods. *ACM Communications*, 51(3):24–33, March 2008.
7. Linda Deng and Landon P. Cox. Livecompare: grocery bargain hunting through participatory sensing. In *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, HotMobile '09, pages 4:1–4:6, New York, NY, USA, 2009. ACM.
8. Carl DiSalvo, Illah Nourbakhsh, David Holstius, Ayça Akin, and Marti Louw. The neighborhood networks project: a case study of critical engagement and creative expression through participatory design. In *Proceedings of the Tenth Anniversary Conference on Participatory Design 2008*, PDC '08, pages 41–50, Indianapolis, IN, USA, 2008. Indiana University.
9. Ellie DHondt, Matthias Stevens, and An Jacobs. Participatory noise mapping works! an evaluation of participatory sensing as an alternative to standard techniques for environmental monitoring. *Pervasive and Mobile Computing*, (0):–, 2012.
10. Raghu K. Ganti, Nam Pham, Hossein Ahmadi, Saurabh Nangia, and Tarek F. Abdelzaher. Greengps: a participatory sensing fuel-efficient maps application. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, pages 151–164, New York, NY, USA, 2010. ACM.
11. R.K. Ganti, Fan Ye, and Hui Lei. Mobile crowdsensing: current state and future challenges. *Communications Magazine, IEEE*, 49(11):32–39, 2011.
12. Eiman Kanjo. Noisespy: A real-time mobile phone platform for urban noise monitoring and mapping. *Mob. Netw. Appl.*, 15(4):562–574, August 2010.
13. J. S. Lamancusa. Outdoor sound propagation. PA: Penn State University. pp. 10.610.7.
14. N.D. Lane, E. Miluzzo, Hong Lu, D. Peebles, T. Choudhury, and A.T. Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, 2010.
15. Hong Lu, Wei Pan, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. Soundsense: scalable sound sensing for people-centric applications on mobile

phones. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, MobiSys '09, pages 165–178, New York, NY, USA, 2009. ACM.

16. Irene Garcia Martí, Luis E. Rodríguez, Mauricia Benedito, Sergi Trilles, Arturo Beltràn, Laura Díaz, and Joaquín Huerta. Mobile application for noise pollution monitoring through gamification techniques. In Marc Herrlich, Rainer Malaka, and Maic Masuch, editors, *Entertainment Computing - ICEC 2012*, volume 7522 of *Lecture Notes in Computer Science*, pages 562–571. Springer Berlin Heidelberg, 2012.
17. D. Mendez, M. Labrador, and K. Ramachandran. Data interpolation for participatory sensing systems. *Pervasive and Mobile Computing*, 9(1):132 – 148, 2013. Special Section: Pervasive Sustainability.
18. Anne Vernez Moudon. Real noise from the urban environment: How ambient community noise affects health and what can be done about it. *American Journal of Preventive Medicine*, 37(2):167 – 171, 2009.
19. Min Mun, Sasank Reddy, Katie Shilton, Nathan Yau, Jeff Burke, Deborah Estrin, Mark Hansen, Eric Howard, Ruth West, and Péter Boda. Peir, the personal environmental impact report, as a platform for participatory sensing systems research. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, MobiSys '09, pages 55–68, New York, NY, USA, 2009. ACM.
20. Damian Philipp, Jaroslaw Stachowiak, Patrick Alt, Frank Dürr, and Kurt Rothermel. DrOPS: Model-Driven Optimization for Public Sensing Systems. In *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom) (PerCom 2013)*, pages 1–8, San Diego, CA, USA, March 2013. IEEE Computer Society.
21. Rajib Kumar Rana, Chun Tung Chou, Salil S. Kanhere, Nirupama Bulusu, and Wen Hu. Ear-phone: an end-to-end participatory urban noise mapping system. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '10, pages 105–116, New York, NY, USA, 2010. ACM.
22. Sasank Reddy, Deborah Estrin, and Mani Srivastava. Recruitment framework for participatory sensing data collections. In Patrik Floren, Antonio Krger, and Mirjana Spasojevic, editors, *Pervasive Computing*, volume 6030 of *Lecture Notes in Computer Science*, pages 138–155. Springer Berlin Heidelberg, 2010.
23. Lukas Ruge, Bashar Altakrouri, and Andreas Schrader. Soundofthecity - continuous noise monitoring for a healthy city. In *5th International Workshop on Smart Environments and Ambient Intelligence (SENAmI 2013) at IEEE International Conference on Pervasive Computing and Communication (PerCom 2013)*, San Diego, California, USA, March 18-22 2013.
24. Immanuel Schweizer, Roman Bärtl, Axel Schulz, Florian Probst, and Max Mühlhäuser. Noisemap - real-time participatory noise maps. In *Second International Workshop on Sensing Applications on Mobile Phones, ACM SenSys 2011*, 2011.
25. Adam Seering, Philippe Cudré-Mauroux, Samuel Madden, and Michael Stonebraker. Efficient Versioning for Scientific Array Databases. In *ICDE*, pages 1013–1024. IEEE Computer Society, 2012.
26. Marcin Wylot, Jigé Pont, Mariusz Wisniewski, and Philippe Cudré-Mauroux. dipLODocus[RDF] - Short and Long-Tail RDF Analytics for Massive Webs of Data. In *International Semantic Web Conference*, pages 778–793, 2011.