CrODA-gator: An Open Access CrowdSourcing Platform as a Service

Michalis Massalas*, Andreas Konstantinidis*, Achilleas Achilleos*, Christos Markides* and George Papadopoulos†
*Department of Computer Science and Engineering, Frederick University, Nicosia, Cyprus
Email: {st009194,com.ca,com.aa,com.mc}@frederick.ac.cy
†Department of Computer Science, University of Cyprus, Nicosia, Cyprus
Email: george@cs.ucy.ac.cy

***Abstract*—The huge increase of mobile devices and the advancements of their sensing and computing capabilities have made the mobile crowd a real-time opportunistic data generator. Leveraging crowdsourced data creates new opportunities and challenges in many computing domains. As a result extensible, scalable and inter-operable cloud-based platforms have been implemented to simplify management and visual mapping of the large volume of data to meaningful representations, which can be then used for the development of novel applications. Still, to the authors best knowledge, these platforms do not offer direct open access to cloud-based crowdsourcing service(s). In this paper, *"CrODA-gator"*, an Open Access Crowdsourcing Platform as a Service, is introduced that follows a scalable and extensible architecture, which offers public open access to the platform's features for direct use by data contributors and application developers. This is a key attribute for the uptake of such a platform. Finally, an experimental evaluation is conducted to support the design choices, providing qualitative evidence on the expected performance of the platform's mechanisms.**

I. Introduction

According to global statistics the number of smartphone users currently reaches an estimate of over two billion worldwide[1]. Given that people spend most of their day with their personal mobile devices in their close vicinity and considering that most devices contain more than 20 sensors[2] (wireless signals, temperature, gravity, light, magnetic field, humidity, proximity and sound, just to name few), it is easy to imagine the large volume of data that can be potentially collected every day and used in numerous applications [3].

Therefore, the implementation of a scalable and extensible cloud-based Platform as a Service (PaaS) for collecting and leveraging crowdsourced data may provide tremendous opportunities with respect to new application development. For example, health organizations and insurance companies use temperature, humidity, air pressure and sound data combined with a person's common route to diagnose, prevent or calculate the risk of illnesses [1]. Retailers and advertisement companies use data derived from the accelerometer, gyroscope and microphone to analyse shopping behavioural patterns, or offer advertisements more effectively [4]. Moreover, the interactive entertainment industry has already started developing augmented reality games that utilize real data such as location, acceleration, orientation, light, humidity and temperature from the players actual environment to produce a more realistic game play [5].

The development of a platform, however, that will collect and aggregate opportunistic data generated from sensors and computations performed by the crowd's mobile devices automatically [6], give rise to several challenges that can be roughly classified into: (i) crowdsourcing related challenges that include outliers filtering without using intrinsic metrics and providing proper incentives to the crowd for contributing their data, (ii) data management and aggregation challenges that include the manipulation of a large volume of crowdsensed data and the avoidance of dataset distortion (through loss of data, due to the different number and variety of sensors in smart devices), and (iii) data mapping challenges that include designing an easy and efficient process for expressing data in a summarized, but still meaningful, form for individual or corporate use (e.g., for statistical purposes); and mapping data to geographic information systems (GIS), also known as Neogeography [7], in order to shape a context and conveying understanding through knowledge of places.

This paper presents an Open Access Crowdsourcing Data Aggregation platform, coined CrODA-gator. The platform provides the necessary mechanisms for collecting sensor data from heterogeneous smartphone devices, as well as aggregating and enabling those data for public use, respecting anonymity and data privacy by collecting non-sensitive data. In addition, it provides a web viewer that allows users to navigate through Google maps using numerous different filters. All data are available through well-documented open APIs that enable to use them for developing novel solutions and applications in a disperse area of domains, ranging from corporate to education and research. For instance, CrODA-gator may be utilized in the domain of education by allowing institutions to analyze crowdsensed spatiotemporal data (such as temperature, humidity, noise) and provide visual representations to their students for understanding concepts related to climate changes and global warming effect. Furthermore, the CrODA-gator follows a modular architecture that can easily extend and/or enhance its existing functionalities, and offers an effortless procedure for adding new modules (e.g.,

[1]"Smartphone Users Worldwide", Source: https://www.statista.com/.
[2]"Sensors Overview", Source: https://goo.gl/z8s8GK.

978-1-5386-3227-7/18/$31.00 ©2018 IEEE

adding new sensor types). For dealing with the potentially huge amount of data simultaneously, CrODA-gator uses a simple implementation of the Map-Reduce mechanism that allows easy data aggregation and mapping in GIS as well a mechanism for filtering out the outliers. In this way, the users are not overwhelmed with a huge number of raw sensor records but rather with a smoother layer of indicative data.

CroDA-gator's key contribution is the public crowdsourcing Platform as a Service, which provides open access to data contributors via the following features:

- An open access cloud-based platform that collects sensors data available from heterogeneous smart mobile devices, filtering out outliers and aggregating useful data.
- A number of well-documented APIs with public access and data formatting guidelines for (a continuously growing number of) supported sensors, to be used for contributing real-time and offline bulk data, as well as using these datasets for developing novel applications.
- A web viewer that optimally aggregates and maps spatio-temporal data in GIS and visually depicts the dataset representations along with a number of filtering options for developing new maps.

The rest of the paper is organized as follows: Section 2 overviews the related work, Section 3 presents the CrODA-gator architecture including its individual modules and mechanisms. Section 4 presents the experimental study and finally Section 5 concludes the paper.

II. Related Work

The smartphones' unique characteristics and features, such as their pervasiveness, ubiquitous availability and multi-sensing capabilities, have provided a new variety of efficient means for data collection enabling the so-called mobile crowdsourcing applications [6], [11]. Mobile crowdsourcing can be classified by whether the crowd's contribution is *participatory*, meaning that the input is computations performed by users and user generated data similar to how crowdsourcing services are used in the web, or *opportunistically* that includes data generated from sensors and computations performed by the crowd devices automatically [2]. The latter is also known as *Crowdsensing* [12] due to the procedural importance and extended use of the device's sensors.

Providing crowdsourcing as a service, which is making the collected dataset available to the public through the use of well-structured and implemented web APIs, is an incentive and reward for the community to support such services. Crowdsourcing platforms are already becoming popular in the computing community. Some platforms provide crowdsensing as a service [13], [14], but do not provide public open access to the service, to the best knowledge of the authors.

For example, Device Analyzer [8] is a data collection Android tool that collects various device information ranging from the operating system and the list of applications installed to periodic sensor readings (e.g., acceleration, air pressure, brightness) depending on the sensors availability on the device. OpenStreetMap (OSM) [10] is a knowledge collective that provides user-generated street maps. OSM is much more than a simple crowdsensing application since it provides a complete service with advanced functionality. For example, it offers a Google Maps style online mapping interface that maps and represents the collected data in a so-called OSM map as well as exporting possibilities in different vector formats for further use or processing. It also provides to contributors APIs for uploading newly collected data and correcting spotted errors. CrowdSignals.io[3] aims to create the largest set of rich, longitudinal mobile and sensor data recorded from smartphones and smartwatches. Finally, Anyplace [9] developed at the University of Cyprus is an open architecture that collects indoor information using crowdsourcing for providing users fine-grain indoor navigation services.

TABLE I
Taxonomy of Crowdsourcing Platforms and Applications

Platform	MPS	Open APIs	Dedicated Application	MSS	Bulk Data Import
Device Analyzer [8]	×	×	✓	✓	×
OpenStreetMaps [10]	✓	✓	×	×	×
CrowdSignals.io [3]	✓	×	×	✓	×
Anyplace [9]	✓	✓	✓ .	×	×
CrODA-gator	✓	✓	✓	✓	✓

MPS: Multi-Platform Support; MSS: Multi-Sensors Support

Table I summarizes key high level features of the various platforms and compares them with the proposed CrODA-gator PaaS. Device Analyzer is only an Android application that allows the collection of multiple device data. Anyplace and OSM, on the other hand, offer multi-platform compatibility, provide open and well-documented APIs and offer an online viewer that efficiently map and present the data, similarly to CrODA-gator. Both however, do not offer a dedicated application to allow contributors to enrich the existing datasets and manage the data. Instead, it requires the development of a native application to utilize the data via API calls. Moreover, Anyplace and OSM, rely only on particular device readings, such WiFi RSSI for Anyplace and GPS coordinates for OSM. Crowdsignals.io provides multi-sensors support, but it does not offer open APIs for data access and management. It rather merely offers a single sample free dataset along with code in Java and Python to aid developers, while requests for additional datasets access can be send. Finally, none of those platforms offer a bulk data import functionality.

Other well-known repositories with general datasets, including KDnuggets, DRYAD, Datahub and re3data are not primarily focused on hosting ubiquitous or small-device datasets, but some of their datasets may include small devices as well.

III. Overview of CrODA-gator

In this section, the platform architecture is presented. As illustrated also in Figure 1, this section starts with an overview of the user interface, then the data layer and finally the APIs and their documentation. This is followed by a more detailed

[3]"CrowdSignals DataSet and Platform", Source: http://crowdsignals.io.

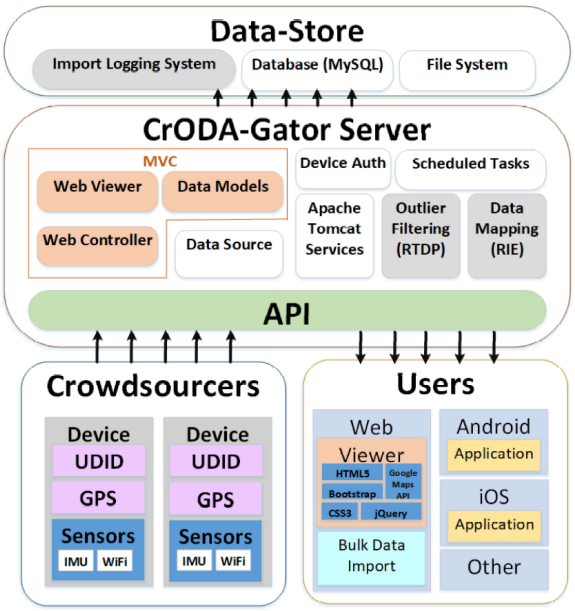


Fig. 1. **The CrODA-gator Architecture.**

description of the most important modules and algorithms as well as the CrODA-gator's dedicated smartphone application.

Web Server and Data Store: The CrODA-gator platform is currently deployed on Microsoft's Azure Cloud that offers scalability, security and extensibility. In particular, a Windows Server 2008 R2 is used with an AMD Opteron(tm) Processor 4171 HE 2.10 GHz, 1.75 GB RAM and a 64-bit Operating System. All web-based modules were developed using the Adobe's Coldfusion programming language that is built on top of Java and uses Apache Tomcat J2EE container, allowing for raw Java invocations and class reuse. The web server also hosts a conventional MySQL 5.5 database utilized for storing devices related data and all mobile crowdsensed data.

Web User Interface: The CrODA-gator's web interface that visually maps and represents the crowdsourced data in real-time adopts the asynchronous non server-side technologies jQuery and AJAX to handle the events, the documentation and map manipulation, as well as any other client-side activities offered by CrODA-gator's website. For the more complex and time-consuming tasks, several community tested and widely used plugins were used along with jQuery that include: (i) Simple upload (v1.0) that is an extremely simple yet powerful jQuery file upload plugin designed to be non-intrusive, backwards-compatible, flexible, and very easy to understand, (2) jqBootstrapValidation (v1.3.6) that is a JQuery validation plugin for validating bootstrap form fields, (iii) Bootstrap (v3.3.7) web framework for developing responsive, mobile first projects on the web and (iv) Font Awesome (v4.6.3) that provides scalable vector icons, which can instantly be customized via CSS. Moreover, the CrODA-gator follows the Model-Viewer-Controller (MVC) design pattern that offers code reusability, modularity and allows parallel development.

The web interface is mainly composed of three components, namely the Bulk Data Import, the Upload Logging and the Web Viewer:

- Bulk Data Import: allows users to upload files in json or csv format in the database, offline.
- Upload Logging System: keeps track of the Bulk Upload scheduled tasks operated and if any issues come up, it stores a message describing the issues in detail, for the administrators to handle them at their convenience.
- Web Viewer: provides access to all crowdsourced data stored in the database sorted by sensor/measurement type and view them in two different map layouts, as well as averages scaled from country level to small areas level.

API & Documentation: Open APIs are the most important component of the CrODA-gator framework, since they consist of a number of clearly defined methods used to communicate with any multi-platform application for either contributing or retrieving data from the CrODA-gator's database. All APIs were developed with a built-in verification process that checks the validity of a request before proceeding to execution. This avoids redundant processing on the server side and faulty data insertion in the database. The open APIs offered by CrODA-gator along with a well-defined documentation can be found in CrODA-gator's web site[4].

A. Android Application

The CrODA-gator application is currently developed on top of the ubiquitous Android OS and allows a user to select one or more data recordings to be either stored locally at the mobile device and then uploaded to the database using the Bulk Data Import component, or forwarded to the CrODA-gator's data store directly through the open APIs, in real-time. The application currently tracks various types of sensor data including noise, WiFi RSSi, pressure, mobile data, location data (GPS) accelerometer, compass and gyroscope data, but it can be easily expanded to include more sensor types. The user can start/stop the recordings at any time and can visually follow the recordings in real-time. Figure 2 shows some of the activities of the dedicated CrODA-gator Android application.

B. Plotting and Data Mapping Algorithm

The CrODA-gator's online viewer provides a visual representation of all data in two view-forms depending on the zoom level, as shown in Figure 3. For low zoom levels, the grid-view is adopted, which projects geographic coordinates to screen coordinates, draw grids representing few square meters area each and matches the geo-located crowdsourced data to each grid. For high zoom levels, the per country pin-view is adopted that shows a pin at the center of each country and shows the average of each data type for the whole country.

In particular, when the map is zoomed out to a zoom value greater than ten (10) in Google maps API, the per country pin-view appears showing the averages for the whole country only, as depicted in Figure 3 (a). If a country does not

[4]CrODA-gator: http://mdl.frederick.ac.cy, https://goo.gl/GVeUSX

Fig. 2. **The CrODA-gator Android application.**

contain any data, then a different pin appears to indicate the absence of data. To keep track of the averages of each country for each data type, an algorithm is used that calculates the average values each time a new measurement is contributed to the dataset. If, for instance, a mobile device contributes a new recording through the API (e.g. a Wi-Fi signal strength, temperature), then CrODA-gator initially checks and finds, using the Google reverse geo-coding API, to which country this recording belongs to and then recalculates the average for the whole country for that particular data type. Note that bad recordings are automatically discarded by the outlier filtering algorithm described below in Subsection III-C and they are not included in the calculations.

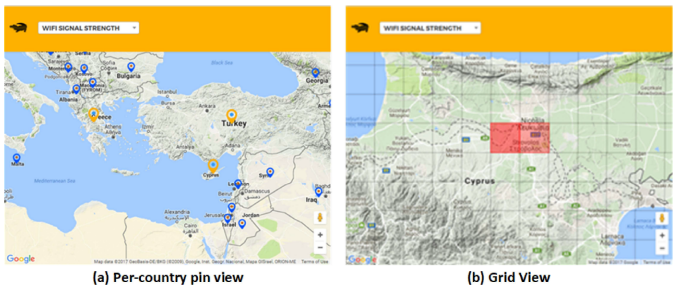


Fig. 3. **The CrODA-gator Viewer.**

For zoom levels below ten (10) in Google maps API, the grid-view appears that shows averages for small areas of few square meters, for each data type. This is achieved by initially retrieving the latitude and longitude of both the North-East and the South-West Google map corners through the Google maps Javascript API. These values are then forwarded to CrODA-gator's API along with the selected data type. Then the CrODA-gator retrieves all data sensed by the mobile crowd for that particular area. The data are then passed to the plotting function, which associates each data recording to the correct grid and finally finds the averages of all grids. The opacity value of each grid indicates the average value level of the selected data type as shown in Figure 3 (b). The higher the opacity the lower the average value and vice-versa. Algorithm 1 summarizes the *Recursive Indexing with Elimination (RIE)* plotting function just described.

Algorithm 1 Recursive Indexing with Elimination (RIE): Plotting and Data Mapping

Input: Current viewport bounds (North-East/South-West latitude and longitude)

Output: Grid layer with data averages on top of Google maps

1: hr $\rightarrow$ d(NE,NW)/zoomLevel(m) ▷ # of horizontal rectangles
2: vr $\rightarrow$ d(NE,SE)/zoomLevel(m) ▷ # of vertical rectangles
3: **while** $hri > 0$ || $vri > 0$ **do**
4: **for all** $rz \in R$ **do** ▷ loop Recordings
5: **if** $rz \in b(hri, vrj)$ **then** ▷ if recording in bounds
6: opacity $\rightarrow (m_{av} - m_{min})/(m_{max} - m_{min})$
7: $drawR(i, j)$ ▷ draw rectangle
8: $rm(rz, R)$ ▷ remove recording
9: **end if**
10: **end for**
11: **end while**

C. Outlier Filtering Algorithm

CrODA-gator depends heavily on data contributed by the crowd and the crowd's understanding and decisions depend heavily from the data integrity of the proposed service. Therefore, a simple, yet powerful, outlier filtering algorithm, named *Relative Threshold Divergence Purge (RTDP)*, was implemented that prevents users from distorting the dataset either accidentally or intentionally.

For each new data entry m_i^j retrieved by device i at a specific location (x_i, y_i) for sensor type j, the RTDP works as described in Algorithm 2. The m_i^j is initially compared with a pre-specified m_{min}^j and m_{max}^j values for sensor type j, as well as an average value m_{av}^j of all data falling within r KMs from (x_i, y_i). If the new data entry m_i^j is greater than m_{max}^j or less than m_{min}^j or diverges more than ϵ from m_{av}^j (e.g, $\epsilon = 33\%$) then m_i^j is treated as an outlier and is not inserted in the data store. Each time a new data entry contributed by device i is considered as an outlier then a counter o_i associated with i is increased. Finally, a device i is added in the banned devices list when $o_i \geq b$, where b is the maximum banned entries allowed for each device using the CrODA-gator PaaS. Since CrODA-gator promotes user anonymity and does not include any user

Algorithm 2 Relative Threshold Divergence Purge (RTDP): Outlier Filtering

Input: Data Entry m_i^j of device i for sensor type j at location (x_i, y_i); o_i: outlier counter of i; r: radius of circle with (x_i, y_i) centroid; ϵ: divergence threshold; b: banned devices threshold;
Output: Acknowledges Data Entry

```
for all m_i^j do                                   ▷ for all new entries
    m_av^j → avInRadius(j, (x_i, y_i), r)          ▷ average within r KMs radius
    if (m_min^j < m_i^j < m_max^j)&&|m_i^j − m_av^j < ϵ| then
        accept recording
    else
        reject recording
        o_i + +
    end if
    if o_i > b then
        add i in banned devices list
    end if
end for
```

registration phase, in order to uniquely identify the devices (and not the users), the Unique Device Identifier (UDID) of each device is retrieved each time an API is called. This allows the platform to keep track of contributed data validity and guarantee that users are sending (anonymously) genuine data.

IV. Experimental Evaluation

This section presents an evaluation in terms of the execution time of the platform's modules.

A. Methodology

In this study realistic datasets of different sizes and sensors were constructed, in order to evaluate the scalability and efficiency of the platform offline. In particular, the small data set includes 3000 data entries and its size is around 500KB and the large data set includes 50000 entries and its size is around 1.2 MB. Small and large data sets were generated for both WiFi RSSi and pressure data. The proposed outlier filtering and data mapping algorithms were compared with benchmark approaches in terms of their execution time, i.e., the individual time that an algorithm requires to accomplish its task.

The experimental study compares the proposed approaches with benchmark approaches for the following components:
(i) Outlier Filtering: the proposed *Relative Threshold Divergence Purge (RTDP)* described in Subsection III-C is compared with the *Borderline Outreach Purge (BOP)* approach that adopts only the first step of RTDP. Hence, BOP compares the new entries with pre-specified boundaries (i.e., minimum and maximum values for each sensor type) and if their value outreaches the boundaries then they are treated as outliers.
(ii) Plotting and Data Mapping: the proposed *Recursive Indexing with Elimination (RIE)* approach described in Subsection III-B is compared with the *Basic Recursive Indexing (BRI)* algorithm that starts from a predetermined grid (i.e., top leftmost corner as in RIE for consistency) and for each grid the measurement values contained in the result set are checked to determine if they belong to this grid or not. Then this grid is drawn on the map as an overlay depending on the average calculated value. The main difference between BRI and RIE algorithms is that BRI is not purging each value from the result set once its location is determined, thus keeping its size the same as the initial size and forcing each iteration to use abundant values during the scanning process.

All experimental results below show the averages of 100 consecutive runs, for fairness. The test instances used in the experimental study are summarized in Table II.

TABLE II
Test Instances (TI)

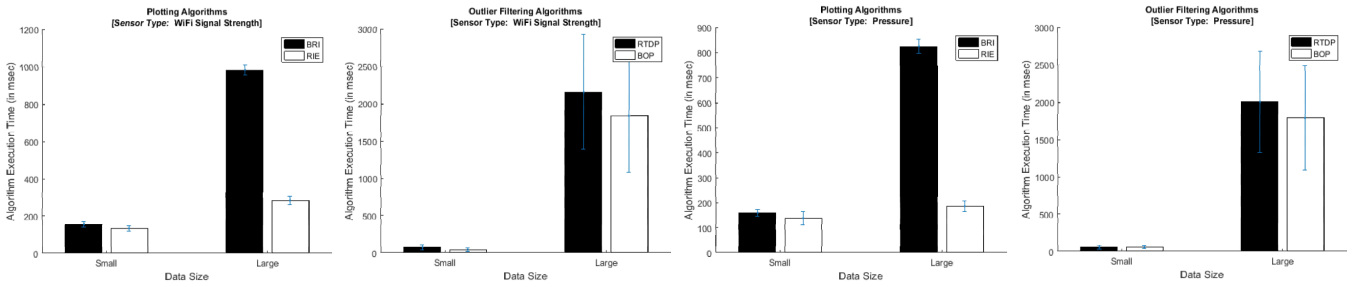
Test Inst.	Dataset Size (Data entries/bytes)	Sensor Type
T1	Small (3000/500K)	WiFi RSSi
T2	Large (50000/1.2M)	WiFi RSSi
T3	Small (3000/500K)	Pressure
T4	Large (50000/1.2M)	Pressure

B. Experimental Evaluation

This subsection compares the proposed plotting and data mapping algorithm RIE and the proposed outlier filtering algorithm RTDP with the BRI and BOP (described in Section IV-A), respectively, in all test instances of Table II in terms of their individual execution time.

The statistical results of Table III as well as their visual representation in Figure 4 show that regarding the plotting and data mapping approaches, the performance of RIE is better than BRI, in all test instances. In particular, RIE executes around 30 msec faster than BRI for the small datasets of test instances T1 and T3, and around 650 msec faster for the large datasets of test instances T2 and T4, on average. In terms of standard deviation, the results show that both algorithms are relatively independent of the data size since the standard deviation is small in all test instances. Even in this case, however, the proposed RIE provides a similar performance than BRI for small datasets and slightly better performance for large datasets. Both algorithm behave similarly for both sensor types, however, it is important to notice the scalability of the proposed RIE approach since its efficiency is not proportionally influenced by the increase of the data size. The increase from the small (3000 entries) to the large (50000) data set was around 16.7 times, but the increase of the RIE's execution time is about only 1.6 times for both sensor types.

Regarding the outlier filtering approaches, the results show a similar performance by RTDP and BOP in small datasets, but a better performance of around 300 msec for BOP with respect to the proposed RTDP in large datasets, as expected. This is due to the fact that BOP compares every new entry with just the pre-specified min-max values of each sensor type. RTDP, on the other hand, makes a more detailed check in order to identify the outliers, since it compares each new entry with the pres-specified min-max values as well as with the average of all values falling within a fixed distance from the contributor's current location. Even though this can be considered a costly process, the difference between the two approaches is still small. Moreover, both algorithm behave similarly in terms of



(a) Test Instances: T1 & T2 (b) Test Instances: T1 & T2 (c) Test Instances: T3 & T4 (d) Test Instances: T3 & T4

Fig. 4. Comparison of all algorithms in all test instances in terms of execution time (in msec)

TABLE III
COMPARISON OF ALL ALGORITHMS IN ALL TEST INSTANCES IN TERMS OF EXECUTION TIME (IN MSEC)

		Plotting/Data Mapping		Outlier Filtering	
Test Inst.	Statistics	BRI	RIE	RTDP	BOP
T1	min	140	117	50	19
	max	208	219	311	175
	average	155.909	134.131	76.909	43.898
	std	13.480	14.926	36.471	22.178
T2	min	956	255	531	465
	max	1060	381	5491	5127
	average	981.808	284.565	2156.252	1835.626
	std	28.168	20.545	768.549	752.548
T3	min	145	117	32	29
	max	215	222	193	173
	average	159.878	138.161	57.898	55.262
	std	13.888	26.707	23.644	18.9
T4	min	801	166	870	335
	max	947	280	4646	5159
	average	824.888	186.464	2004.01	1786.838
	std	28.394	21.359	676.162	696.634

standard deviation for both sensor types and they both scale well and efficiently with respect to the dataset size increase. In particular, BOP performs slightly more steadily for test instances T1 to T3, but offers around 20 msec less std for test instance T4. The 16.7 times increase of the dataset size, however, looks to have influenced more the BOP approach than the proposed RTDP, since the execution time of BOP increased by 38 times and that of RTDP increased by 31 times.

V. CONCLUSIONS

This paper presents an innovative Crowdsourcing Platform as a Service, coined CrODA-gator, which follows a scalable and extensible architecture, offering data access to various devices, supporting both real-time and offline data aggregation and enabling open access to the collected data. CrODA-gator also incorporates a set of mechanisms for outlier filtering and a web viewer with an optimal GIS mapping of the spatial data for real-time visualization as well as a dedicated smartphone application and well-documented APIs. The most important modules of the proposed platform were evaluated over datasets of various sizes, for different sensor types in terms of their execution time. The experimental study, using realistic sensor data (offline), reveals that *CrODA-gator* provides an efficient and scalable PaaS that is open to the public for contributing and using crowdsourced data. Future work aims at an experimental study with crowdsensed data in real-time.

ACKNOWLEDGMENT

The authors would like to thank Ms. Jovanna Zonia for implementing and providing the Android application code.

REFERENCES

[1] J. Schwartz, J. M. Samet and J. A. Patz, "Hospital admissions for heart disease: the effects of temperature and humidity," *Epidemiology*, vol. 15, no. 6, pp. 755-761, 2004.
[2] B. Guo, Z. Yu, X. Zhou and D. Zhang, "From participatory sensing to Mobile Crowd Sensing," *IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, Budapest, 2014, pp. 593-598.
[3] T. Li et al., "Scalable privacy-preserving participant selection in mobile crowd sensing," *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Kona, HI, 2017, pp. 59-68.
[4] D. Guedj and A. Weinberger, "An exploratory look at supermarket shopping paths," *International Journal of research in Marketing*, vol. 22, no. 4, pp. 395-414, 2005.
[5] C. Kronis, A. Konstantinidis and H. Papadopoulos, "Human-Like Agents for a Smartphone First Person Shooter Game using Crowdsourced Data," *IFIP Artificial Intelligence Applications & Innovations (AIAI'13)*, Springer, Paphos, Cyprus 2013.
[6] G. Chatzimiloudis, A. Konstantinidis, C. Laoudias, D. Zeinalipour-Yazti, "Crowdsourcing with Smartphones," Special Issue: Crowdsourcing, *IEEE Internet Computing*, vol. 16, no. 5, pp. 36–44, 2012.
[7] T. Imielinski and B. R. Badrinath,"Querying in highly mobile distributed environments," *VLDB*, vol. 92, pp. 41-52, 1992.
[8] D. T. Wagner, A. Rice and A. R. Beresford, "Device analyzer: Understanding smartphone usage," *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services,*, 2013.
[9] K. Georgiou, T. Constambeys, C. Laoudias, L. Petrou, G. Chatzimilioudis and D. Zeinalipour-Yazti, "Anyplace: A crowdsourced indoor information service," *16th IEEE International Conference on Mobile Data Management*, 2015.
[10] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12-18, 2008.
[11] M.-C. Yuen, I. King and K.-S. Leung, "A survey of crowdsourcing systems," *3rd IEEE Conference on Social Computing*, 2011.
[12] D. Govindaraj, K. Naidu, A. Nandi, G. Narlikar and V. Poosala, "MoneyBee: Towards enabling a ubiquitous, efficient, and easy-to-use mobile crowdsourcing service in the emerging market," *Bell Labs Technical Journal*, vol. 15, no. 4, pp. 79-92, 2011.
[13] G. Merlino, S. Arkoulis, S. Distefano, C. Papagianni, A. Puliafito, S. Papavassiliou, "Mobile crowdsensing as a service: A platform for applications on top of sensing Clouds," *In Future Generation Computer Systems*, Volume 56, 2016, pp. 623-639.
[14] S. Distefano, A. Puliafito, G. Merlino, F. Longo and D. Bruneo, "A Stack4Things-based platform for mobile crowdsensing services," *ITU Kaleidoscope: ICTs for a Sustainable World (ITU WT)*, 2016, pp. 1-8.