Geospatial Data Aggregation and Reduction in Vehicular Sensing Applications: the Case of Road Surface Monitoring

Valerio Freschi, Saverio Delpiori, Lorenz Cuno Klopfenstein, Emanuele Lattanzi, Gioele Luchetti, Alessandro Bogliolo
DiSBeF, University of Urbino, Urbino, Italy
Email: {valerio.freschi, saverio.delpriori, lorenz.klopfenstein, emanuele.lattanzi, gioele.luchetti, alessandro.bogliolo}@uniurb.it

Abstract—Mobile devices present several features which make them attractive as enabling technology for crowdsensing systems. In particular, their spectrum of sensing capabilities, together with consolidated diffusion and ease of use contribute to an increasing adoption in different mobility-based sensing scenarios. On the other hand, the availability of massive volumes of geospatial data provided by large-scale distributed sensing systems prompts the need for innovative approaches to efficient data gathering and processing. Data reduction strategies are often necessary in order to cope with challenges posed by these volumes, for instance when dealing with real-time visualization of query results. In this paper we present a data reduction and aggregation approach for mitigating the impact of data size in a vehicular sensing application aimed at monitoring the roughness of road surfaces. Data collected by smartphones on board of vehicles is progressively thinned at different levels of the proposed architecture through sampling and spatial/temporal aggregation. Preliminary results show that the proposed methodology provides substantial benefits in terms of reduced impact of data while, at the same time, it enables full exploitation of statistical error compensation.

I. INTRODUCTION

The increasing diffusion of mobile embedded sensing devices with wireless communication capabilities opens the way to unprecedented opportunities in the development of large scale crowdsensing systems [1]. Current smartphones, in particular, are commonly equipped with sensors that can continuously monitor several physical quantities (e.g. acceleration). This provides, in combination with location coordinates available through GPS or other localization systems, a rich source of geo-referenced information. Moreover, the pervasiveness of these devices makes them the enabling technology for designing large scale mobile distributed systems aimed at massive sensing, either volunteer or incentive-based [2].

Needless to say, this perspective also poses significant challenges to the research community in order to build systems capable of efficiently and accurately collecting, processing and making available this wealth of data, ranging from system architecture to algorithmic design, from communication protocols to database. An additional dimension is represented by the massive nature of geo-referenced data to be handled by these hardware/software systems, especially when vehicular sensing applications are foreseen, which entail further peculiar issues to be addressed [3]. As such, a common feature of several research studies in recent scientific literature is represented by efforts made towards effectively scaling systems for storage and analysis of big geospatial sensing data.

In this paper we introduce a system for reduction and aggregation of geospatial data in vehicle-based monitoring applications. In particular, we describe a novel approach to manage data produced in a crowdsensing application for road surface quality control by means of spatial and temporal aggregation techniques. We demonstrate that the proposed distributed architecture is suitable to reduce the burden of large scale geo-referenced data volumes produced by sensing devices mounted on common smartphones. This system architecture provides the opportunity for fine grained data gathering and batch processing while, at the same time, it enables effective visualization and real-time analysis. Progressive spatial and temporal aggregation of data is performed at different levels of the proposed architecture (from user mobile devices on vehicles, up to the cloud) resulting into a significant reduction (w.r.t. raw data logged from smartphones) of the amount of data to be analyzed and visualized.

The paper is organized as follows: in Section II we summarize the state of the art of related scientific literature. In Section III we describe the proposed crowdsensing architecture and data aggregation strategy. In Section IV we discuss experimental results. In Section V we draw final conclusions.

II. RELATED WORK

A huge body of literature has flourished in the last decade around the vast field of mobile sensing information systems. In this section we try to summarize which are, in our opinion, some of the main trends related to the topics which are the scope of this paper.

Crowdsensing is an increasingly popular paradigm for gathering significant amounts of data from active communities of users (i.e. participatory sensing) or agents opportunistically carrying on sensing tasks (i.e. opportunistic sensing) [1]. Data is usually sensed by mobile devices whose location can be tracked with a given precision so that useful geo-referenced information can be obtained and geographic information systems (GIS) can be exploited for analytics extraction. The ever increasing widespread diffusion of commodity smartphones and the availability of several sensors (e.g. accelerometers, GPS, ambient light, microphones, cameras, etc.) on board of them, make these devices the ideal candidate sensing platform for many large scale mobile monitoring tasks [1], [2].
A. Vehicle-based sensing system architectures

Eriksson et al. proposed in 2008 CarTel, a system for road surface monitoring focused on pothole detection by means of embedded accelerometers and GPS sensors mounted in cars equipped with embedded microprocessors [4]. Mohan et al. developed Nericell, a smartphone-based mobile sensing system aimed at detecting traffic conditions, bumps, and honking events by integrating audio and acceleration data from microphones and triaxial accelerometers mounted on smartphones [5]. Vtrack is a system that enables road traffic delay estimation using mobile phones, with emphasis on energy consumption and noise compensation [6]. A follow-up paper from the same research group described an approach to trajectory mapping from cellular GSM fingerprints instead of WiFi and GPS traces [7]. A prominent example of large-scale system based on mobile sensing is represented by OpenSense, a system aimed at monitoring air pollution by means of sensor stations deployed on public transport vehicles and through participative sensing from citizens equipped with ad hoc pocket sensors or enhanced smartphones [8]. A system for road surface collaborative monitoring, called SmartRoadSense, has been recently introduced [9]. SmartRoadSense is a mobile/cloud architecture designed for continuous monitoring of road surface quality conditions, estimated by means of a roughness index computed on board of smartphones and stored/processed/visualized in cloud.

B. Big geospatial data analysis

Mobile crowdsensing inherently implies dealing with expected large volumes of data that prompt for efficient and scalable solutions both at system and at algorithmic level. The growing research field of the so called spatial BigData mainly refers to the development of novel methodologies and approaches to address all issues related to geospatial massive datasets. Within this framework, some recent works highlighted the need for new flexible approaches and, at the same time, pointed out the inadequacy of more traditional approaches rooted in database research [3], [10]. Moreover, while modern database management systems routinely face problems related to efficient storage, search, and processing of data, visualization systems need to be re-designed in order to keep pace with BigData. According to this perspective, Keller et al. introduced Vizly, a middleware designed for interactive browsing of large data sets in sensor networks applications which has been integrated in the OpenSense project framework [11]. Battle et al. stressed the lack of a thorough support of visualization systems to larger scales. In order to overcome some of the related challenges they proposed ScalaR, a system for dynamic resolution reduction to be applied when results of a query are expected to be too big to be handled by standard data base management systems (DBMS). Reduction is achieved through a chain of aggregation, sampling, and filtering operations [12].

III. PROPOSED ARCHITECTURE

This section presents the architecture of SmartRoadSense and the solutions adopted for data gathering, aggregation, reduction, and visualization. Scalability issues to be faced at each stage will be then discussed in next section.

The algorithmic pipeline is distributed at different levels of SmartRoadSense, a cloud-based system for collaborative road quality monitoring designed for estimating the surface roughness of roads by means of a smartphone’s triaxial accelerometers.

The architecture is based on three main components, schematically represented in Figure 1: i) a mobile application running on Android devices which reads the data provided by the embedded GPS and accelerometers and computes every second a geo-tagged estimate of the roughness of the road surface; ii) a server that gathers roughness indexes from all the smartphones running the SmartRoadSense application and makes use of OpenStreetMap [13] to perform spatio-temporal aggregation and reduction, and iii) a cloud-based front-end for graphical visualization.

Our approach to data reduction is obtained through different algorithmic strategies developed at different levels of this system. Figure 2 represents the various phases of the implemented algorithmic pipeline, labelled (a), (b), (c), (d) and (e). In the following we sketch the main tasks performed at each level of the pipeline.

- **Phase (a).** During phase (a), synthetic numerical values (called Roughness indexes, RI) are computed real-time on board of smartphones from accelerations sensed by the devices. These values, which provide a reasonable estimate of the roughness of the underlying road monitored by the vehicle, represent the result of a first level of data reduction. In fact, sampling sensed data according to GPS sampling capabilities and summarizing the information coming from three axes into a single numerical estimate value provides sizeable data compression w.r.t. raw data.

- **Phase (b).** This step consists of data serialization and storage of roughness indexes (with geographical coordinates and timestamp) on the memory of the smartphones. Batches of stored data are periodically transmitted to a remote server through GSM channels.

- **Phase (c).** This phase, implemented on the cloud, performs consistent spatial aggregation of points received by the back-end. Each point is mapped onto a map database and aggregated according to specific geometric constraints. This makes it possible to consistently map the sensed physical quantities of several adjacent points into a single aggregate, providing
further reduction of the number of points which are the final target of a visualization task and also smoothing outliers thanks to statistical compensation.

- **Phase (d).** Phase (d) regards temporal aggregation. Weighted average values of the monitored quantities are periodically computed, resulting into manifold benefits. The database is kept updated with last significant changes (incrementally down-weighting older points), data to be (also visually) analyzed is further thinned, and statistical robustness deriving from multiple measurements associated to a given location can be exploited.

- **Phase (e).** This last step entails the visualization of data (namely, the aggregated roughness index) provided as output by previous steps, by means of a graphical front-end.

In the next subsections, we further detail some aspects of the implementation of data reduction and aggregation algorithms, referring them to the three components of the SmartRoadSense architecture.

### A. Smartphone level

The first layer of the system architecture consists of an Android application which is in charge of gathering data from the smartphone’s sensors, namely GPS and triaxial accelerometers and implements phases (a) and (c) of the algorithmic pipeline of Figure 2. Since the sampling frequency of GPS mounted on current smartphones is much lower than that of triaxial accelerometers on the same devices (typically 1Hz and 100 Hz, respectively) the former represents a constraint on the spatial resolution that can be exploited for a first-cut reduction of the data to be collected. In fact, the developed mobile application works on windows of 100 samples (i.e. 100 seconds, taking the above mentioned sampling frequencies) and computes, for each window, an aggregated roughness index RI. RI represents the average value of the power of the prediction errors (named PPE’s) computed when a prediction filter is applied [9]. Prediction errors are computed for each time window and along each of the three axial components of the acceleration. Given the power of the prediction errors $PPE_x$, $PPE_y$, and $PPE_z$ computed by applying a Linear Predictive Coding algorithm (LPC for short) to the collected samples, the roughness of the road surface upon which the vehicle travels is estimated as their arithmetic average.

This estimate provides significant information on the quality of road surface, given the capability of the LPC algorithm of filtering out (up to a certain degree) spurious components of the acceleration signals (engine vibrations, gravitation, inertial forces, etc.). RI values represent a compact sketch that can be usefully exploited in a collaborative setting. In fact, the contribution of many roughness indexes can be taken into account to represent the quality of a given road in a specific geographical position, thus providing a worth of meaningful information that can be properly averaged. RI values annotated with a track identification code and with time and position references are stored in memory according to Java serialization format and periodically transmitted in batch to a remote server through GSM connection. Data payload is encoded in JSON and HTTP protocol is used for data transfer to the cloud.

### B. Cloud back-end level

A server application has been implemented (the YouSense server of Figure 1) that exposes a set of application program interfaces (in particular RESTful API’s) in order to allow permitted users to upload data. The collection back-end has been designed exploiting PostgreSQL with PostGIS extension (for geospatial processing) as database. This layer of the system architecture is in charge of the spatial and temporal data aggregation corresponding to phases (c) and (d) of Figure 2. First of all it makes use of a map matching algorithm in order to map each newly received point (composed of spatio-temporal coordinates, RI and metadata) to its closest road. Road cartography is provided by OpenStreetMap and map matching is currently implemented by associating points to geometrically closest road segments. Artifacts are removed by a simple post-processing that takes new data points sorted by timestamp. The list of these points is analyzed using a window of 3 points, $p_1$, $p_2$, $p_3$. If $p_1$ and $p_3$ are matched to the same road, while $p_2$ is associated to another road, $p_2$ is matched back to the road of $p_1$ and $p_3$, since it is assumed that the road change was misdetected. Needless to say, several alternatives could be taken into consideration to enhance the accuracy of mapping [6].

#### 1) Spatial aggregation

Spatial aggregation is obtained by uniformly sampling road segments matched by points during the map-matching phase. Given an input parameter (termed Spatial Sampling Factor, or SSF) we sample each road uniformly every SSF meters obtaining a set of landmark points.
We then track all points falling within a circle of given radius (called Coverage Circle Radius, CCR) centered around each landmark point. The average RI value to be associated to the landmark is then computed as the average value of RIs associated to points falling within the circle. Data values are weighted by their distance from the landmark point (i.e. from the center of the circle) using an inverse exponential function and annotated with their timestamp.

2) Temporal aggregation: Temporal aggregation (corresponding to phase (d) of the pipeline reported in Figure 2) is achieved by aggregating all roughness values computed for the same position on the same road. Values contributing to the same point are sorted by descending timestamp. The contribution of each roughness value decreases exponentially in time, thus the latest computed value has the highest weight, while older values are steeply down-weighted. This exponential decay is simply implemented by updating the temporal estimate (a daily estimate is a reasonable time horizon in road quality monitoring) as the average between the current value and the previous aggregated estimate (regardless of the days elapsed since last update). This corresponds to an exponential decay if new estimates are provided every day. If there are gaps, they are implicitly filled by assuming the daily value equals previous estimates.

C. Front-end level

The SmartRoadSense graphical front-end is based on CartoDB, a cloud service for visualization tasks of geographical maps and associated overlays [14]. The service offers web APIs that allow the back-end to upload updated roughness values for each geographic point. It also provides functions for retrieving a list of roughness points for all roads inside a given geographic area. This is used to populate a map of roughness points and to display it as an overlay to a geographical map (e.g. Google Maps [15]), thus implementing the functionalities associated to phase (e) of Figure 2. Each point is graphically displayed and filled according to a linear color map that represents the RI values (green to red, from lowest to highest), thus providing useful visual information about the roughness experienced by vehicles travelling along a given road segment.

IV. SCALABILITY ANALYSIS

A full-fledged prototype of SmartRoadSense was developed at the University of Urbino and systematically used for one month to monitor the roughness of the roads along a path of 275Km traveled by public transport twice a day. The selected path went through 744 roads of the OpenStreetMap DB adopted in the test bed. Buses were equipped with Android smartphones (namely, Motorola Moto G) running the mobile SmartRoadSense application. The roughness index values computed on board of mobile devices once per second were opportunistically transmitted to the server exploiting either Wi-Fi or m2m 3G connections when available. Connection attempts were automatically performed by the application every 15 minutes.

Spatial aggregation was performed by the server with a sampling step of 20m along the travelled roads, corresponding to a coverage circle radius (CCR) of 40m. Aggregated data were re-computed every day and stored as geo-tagged time series. Temporal aggregation was then performed at each sampling point to obtain a scalar value to be graphically represented on the map as the combination of all the elements of the roughness time series associated with that particular point.

Scalability is analyzed and discussed in the following section focusing on each single step of the data flow.

A. Data size

In order to investigate the impact of the different phases of the proposed data flow we studied how the size of the payload changes at each step, taking into account both the information content and the encoding adopted. Moreover, we analyzed the overhead introduced either by the protocols or by the data base management system for performance optimization.

Figure 3 summarizes the results of this analysis. The figure is divided into three conceptual sections, from left to right. On the left a bar plot is used to represent data size (split into payload, protocol overhead, and DBMS overhead) at each distinct phase starting from raw sensor data (at the bottom) up to CartoDB visualization (at the top). The effects of data serialization, JSON encoding, and HTTP transmission are also represented. Data size is expressed in bits per sample (bps), which corresponds to bits per second in the first 5 steps and to bits per point after spatial aggregation. The phases of the data flow, labelled (a), (b), (c), (d), and (e) as in Figure 2, are graphically represented in the middle, pointing out the element of the architecture involved at each step (smartphones, cloud, front-end) and the communication among them. Finally, multiplicative scale factors are reported on the right, using circles to denote the steps affected by each of them. The scale factors are: the absolute number of seconds of activity of the system (secs), the average number of simultaneous users collecting roughness indexes at each second (users), the absolute number of days of activity of the system (days) which corresponds to the length of time series associated with each sampling point, and the total length of the monitored streets (length) which is upper bounded by the total length of the streets of the underlying OpenStreetMap DB. White circles are used to mark phases which are not critically affected by scale factors in terms of performance. For instance, the number of users doesn’t impact processing steps carried on board of smartphones, since we have one device per user.

In the following we detail the results reported in the bar plot of Figure 3. For what concerns the size of the payload, it is worth noticing that it depends on the amount of information to be conveyed but also on the type of encoding, thus determining different space requirements even without processing steps.

1) Raw sensor data: Payload of raw sensor data is composed of: three values encoding the accelerations (32 bits each), two values encoding longitude and latitude (64 bits each) and two values representing other data (termed bearing and accuracy) to be possibly used for post-processing (each parameter being 32 bits long). This results into 9856 bits for each sample of data.

2) Roughness index: After on board-processing, the resulting roughness index needs only 320 bits per sample for its encoding because we need only to keep a single RI value instead of three acceleration values.
3) Java binary serialization: Since roughness indexes are written on the memory of mobile devices for batch transmission, we need to take into account the overhead due to this process (called Java binary serialization). In particular, the size of the payload is increased because of the encoding of additional required information (an ID number, start time, duration, etc.) amounting to 704 bits per sample, while the overall protocol accounts for a 6504 bits of overhead.

4) JSON over HTTP: In order to be sent over HTTP data packets are JSON encoded. The resulting payload is 1368 bits long, while the contribution of protocols to overhead is 1016 bits (JSON), 22 bits (HTTP) and 5 bits (JSON over HTTP): in total 1043 bits per sample.

5) PostgreSQL storage: When data is received on the cloud, it is stored in PostgreSQL format. While 1024 bits are sufficient for encoding the payload, the DBMS introduces a significant overhead of 4736 bits, that are to be added to 1760 bits needed by the indexing structures.

6) PostgreSQL processed: The impact of aggregation at this phase of the algorithmic pipeline is apparent, since the payload reduces to 512 bits for each sample, while the overhead decreases to 802 bits.

7) Final CartoDB data: Data needed by the visualization front-end consists of 320 bits (needed for encoding geometry and roughness index) of payload and remaining 2528 bits required by CartoDB as database overhead.

The effectiveness of data reduction and aggregation strategies is evident from the results reported in Figure 3, previously described. It is also worthwhile to notice that scalability analysis also clearly points out the impact of database management systems on the data size. For instance, CartoDB makes use of indexes in order to provide effective visualization support. Nonetheless, the adoption of these optimization structures leads to substantial increase of data dimension (up to 88.7% of the whole size).

V. Conclusion

The increasing diffusion of mobile devices with sensing and communication capabilities (i.e. smartphones) provides the opportunity for capillary crowdsensing applications. The enormous amount of data potentially produced in these settings raises the question of how to handle it for building efficient analytics frameworks which are the ultimate target of these systems. In this paper we introduced a data reduction and aggregation approach aimed at mitigating the impact of geospatial BigData in vehicular sensing applications. Our approach consists of a sequence of geometric sampling and temporal aggregation steps implemented at different levels of SmartRoadSense, a system architecture that supports road quality monitoring. Experimental results show that the proposed methodology is beneficial in terms of the reduced impact of data size in geospatial applications, while it provides full support to exploit statistical robustness in a massive distributed sensing environment.

REFERENCES


