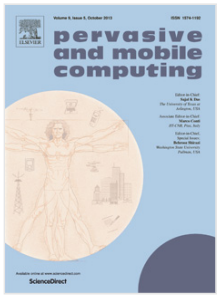
Accepted Manuscript



A novel localization and coverage framework for real-time participatory urban monitoring

Adnan Khan, Sk Kajal Arefin Imon, Sajal K. Das

PII: S1574-1192(15)00139-X
DOI: http://dx.doi.org/10.1016/j.pmcj.2015.07.001
Reference: PMCJ 617

To appear in: *Pervasive and Mobile Computing*

Please cite this article as: A. Khan, S.K.A. Imon, S.K. Das, A novel localization and coverage framework for real-time participatory urban monitoring, *Pervasive and Mobile Computing* (2015), http://dx.doi.org/10.1016/j.pmcj.2015.07.001

A Novel Localization and Coverage Framework for Real-time Participatory Urban Monitoring

Adnan Khan

Department of Computer Science and Engineering
The University of Texas at Arlington

Sk Kajal Arefin Imon

Department of Computer Science and Engineering
The University of Texas at Arlington

Sajal K. Das

Department of Computer Science
Missouri University of Science and Technology

Abstract

Participatory sensing is a powerful paradigm in which users participate in the sensing campaign by collecting and crowdsourcing fine-grained information and opinions about events of interest (such as weather or environment monitoring, traffic conditions or accidents, crime scenes, emergency response, healthcare and wellness management), thus leading to actionable inferences and decisions. Based on the nature of user involvement, participatory sensing applications can be of two types—automated and user manipulated. The first type of applications automatically collects data samples from smartphone sensors and sends them to the server. On the other hand, the second type of applications depends on the users to manually collect data samples and upload them at their convenience. Because of the high density of smartphone users in urban population and ease of participation, the automated participatory sensing paradigm can be effectively applied to continuous monitoring of various phenomena in urban scenarios (e.g., fine-grained temperature monitoring, noise or air pollution), leading to what is called participatory urban sensing. However, for creating a fine-grained and real-time map of the monitored area, the data samples need to be collected continuously (at a high frequency) which poses several research challenges. First, how to ensure coverage of the collected data that reflects how well the targeted area is monitored? Second, how to localize the smartphones since continuous usage of the location sensor (e.g., GPS) can drain the battery

Email addresses: adnan.khan@mavs.uta.edu (Adnan Khan), skkajal.imon@mavs.uta.edu (Sk Kajal Arefin Imon), sdas@mst.edu (Sajal K. Das)

in few hours? Third, how to provide energy efficiency in the data collection process by collecting necessary data samples in each data collection round? In this article, we propose a novel framework called PLUS to address three major issues in real-time participatory urban monitoring applications, namely, ensuring coverage of the collected data, localization of the participating smartphones, and overall energy efficiency of the data collection process. Specifically the PLUS framework can guarantee a specified requirement of partial data coverage of the monitored area in an energy efficient manner. Additionally we devised a Markov-Predictor based energy efficient outdoor localization scheme for the mobile devices to participate in the data collection process. Simulation studies and real life experiments exhibit that PLUS can significantly reduce energy consumption of the mobile devices for urban monitoring applications as compared to traditional approaches.

Key words: Participatory sensing, real-time urban monitoring, localization, energy efficiency.

1. Introduction

Over the last decade, we have witnessed tremendous advancement in the smartphone technology. Mobile phone has evolved from basic communication device to a powerful sensing platform with a rich set of built in sensors. Examples include microphone for capturing audio, camera for capturing video, RGB light sensor for measuring intensity of light, gesture sensor for detecting hand movement, Global Positioning System (GPS) for retrieving location, barometer for measuring atmospheric pressure, accelerometer for measuring acceleration, gyro sensor for determining rotation state of the device, fingerprint sensor for identifying user fingerprint, pulse sensor for monitoring heart rate, and so on. Moreover, modern smartphones provide convenient interfaces (e.g., Bluetooth, WiFi, NFC) to connect with external sensors and devices, hence giving birth to a host of wearable gears such as Apple Watch [1], Samsung Gear [2], iHealth Edge [3], etc., which stay connected with the smartphones while monitoring various phenomena surrounding the user. As a result, a new paradigm of applications exploiting the sensing platform of the mobile devices emerged in the last decade and gained significant attention both in the industry and research community.

Due to the ubiquitous nature of smartphone and its extremely large user base, it has been leveraged to design applications in different domains of our life. For example, applications were designed where the smartphone sensors are used for assisting and monitoring individuals, such as activity, sleep, and overall health and well-being monitoring [4, 5, 6]. By taking it a step further, data collected from individuals can be compiled by healthcare providers or government agencies and associated with environmental factors to analyze not only individual but also community-wide activity, habit and exposure [7].

Another important class of applications aims to monitor different environmental and urban scenarios by collecting data from the smartphone sensors. Similar to monitoring environmental or urban scenarios with Wireless Sensor Networks (WSNs), a network of smartphones can be conceived for such tasks where a smartphone is treated as a location-aware multi-modal sensor node, and participates in the data collection process by sampling the required sensor. The notion of sensory data collection through the participation of a group of smartphone users to create knowledge is known as *participatory sensing* [8, 9, 10, 11, 12] in general.

Depending on the use case, there are several design choices for a participatory sensing application. For example, based on the feature to monitor, it can be designed for monitoring urban/environmental feature or personal (smartphone user's) scenario. Based on user involvement for data collection, an application may notify the user each time to collect data sample, whereas, an application can be designed to automatically collect data samples as required, once authorized by the user. Some application can be designed for monitoring only indoor events, while some other may target outdoor scenarios. Based on the activity or motion state of the smartphone user, there can be variations too—some application can be designed to collect data samples from pedestrians, while some other may collect data from smartphone users moving in vehicle. Finally, based on the update frequency requirement of the underlying monitored phenomenon, an application could be designed to collect data continuously or at a much lower frequency. Adopting each of these design choices poses its own research challenges that must be solved for successful deployment of a participatory sensing application. In our work, we primarily address the challenges of participatory sensing applications designed for monitoring urban scenarios automatically and continuously, also referred as real-time participatory urban monitoring application.

One important challenge in the data collection process is how to ensure the coverage of the collected data? For traditional Wireless Sensor Networks (WSNs), this problem has been extensively investigated where sensors are either static and deployed in a known area, or moving with predefined trajectories [13, 14, 15, 16, 17]. Hence, sensors in WSNs can be scheduled to collect data samples as required to achieve the coverage. In case of participatory sensing applications, data collection is performed by mobile devices (smartphones) carried by human users with uncontrolled mobility. Another important challenge is the localization of the smartphones especially when the location information is required continuously. And lastly an important challenge is to ensure energy efficiency in the data collection process. For WSNs, energy efficiency in the data collection process has been widely investigated from different aspects [18, 19, 20, 21]. Similar to wireless sensors, the mobile devices are battery powered and hence energy efficiency in the data collection process is also very important [22, 23, 24]. The existing approaches that address these challenges broadly fall into two categories. In the first category, the server collects data samples from all participating mobile devices at a predefined frequency (as low as 1 sec) [25], resulting in redundant data sample collection from densely popu-

lated urban area. In the second category, each participant informs its location to the server. In turn, the server selects the mobile devices as required to ensure different criteria (e.g., coverage, battery life, etc.) [26, 23]. However, for a real-time continuous monitoring applications by pedestrians, location needs to be frequently updated and informed to the server, which can take significant amount of battery life.

In this paper, we propose a novel framework called *Energy Efficient Framework for Localization and Coverage in Participatory Urban Sensing* (PLUS) [27], where a mobile device is not required to send its location information to the server. PLUS divides the monitored area into smaller blocks, and uses a metric called *desired sensing coverage (DSC)* [28], to ensure partial coverage of the collected data associated with each block. A participating mobile device executes a novel energy efficient localization scheme to determine the block level location (i.e., the block where a user currently is) of the user. The sLoc scheme maintains individual mobility history and uses a Markov Predictor Model to predict his future block transitions (i.e., the block where the user is likely to move from the current block). Moreover, to activate the GPS effectively to determine a block transition, sLoc learns from previous behavior of the user in each block. After determining the block level location, a mobile device probabilistically performs the sensing task and sends the collected data to the server to ensure the DSC requirement. The major contributions of this paper can be summarized as follows.

- We propose PLUS, a framework for continuous monitoring applications using participatory sensing. PLUS is able to guarantee partial coverage of the collected data in an energy efficient way without knowing the location of the participating mobile devices.
- We formally define the *block transition detection problem* to continuously determine the block level location of a user with smartphone while minimizing the usage of the location sensor (GPS). Then we propose our localization scheme *sLoc* to solve it.
- We implemented sLoc in Android devices. Through real world experiments, we show that sLoc can reduce GPS usage as much as 85% when a mobile user follows his regular routes, as compared to a method that continuously uses GPS.
- We emulated a participatory sensing application to continuously monitor WiFi signal strength in our university campus using the PLUS framework for different DSC (a partial coverage metric) requirements. Experimental results demonstrate that our framework can significantly reduce energy consumption as compared to the traditional approaches.
- We discuss the impact of data collection frequency on the sLoc scheme and its applicability on participatory monitoring applications. We further show the effect of data collection frequency on the energy savings of the data collection process.

- We derive conservative condition on the applicability of our framework to perform data collection in an energy efficient manner.

The rest of the paper is organized as follows. Section 2 gives an overview of the related work dividing them into different category. Section 3 proposes the PLUS framework and describes the coverage metric and the localization scheme used for the framework. Results from simulation and real world experiments are presented in Section 4. Section 5 discusses several observations and challenges for successful deployment of the proposed framework. Finally, conclusions are drawn in Section 6 with direction for future research.

2. Related Work

In the recent years, a plethora of applications in the field of mobile sensing and participatory sensing have been proposed. Detail surveys on these types of applications can be found in [29, 30]. Both mobile sensing applications and participatory sensing applications undergo several common research challenges, e.g., energy efficient data collection, duty cycle GPS usage. Additionally, participatory sensing applications experience some extra challenges because of the collaborative nature of monitoring task. In the following, we will categorize and summarize the work that are relevant to the research challenges we intend to solve (coverage, localization and energy efficiency in data collection).

2.1. Continuous Mobile Sensing

In a continuous mobile sensing application, battery can be depleted fast because of the frequent usage of a sensor. To address this, many work have been proposed in the literature. For example, the *Jigsaw continuous sensing engine* [31] proposed a platform for continuous monitoring of user context and activities in an energy efficient way by balancing the performance needs of the underlying application and the resource demands for sensing. In [32], Yan et.al proposed a method for continuous locomotive activity recognition based on accelerometer. To reduce energy consumption of continuous sensor sampling, they applied an activity-sensitive strategy to control the sensor sampling in real time, based on user activities. In [33], Nath presented Acquisitional Context Engine (ACE) that uses the inherent structure of the context inference problem for continuous context sensing. The main idea behind ACE is to use associative rules among contexts to infer one context from another, sometimes without using any sensor. In our case, we also control the collection of data samples based on application requirement. Moreover, we apply participant's velocity in a block as context to schedule GPS usage appropriately.

2.2. Delay Tolerant Mobile Crowdsourcing

There are some scenarios where the collected data is not time-sensitive, and can be delivered at a later period than the time of collection. In this case, applications choose to upload the collected data only when the smartphone is

connected to an access point/Wi-Fi/PC following a *delay tolerant crowdsourcing* approach, making the data upload process energy efficient. Some recent articles applied this approach [34, 35, 36, 37] for data collection in an opportunistic manner. Some work also proposed to enhance the opportunistic data transfer experience by using prediction models [38, 39]. In our scenario, pedestrians need to send the collected data right-away to contribute in creating a real-time map of the monitored area. Therefore, this method of energy saving is not directly applicable in our case.

2.3. Piggyback supported Mobile Crowdsourcing

A variant of the delay tolerant data collection application intends to collect data not only when a user comes in contact to the Wi-Fi/PC, but also when he makes/receives a call, uses an app, etc., following a *piggyback supported mobile crowdsourcing* (PCS) approach. In [40], the authors showed that the PCS approach can collect data samples using significantly less energy (up to 90%) by exploiting prediction models on the app usage pattern of the user. However, they did not provide any delay constraint within which data can be collected, making it unusable in our context. CrowdRecruiter [41] proposes an energy efficient participant recruitment framework that achieves probabilistic coverage constraint while minimizing the number of required participants. However, the authors used cell tower ID as the coverage metric which is not granular enough for many scenarios where creating a fine-grained map is required.

2.4. Real-time Mobile Crowdsourcing

Applications supporting real time data offloading usually follow two approaches. In the first approach, the participating mobile devices collect data samples at a predefined frequency and upload to the server to monitor different urban and/or environmental features, such as Carbon Monoxide [25] level, event map of a city [42], personal exposure to pollution [43], and so on. The major challenge of this approach is that, a significant amount of energy can be consumed when data collection frequency is high as each participant performs the sensing task in every data collection round. As a consequence, redundant data samples can also be collected. In the second approach, the server tracks the participating devices and selects only the required ones for data collection. Some applications following this approach also guarantee coverage ensured data collection to make sure that the collected data samples represent the monitored area and avoid unnecessary sample collection [26, 44, 23, 45]. However, the major limitation of these work is that the position fix by location sensor and sending location update to the server can introduce significant energy overhead on the mobile devices, especially when the data collection frequency is high. PSense [24] was proposed to reduce unnecessary position fixes by GPS to save energy but still it spends a lot of energy by sending location updates to the server. Besides these two major approaches, some work also suggested to apply an adhoc networking assisted crowdsourcing [26, 44, 46]. However, we argue that, for real-time monitoring application, an adhoc networking based method

may not be suitable since the server requires continuous data samples from the participants for creating fine-grained maps.

Our framework is based on the concept of minimizing server influence in the data collection process [27, 47]. It is somewhat close to the first category of applications, as the server does not receive any location update from the participating mobile devices. But unlike the first approach, it does not collect data samples from all participating mobile devices. It collects necessary data samples to probabilistically ensure partial data coverage requirement. To this end, the participants use a localization scheme that provides continuous block level location—a requirement for the decision making of the participation in each data collection round. Unlike the above two approaches, our method does not overuse the location sensor or data collection sensor for high data collection frequency, thus making it an excellent choice for real-time participatory monitoring application. In the next section, we will describe our proposed framework in detail.

3. PLUS Framework

In the proposed PLUS framework, there is a server (or cluster of servers), offering web services to receive data samples from participating mobile devices. A participant is required to install a software package developed for PLUS on his mobile device that includes a sensory data sampling component shown in Figure 1a. This component takes help from *sample requirement calculator* and *localization scheme* that are described later in this section. Each participating mobile device has one or more built-in sensors (e.g., microphone, barometer, etc.) to perform the sensing task, as required by the application. Note that, PLUS is used for continuous monitoring scenarios at outdoor locations in densely populated urban areas (university campus, downtown areas, etc.). Therefore, sensory data samples are collected by a mobile device only when the user walks around the monitored area.

A mobile device may perform the sensing task only once in a predefined time interval, known as the *data reporting round*. We assume that each mobile device has uniform sensing range of r and can sense a circular area with radius r. The *monitored area* Q is divided into multiple equal sized *blocks* denoted by a unique ID B_i, where $1 \leq i \leq N$, and N is the total number of blocks (e.g., $100 \times 100m^2$ or $150 \times 150m^2$). In each data reporting round, a mobile device detects its block level location, i.e., the block number where the user currently is, and with a certain probability, performs the sensing task and uploads the data to the server.

As battery life is a critical issue for mobile devices, PLUS has several principles to reduce energy consumption. First, a mobile device does not need to know its exact location, as continuously using location sensor can take significant amount of battery life. Second, the server is not required to know the location of the mobile devices, as sending the location updates to the server also consumes a lot of energy. Finally, PLUS does not intend to collect data samples from all participants invariably. Rather it collects data samples from

necessary mobile devices to meet the given value of DSC which is a metric for partial sensing coverage described in the next subsection. The sample requirement calculator and the localization scheme ensure that, all these principles are followed.

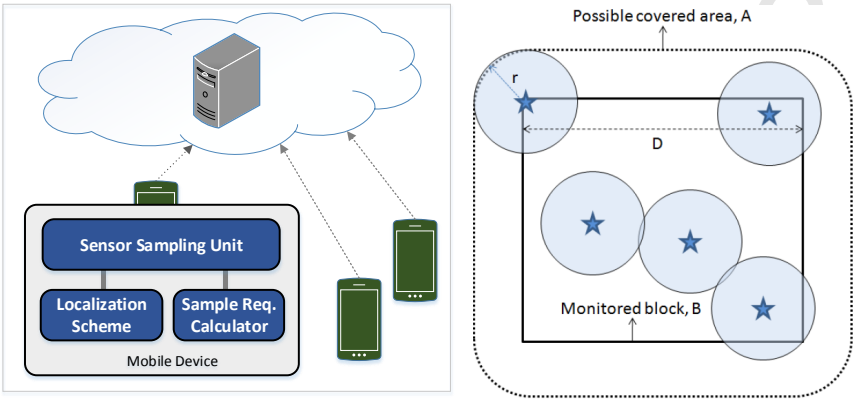
To give an insight how PLUS ensures energy efficiency, let us consider a simple example with n participants in a block where a partial coverage of c is required. In traditional approaches, all n participants send their locations to the server that selects the necessary number of mobile devices (say m) to ensure the given coverage requirement implying a total of $n + m$ server communications. On the other hand, in our approach a mobile device does not send its location to the server. Rather it determines which block the participant is at any instant (with the help of our localization scheme) and performs the sensing task with a certain probability. Suppose, to ensure the partial coverage c, a total of m' data samples are required from the block in our approach. Hence a participant collects data sample with m'/n probability and sends the data to the server implying only m' server communications. For decent participant density, m' is much smaller than $n+m$, thus saving significant amount of energy. In the next two subsections, we will describe how the required data samples are collected, and the block level location is determined, respectively.

3.1. Sample Requirement Calculator

The primary job of the *sample Requirement Calculator* is to ensure that the participating mobile devices upload necessary data samples to satisfy the coverage requirement. PLUS uses *desired sensing coverage* (DSC), which is a partial sensing coverage metric of the collected data for a block. The application is allowed to specify its required DSC, and PLUS decides the number of data samples (k) required for each block accordingly.

Definition 1 (Desired Sensing Coverage) [28] Given the block partition of an area, *Desired Sensing Coverage* (DSC) is the probability of any point in a block B_i to be covered by the circular sensing area of at least one of the k mobile devices residing in B_i that are selected during a data reporting round. We denote DSC with λ. □

Now we briefly show the relationship between the λ and k as presented in [28]. Let us consider a square shaped block B_i with sides of length D. We assume that, at a given time instant a mobile device can be located at any point in the block with uniform probability. Based on its position in the block, a mobile device can sense any point from an extended area $A = D^2 + 4Dr + \pi r^2$ that we refer to as the *possible covered area* of the block as depicted in Figure 1b. Let α represent the area corresponding to the block B_k covered by a randomly selected mobile device. Therefore, the expected value of α can be written as $E[\alpha] = \pi r^2 \frac{D^2}{A}$. The probability that a point (x, y) in the block B_i is not covered by k mobile devices selected independently and at random can be derived as $P_k(x, y) = \left(\frac{D^2 - E[\alpha]}{D^2}\right)^k$. Therefore, the desired sensing coverage



(a) (b)

Figure 1: (a) Block diagram of PLUS, (b) Area covered by mobile users in a block.

λ can be written as $\lambda = 1 - \left(\frac{D^2 - E[\alpha]}{D^2}\right)^k$. From this relationship, the smallest value of k that satisfies the DSC is obtained as [28],

$$k = \frac{\log(1-\lambda)}{\log\left(\frac{D^2+4Dr}{D^2+4Dr+\pi r^2}\right)} \tag{1}$$

Next, we apply a simple observation, that is, some parts of the monitored area may have more participants than other parts. Moreover, the number of the participants can vary with time. For example, in a university campus, most of the areas (especially where the academic activities are performed) may have a significantly higher mobile user density in the daytime as compared to the evening. On the other hand, in the evening the user density might be quite high in selective places (e.g., students activity center or housings). Such mobile user density information is often readily available from commercial vendors [48] or can be derived from cell tower data [49] and is out of scope of our work. Based on the participant density and the value of k (from Equation 1), each block is associated with different probabilities for different time periods. The server delivers this probability values to the mobile devices only once during the program setup. In each data collection round, a participating mobile device detects the current block using the localization scheme (described in the next subsection), performs sensing task according to the probability associated to this block, and uploads the sensed data to the server. As a result, the server is expected to receive the minimum number of data samples for each block.

3.2. Localization Scheme

In this subsection, we present *sLoc*, a localization scheme used on the mobile device of each participant. The sLoc scheme is designed to provide block

level location continuously while the user walks around the monitored area. It uses the GPS as location sensor (since PLUS is designed to monitor outdoor scenarios). A naive approach would be continuously using the GPS to determine location. But that would drain the battery very quickly. One possible way to save energy is to turn off the GPS once the current block is detected, and after a suitable time interval, activate it so that the entrance to the next block is also detected. Our scheme sLoc uses this technique intelligently to reduce energy consumption. Specifically, it derives an efficient schedule to activate and deactivate the GPS to detect the block transitions of the user.

Now we discuss technical components of sLoc. At its core, sLoc is based on a simple observation that, a user often follows similar routes. Moreover, a user is likely to spend similar amount of time in a block while following the same route. This spatial consistency is exploited by the sLoc scheme to predict not only the next block in which the user is likely to move to, but also the amount of time the user is expected to spend in the current block.

Figure 2a shows the block diagram of the sLoc scheme. As the user moves around the monitored area, block transition data are produced. The *Block Predictor* unit uses the historical block transition data to predict the next block the user is likely to move from the current block. Considering the possible block transitions, the *Duration Estimator* unit determines the amount of time the user is likely to stay in the current block. Finally, once the user is detected to be in the current block, the *GPS Scheduler* keeps the GPS in sleep mode as long as the user is likely to stay in this block, and activates it as necessary to capture new block transition data. As a result, sLoc simultaneously perform learning and prediction to avoid the usage of GPS as much as possible with robust block level location detection. The key problem in block transition detection is choosing optimal interval to activate the GPS defined as follows.

Definition 2 (Block Transition Detection Problem) Given a mobile user is moving through the block sequence $B_1, B_2, B_3, \ldots$, with a block transition probability function $\mathcal{F} : i \times j \rightarrow \mathbb{R}$ where i and j are the current and next block index, the Block Transition Detection Problem is to determine time intervals $\Delta t_1, \Delta t_2, \Delta t_3, \ldots$ to activate the GPS that maximizes $\sum_{k=1}^{\infty} \mathcal{R}(\Delta t_k)$ while detecting the block transitions. Here $\mathcal{R}$ is a function of the block transition detection accuracy and the energy required to activate the GPS at a given schedule. □

Now we will introduce two metrics, namely *accuracy payoff* and *energy payoff*, that would be used to define the function $\mathcal{R}$. To determine a block transition, if sLoc receives the location at time t_{sense} after scheduling the GPS according to an interval Δt_k, the *accuracy payoff* is defined as

$$\mathcal{A}(\Delta t_k) = \begin{cases} \frac{t_{actual}}{t_{sense}}, & \text{if } t_{sense} > t_{actual}. \\ 1, & \text{otherwise.} \end{cases} \quad (2)$$

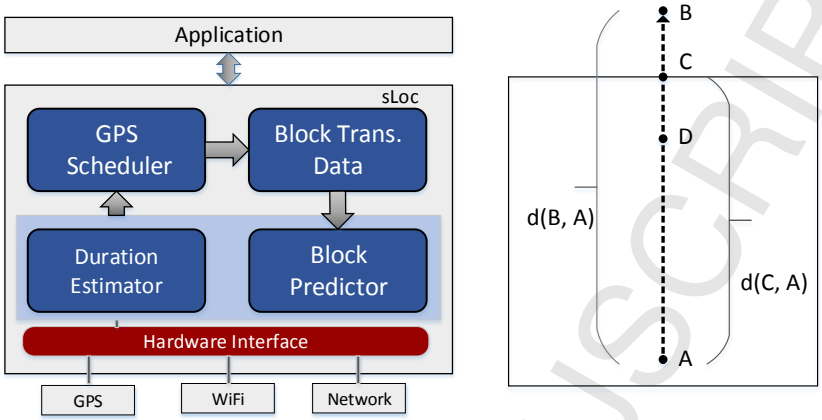


Figure 2: (a) block diagram of sLoc, (b) Location retrieval at different point in a block.

And the *energy payoff* is defined as

$$\mathcal{E}(\Delta t_k) = \begin{cases} \frac{t_{sense}}{t_{actual}}, & \text{if } t_{sense} < t_{actual}. \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

Here, t_{actual} is the time instant when the user leaves the current block. However, measuring t_{actual} is not possible when $t_{sense} > t_{actual}$. In that case, we estimate the accuracy payoff as follows. Suppose, the user location is detected at point A in the current block (as shown in Figure 2b) and after that sLoc puts the GPS in sleep mode for time interval Δt_k. Upon next activation, the GPS detects the user location at point B. Then, $\mathcal{A}(\Delta t_k)$ is measured as $\frac{d(C,A)}{d(B,A)}$, where $d(X,Y)$ is the Euclidean distance between the points X and Y. Finally, $\mathcal{R}$ is defined as follows:

$$\mathcal{R}(\Delta t_k) = \mathcal{A}(\Delta t_k)\gamma + \mathcal{E}(\Delta t_k)(1-\gamma) \tag{4}$$

where γ is a constant. Now we will describe the components of the sLoc scheme and how it solves the block transition detection problem.

3.2.1. Block Prediction Unit

The goal of the sLoc scheme is to determine an effective schedule to activate and deactivate GPS periodically to solve the Block Transition Detection Problem. And for that, it is important to determine the next block a user is likely to move from the current block. However, a user may move to different blocks from the current block with different probabilities. The role of the *block prediction unit* is to determine the transition probabilities of the user moving into next blocks. That means, given the current block, the block predictor outputs a list of block ID and probability pairs.

Mobility prediction for the mobile phone users is a fairly well studied area. While there are some methods that track the mobile user from the server [50], some other methods run locally on the mobile devices [51]. As we desire to keep the localization task at the user end, we are interested in using local predictors. The performance of several such predictors were presented in [51] to predict locations using the Wi-Fi access point observations by human subjects. They concluded that the low-order Markov predictor outperforms more complex and space-consuming methods, such as compression-based predictors. Therefore, we choose Markov predictor model and applied that in our block transition detection problem to predict the future block(s) where the user is likely to move from the current block. Now we will briefly describe how the Markov predictor is used for future block prediction.

Let us assume, X is a random variable that represents the block ID of the user. After making $(n-1)$ transitions, the current block of the user is represented by X_n. For example, if the user is currently in block B_k after making $(n-1)$ block transitions, then $X_n = B_k$. Let, $X(i,j)$ represents a sequence of random variates $X_i, X_{i+1}, \ldots, X_j$. The order-k Markov predictor assumes that the future block of the user can be predicted from the user's current *context* that is the sequence of k most recent blocks (i.e., $X(n-k+1,n) = X_{n-k+1}, X_{n-k+2}, \ldots, X_n$). Then the block transition probability can be expressed as

$$\begin{aligned} Pr(X_{n+1} = B_{k'}|X(1,n)) &= Pr(X_{n+1} = B_{k'}|X(n-k+1,n)) \\ &= \frac{N(X(n-k+1,n+1))}{N(X(n-k+1,n))} \end{aligned}$$

where $N(a)$ denotes the number of times the block sequence a occurs in the block transition history. With this estimation of transition probability, the $O(k)$ Markov predictor predicts the future block as $X_{n+1} = \underset{B_i}{\operatorname{argmax}}\, \widehat{P}(X_{n+1} = B_i|H)$.

Note here that the Markov predictor chooses the block with highest transition probability from the current state as the future block. However, we consider more than one blocks as tentative future block. To this end, we extract all the blocks with non-zero probability from the current block and select the block transitions with top-κ probabilities, and deliver those block IDs to the *sensor scheduler unit* to determine an effective GPS activation schedule. As the user moves with time, more block transitions are detected and the block transition probabilities are updated accordingly. In Section 4 we evaluate the performance of three variants of Markov predictors (namely, order-1, order-2 and order-3) with varying values of κ to determine effective values of κ and the order of the Markov Predictor.

3.2.2. Block Duration Estimator

The Block Prediction unit discussed above provides a list of future <block, probability> pairs. However, it does not provide information on how long the

user is likely to stay in the current block. Consequently, it does not answer the question what would be the most suitable time to activate the GPS again to detect the transition from current block. A simple approach is to use a predefined velocity (say 1.5m/s or 2m/s) for the user and associate that with the block dimensions to predict the duration. But such an approach is not dynamic enough to incorporate varying human mobility behavior. Different users may have different velocities. Another approach could be using the accelerometer and/or gyroscope to estimate the velocity and direction to predict the amount of time the user will spend in the current block. Such an approach may experience some challenges. First, accelerometer consumes significant amount of energy and needs to be duty cycled carefully [52]. Second, GPS takes different amount of time in different places. Figure 3a shows the amount of time GPS takes to retrieve location in different parts on our campus by different mobile phones. Therefore, this factor also needs to be considered while scheduling the GPS to keep the activation period as short as possible.

A user may spend different amounts of time in different blocks which depend on his walking speed, remaining distance to reach to the next block, and other conditions (e.g., time to retrieve location, smartphone model). So we consider two factors in determining the remaining time in the current block, x_1 - that corresponds to the perpendicular distance to the previous block already traveled by the user in the current block, and x_2 - that corresponds to the total time required to cross the block and time to retrieve location by GPS in the current block. Thus, the remaining time in the current block can be written as, $t_r = f(x_1, x_2)$. To explain this, let us consider a simplified example where a user has entered block B_1 from B_0 and moving towards B_2. He takes a total time T_1 to cross B_1 and reach to B_2. Suppose his average linear walking speed in B_1 is v, and he has already traveled a linear distance of D in B_1. Therefore, the time after which GPS needs to be activated is $t_r = T_1 -$(delay for location retrieval) - D/v implying $t_r = w1.x_1 + w2.x_2$.

The values of w_1 and w_2 can vary in different blocks for different users. Moreover, it can change over time if the walking pattern of the user changes in the same block. Therefore, we take help of *associative reinforcement learning* (ARL) system [53] to adjust the values of w_1 and w_2 and estimate the time duration after which the GPS needs to be activated to detect the transition from the current block. In an associative reinforcement learning task, the learning system and its environment interact in a closed loop. The learning system takes a context vector as input and produces output for that. The environment provides a reward to the learning system by evaluating the output. As a consequence, the learning system is updated. In particular, we exploit an ARL system where the output can be real numbers [54]. For such cases, the interaction between the learning system and the environment can be described as follows. At iteration t, the environment provides the learning system with some pattern vector $\mathbf{x}_n(t)$ from $X = \mathbb{R}^n$ where $\mathbb{R}$ is the set of real numbers. The learning system then produces a random output $y(t)$ selected over some interval $Y \subseteq \mathbb{R}$. The environment evaluates $y(t)$ and sends the learning system a reward signal $r(t) \in R = [0, 1]$, where $r(t) = 1$ denotes the maximum reward.

In our case, the input of the ARL system as $\mathbf{x} = [x_1, x_2]$ where x_1 is the perpendicular distance from the current location to previous block, and x_2 is the total time taken to cross the block (measured initially), and the output (y) is the duration value after which the location sensor would be activated (ΔT_k). The function $\mathcal{R}$ (from equation 4) is used as the reward for the ARL system. The value of the constant γ in $\mathcal{R}$ was set to 0.5. Note, the function $\mathcal{R}$ can achieve highest value of 1, and it happens for a ΔT_k that results in the location retrieval exactly on the point of block transition. Thus, once the ARL system converges, the output values will lead to the activation of GPS close to the point of block transition. For solving the ARL task, an approach called *Stochastic Real Valued (SRV)* learning algorithm was proposed in [54, 55] that we used. For the sake of space, we will just give the basic idea of their approach. Their proposed algorithm maintains a Gaussian distribution $\mathcal{N}$ to generate the output value (y) for a given context. It computes the mean value μ of the distribution as $\boldsymbol{\theta}_n^T \mathbf{x}_n$, where $\boldsymbol{\theta}$ corresponds to $[w_1, w_2]$ in our case. The other required parameter of the distribution σ is generated as, $\sigma_n = \{1/n^{1/3}\}$. The parameter vector $\boldsymbol{\theta}$ is updated as follows:

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n + \sigma_n(r(y_n, \mathbf{x}_n) - r(\mu_n, \mathbf{x}_n))(y_n - \mu_n)\mathbf{x}_n. \tag{5}$$

Finally, the output value is generated as $y_n \sim \mathcal{N}(\mu, \sigma^2)$.

3.2.3. Sensor Scheduler

The primary responsibility of the sensor scheduler unit is to activate the GPS when required, and put it in the sleep mode, otherwise. Recall that, we use Markov predictor to predict the next block. And depending on the order of the Markov predictor, sLoc may consider the current block and the previous block(s) of the user to make a prediction on the future block transition. For the convenience of discussion, we will use the term *state* that comprises of the block ID(s) used in underlying Markov state. For example, if a user followed this block sequence B_1, B_2, B_3, we can also say the user moved from state ($< B_1, B_2 >$) to state ($< B_2, B_3 >$) when using order-2 Markov predictor. The sensor scheduler unit maintains an ARL system for each (current-state, next-state) pair that computes the duration in the current-state before moving to the next-state. As soon as the sensor scheduler senses that the user is in the current state, it looks up the possible next states and the corresponding ARL systems. Then it receives the duration values from each of the ARL systems and schedules the GPS for activation.

Algorithm 1 describes how the sensor scheduler manages the activation of the GPS. Note, the two variables s and T are maintained as global variables representing the current state of the user, and the list of activation times in increasing order for the current state, respectively. Once activated, the GPS periodically (every t_f seconds) receives the location for a predefined time period (τ) unless the user moves into a new state. If the user does not move to a new state in the meantime, the GPS is switched off and scheduled to be activated on the next activation time from the list T. If the list is exhausted, the GPS

Algorithm 1: OnActivation()

```
1  s: last known state;
2  T: list of activation schedules for s;
   Input: index
3  time_l ← Now();
4  while true do
5      if (Now() − time_l) ≥ τ  &  index < T.size then
6          break;
7      s′ ← GetCurState();
8      if s ≠ s′ then
9          break;
10     Sleep(t_f);
11 if s′ = s then
12     index ← index + 1;
13     Schedule(T, index);
14 else
15     Updatelearner(s, s′);
16     s ← s′;
17     T ← GetDurationValues(s);
18     Sort(T);
19     Schedule(T, 0);
```

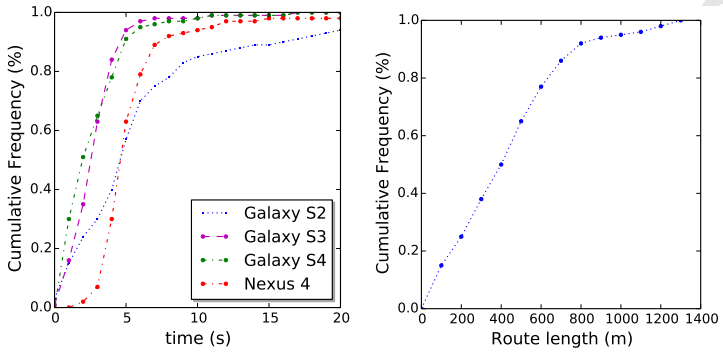
Mobile device	Number of participants
Galaxy S4	2
Galaxy S3	2
Galaxy S2	1
Google Nexus 4	2

Table 1: Number of participants with different mobile phones

keeps on receiving the location with the same frequency (once every t_f seconds) until the user moves to another state. Once the user moves to the new state s', it updates the ARL system associated with (s, s'). Then it looks up the ARL systems for the new state, retrieves the activation time from each of the ARL systems, and sorts those in increasing order. Finally, it selects the smallest activation time from the list for GPS activation.

4. Performance Evaluation

In this section, we evaluate the performance of the proposed participatory sensing framework. For that, we first evaluate the performance of sLoc in terms of GPS usage and accuracy in continuous block detection implemented on Android based smartphones with the help of real experiments. Then we discuss



(a) (b)

Figure 3: (a) Location retrieval time from GPS for different mobile phones in our campus, (b) Route length distribution of the collected trace.

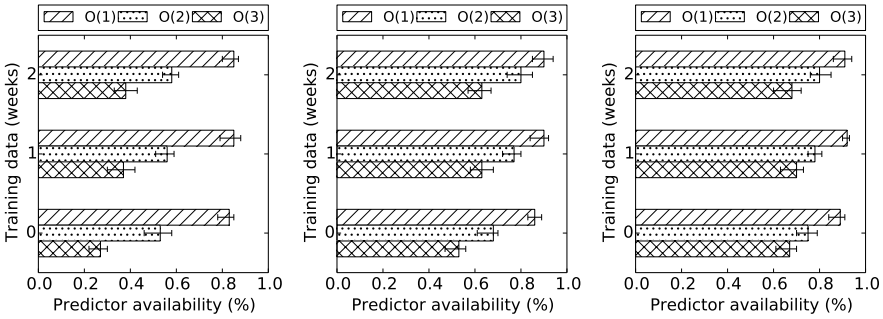
the effectiveness of the data collection process by our framework in terms of energy efficiency. However, real experiments of the data collection process would require access to many test subjects which was difficult for us. Hence, we opted for the simulation studies to show the energy efficiency of the data collection process by emulating an urban monitoring application.

4.1. Evaluation of the localization scheme

In this subsection, we will discuss the performance of our proposed localization scheme. There are several parameters used by the sLoc scheme which can affect its performance, namely, the size of the block, the order of the Markov Predictor and the number of transitions to consider from each block. Therefore, we will first experimentally determine appropriate values for these parameters and then use those to the sLoc implementation.

4.1.1. Determination of the model parameters

We collected real traces of location data with the help of GPS for 4 weeks from seven participants who live nearby the university and walk to the campus for classes and other activities. The considered area of the university campus (including student housing) is nearly 2.5 km^2 that we divide into equal sized blocks. For the trace collection, the participants used different smartphones as shown in table 1. Each user started the trace collection program to log only when they walk to the campus. From the collected data, we omitted the traces when users were at indoor locations. Thus the resulting traces represent different routes taken by the users nearby their residence and the campus (e.g., home to class, class to library, class to gym, etc.). In total, there are more than 430 routes (with repetition) in the traces. The route length distribution is shown in figure 3b. We split the collected traces into two sets, the training set and the evaluation set. The training set is used to initially compute the block transition probabilities and then the evaluation set is used to measure the performance



(a) (b) (c)

Figure 4: Predictor availability for block size (a) 50×50, (b) 100×100, (c) 150×150 meter^2.

of the block predictor unit. As the model is expected to simultaneously learn and predict, test data from the evaluation set is also added to update the block transition probability after it is used to measure performance.

The values to be considered for the parameters as follows. For block size, we considered three variations, i.e., $50 \times 50, 100 \times 100$, and 150×150 m^2. For order of the Markov Predictor, we considered order-1, order-2 and order-3. And for κ, we considered 1, 2, 3 and 4. We measure two properties, namely *predictor availability* and *future block prediction accuracy* as defined below to determine the appropriate choice of the parameters.

- **Predictor availability**: The percentage of cases when a state transition with non zero probability is found from the current state of the user.
- **Future block prediction accuracy(κ, t)**: The ratio $\frac{m(\kappa)}{n}$, based on the predictor model with training set of length t, n is the total number of block predictions made on the evaluation set, and $m(\kappa)$ is the number of cases when a user actually moved to one of the states with top-κ probabilities.

Figure 4a, 4b and 4c reports predictor availability by all three Markov predictors for three different block sizes. For each case, the training period was varied by 0, 1 and 2 weeks. There are several observations from these figures. First, for all the considered block sizes, the predictor availability increases significantly with one week of training data. However, with two weeks of training, the predictor availability does not increase noticeably. This is because of the fundamental limitation of the Markov predictor used in our model as it cannot make a prediction from a state if it was not visited before by the user. For higher order Markov predictor (e.g., order-3), the predictor availability increases to some extent using more training data. In other words, the predictor availability is lower for higher order Markov model using same amount of training data. As a state consists of multiple blocks in higher order Markov model, the state space becomes larger. As a result, it may require more time for higher

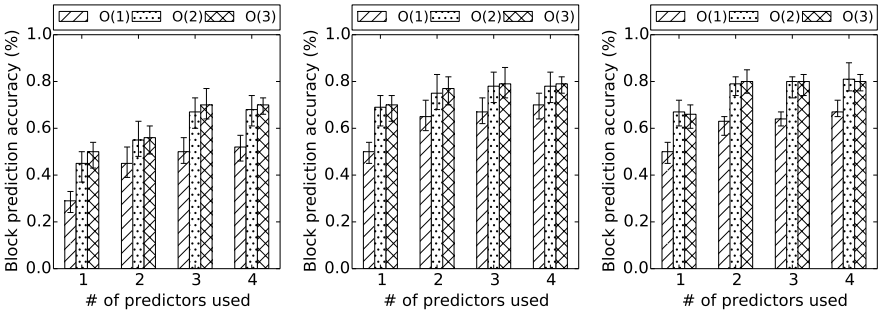


Figure 5: Future block prediction accuracy for different values of κ (a) 50×50, (b) 100×100, (c) 150×150 meter^2.

order Markov Predictors to become effective compared to lower order Markov Predictors.

Next we discuss the impact of block size on predictor availability. We observe that the predictor availability increases with increased block size for all variations of Markov Predictors in general. This observation is intuitive since a user may not always follow the exact same block sequences even when he follows the same route. For example, a user may take different sidewalks on the same road on different days. Thus deviating few meters is not uncommon when the user follows the same route in different occasions. This deviation can generate different state sequences. Choosing a smaller block increases the chance of such fluctuations, hence reducing the predictor availability. And this is true for all variants of Markov Predictors. However, arbitrarily increasing the block size also has a downside in our case. Since we are interested in predicting future blocks of pedestrians, a short route may suffer from inadequate block sequence information to build its state, especially for higher order Markov Predictors. For example, a route with length 200m may not even build any state transition information for order-3 Markov Predictor when the block size is 150m. To understand the impact of block size on accuracy, we measure future block prediction accuracy by the predictor model built with one week of training data as shown in Figure 5a, 5b, and 5c by varying the value of κ for different block sizes. We observe that the block prediction accuracy is quite low for small block ($50m \times 50m$). But it is fairly high for blocks with size $100m \times 100m$ and $150m \times 150m$. This happens because a smaller choice of block size may have fluctuations in block sequence even for the same route as mentioned before, resulting in lower accuracy. Therefore, considering both block prediction availability and prediction accuracy, we conclude that blocks with size $100m \times 100m$ would be a reasonable choice for our purpose.

Next we intend to determine a suitable choice of Markov Predictor model, both in terms of predictor availability and accuracy. We observe from Figure 4a, 4b and 4c that the predictor availability is significantly higher for order-1 Markov Predictor, as compared to order-2 and order-3. However the block prediction

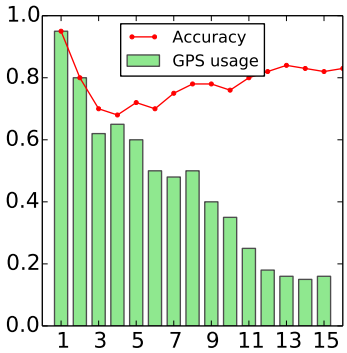


Figure 6: GPS usage and block detection accuracy together

accuracy is quite low (as reported in Figure 5a, 5b, 5c) for the order-1 model as compared to order-2 or order-3 model. For the order-3 model, the prediction accuracy can be sometimes slightly higher that order-2, whereas, the prediction availability is significantly lower than order-1 and order-2. Considering all these observations, we find that an order-1 model will not be suitable as there will be many wrong predictions. And there is no significant gain by choosing an order-3 model over order-2 in terms of the combined effect of predictor availability and accuracy, rather it has significantly increased memory overhead for its large state space. Therefore, we choose the order-2 model to serve our purpose.

Now we will determine suitable value for the number transitions to consider from each state, i.e., κ. In general, increasing the value of κ would increase the predictor accuracy. However, a greater value of κ also increases the risk of premature turning on of GPS. For example, if a user can move to block b1, b2, b3 and b4 with probability 0.1, 0.1, 0.2, and 0.6 from his current position, and after time interval 10s, 20s, 50s, and 100s respectively, a κ value of 4 will require the GPS to turn on after 10s, 20s, 50s and 100s, while the user may make the actual move to block b4 after 100s. On the other hand, choosing a value of 1 would only turn on the GPS after 100s. However, choosing very small value of κ may result in unidentified state transitions. Figure 5a, 5b, 5c demonstrate the impact of κ on prediction accuracy. From the figures we observe that the block prediction accuracy increases for increased values of κ. However, making κ larger than 3 does not bring any noticeable improvement. Therefore, we choose $\kappa = 3$ for our implementation.

4.1.2. Block detection accuracy and GPS usage by sLoc

With the choice of the parameters from previous section, we implemented the sLoc localization scheme for Android 4.0.4 to see its performance for continuous block level localization. This time, we asked 3 participants to carry two smartphones together, one running sLoc, and the other running GPS with a periodicity of two seconds. Each of them were requested to follow 2 routes from their regular movements for 15 iterations. The routes are different from each

other.

with the help of two features, namely, *block detection accuracy* and the *GPS usage* defined as follows.

- **Block detection accuracy**: The ratio $\sum_1^n T_i^s / \sum_1^n T_i^m$, where T_i^s is the amount of time spent in block B_i, as detected by the smartphone running sLoc. T_i^m is the actual amount of time spent in block B_i, as detected by the smartphone running GPS with 2s frequency, and n is the total number of blocks in the considered routes.
- **GPS usage**: The fraction of the amount of time GPS was used by the smartphone running sLoc and the total amount of time GPS was used by the second smartphone, summed over all routes.

Note, the block detection accuracy is different than the block prediction accuracy (used in the previous section). When there is no prediction for a state transition, the block detection accuracy is not necessarily low. In that case, it uses the GPS continuously to detect a new state transition and use that information to update the block transition probabilities. There are two variables used in Algorithm 1; the GPS polling interval when in use t_f and the maximum amount of time interval GPS τ that we set to 2s and 10s respectively.

Figure 6 shows the GPS usage and block detection accuracy for 15 iterations. Initially, the accuracy is very high (close to 100%) and GPS usage is also very high (almost 100%) as the block transition probabilities are not available yet. This indicates that sLoc can provide block level location with very high accuracy for an unknown route, however it cannot save any energy consumed by GPS. After that, the accuracy sharply reduces for several iterations. This happens because after the first few iterations, the block prediction unit of sLoc starts to make prediction on future block transitions. However, the block duration estimator unit needs several iterations to adjust its parameters to estimate the duration values in the current block. Once the duration estimations converge to actual duration values, and GPS usage also decreases.

4.1.3. Reducing Learning period of sLoc

From the discussion presented in Section 4.1.1, we notice that it requires some learning period (i.e., 1 week in our experiments) to build the mobility history of the Markov Predictor model to have an acceptable predictor availability. However, in a public place like university campus, there can be similarity between different people's walking pattern. Therefore, there is a possibility that previous block transition (i.e., state transition) information of other users may become helpful in the Markov Predictor model of a new user, especially when the new user has few or no state transition record. In this section, we investigate this possibility. Note, it would be really helpful if another user could be found who followed similar routes as the new user would follow. Then we could simply take that user's state transition information and apply that for the new user's predictor model. But in reality, it is not an easy job to find such a user. Therefore, we apply an alternate method. For a new user we apply only

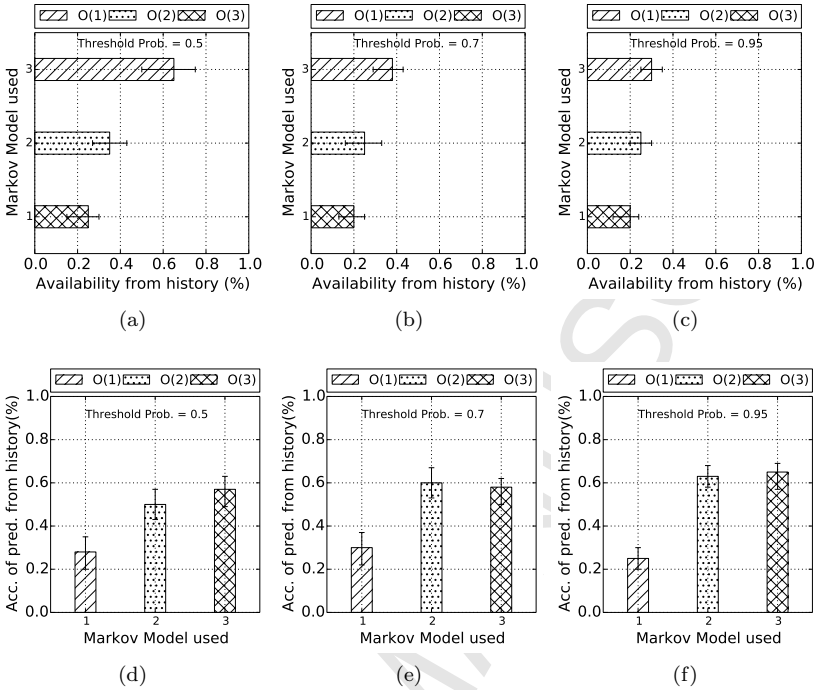
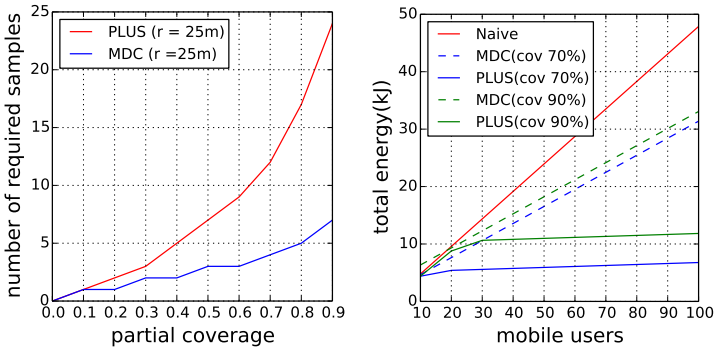


Figure 7: Predictor availability and accuracy from other users' mobility history.

the most popular state transition records from all other users, assuming that the popular state transition records would include the block sequences that the new user would also follow. So we take the transition with highest probability as well as greater than a given threshold, and consider that as the predicted state. Figure 7 shows the predictor availability and accuracy using this strategy for block size $100m \times 100m$ and varying the threshold value i.e., 0.5, 0.7, and 0.95. From the figures, we observe that smaller choice of threshold probability may result in higher prediction availabilities but quite low prediction accuracy. However, choosing very high value of threshold probability can result in high prediction accuracy, but a lower predictor availability. The reason behind low predictor availability is that the routes in the collected traces are quite different for different users. So in many cases there is no transition record to apply. Also there are places from where multiple transitions are possible, making it useless to apply. For example, if there are equal or similar transition probabilities from a block b1 to block b2, b3, b4 and b5 in other users' state transition history, applying that to a new user may not be beneficial. But if it is almost certain that b2 is always followed by b1 for most users, that transition data can be helpful in predicting a new user's future block transition. To summarize, we may attain limited benefit by applying state transitions with very high probabilities from other users to a user with no transition record. But it would be preferable to use a user's own transition records once available.



(a) (b)

Figure 8: (a) Sample requirement for varying coverage, (b) Total energy consumed by all the mobile devices

4.2. Performance of the data collection process

Now we discuss the performance of the proposed data collection framework. As stated before, our framework is targeted for dense urban area. Therefore, performing real experiments would require access to many test subjects for real experiments. So we opted out to emulate an outdoor Wi-Fi signal strength monitoring application and measure the performance in terms of the collected data samples and energy efficiency. For the experiments, we consider each block size as $100m \times 100m$. Inside a block, we assume a mobile device can be at any point with a sensing radius of 25m. The data reporting window is considered as 15 seconds. A mobile phone runs sLoc in the background for localization. If it decides to contribute data in a particular data collection round, it performs a WiFi scan and sends the data (scan result) to the server using 3G communication. We assume that size of the data file is 1KB on average. For energy consumption by different sensors, we used the model proposed in [56]. According to that model, a Wi-Fi scan takes 545mJ, GPS takes 1425mJ. For 3G communication, energy consumption ranges between 6000mJ and 12000mJ for 3B and 16KB of data respectively.

We compare the performance of our PLUS framework with two other methods, namely, the *naive method* and the *Minimum Device for Coverage (MDC)*. In the naive method, each mobile device sends the collected data to the server. In the MDC method, the server needs to know the current location of each mobile device. Based on the coverage requirement, it then selects the minimum number of mobile devices to collect data samples. Note, MDC works as the optimal method to achieve partial data coverage when server knows the location of the mobile devices. Figure 8a displays the number of data sample requirement across varying partial coverage. We observe that to ensure the same partial coverage, almost all of the case our approach requires more data samples than the optimal one. Now this is expected, since the optimal method knows the exact location of the participating devices, and chooses the minimum number

of devices to guarantee the coverage requirement. However, our method still has the potential to be energy efficient since it minimizes the expensive server communication.

Figure 8b shows the total energy consumption across the number of mobile devices for a single block by simulating the application for 10 minutes. For both PLUS and MDC methods, we considered two different coverage requirements, namely 70% and 90%. We observe that the naive method consumes more energy than all other methods for most of the cases. This is expected because the naive method collects data from all the devices invariably. The naive method only consumes marginally less energy than the MDC method for 90% coverage when the number of mobile devices is less than 20. To explain the reason, we consider an example where the total number of mobile devices in the block is N_t, and the number of mobile devices selected by the MDC method to collect data is N_s, for a certain coverage requirement. For the MDC method, each of the N_t devices needs to send its location to the server at first. Then only the N_s devices send the collected data to the server. Note, sending the collected data to the server consumes more energy than sending only the location. In the naive method, all the devices need to send data to the server. Therefore, for the MDC method, when N_t is significantly larger than N_s, the total consumed energy is reasonably lower than the naive method. Only when N_t and N_s are same or very close, the MDC method takes more energy than the naive method. On the other hand, in our framework PLUS, a mobile device does not send its location to the server, thus saves significant amount of communication energy. Therefore, for both the considered coverage requirements, PLUS consumes significantly less energy than the MDC method.

5. Discussion

The design of the PLUS framework has several interesting observation and challenges for its applicability that are discussed below.

5.1. Applicability of sLoc in monitoring applications

In the previous section, we discussed how sLoc can be applied to predict future block transitions and minimize the duration of the GPS usage. And we noticed that sLoc can reduce the GPS usage significantly compared to continuously using the GPS when a user follows the same route. However, sLoc is designed to provide continuous block level location information, thus it can reduce GPS usage for real-time continuous monitoring application where data collection frequency is very high. On the other hand, if a monitoring application has a low data collection frequency, applying sLoc can be an overkill, since continuous location information is not required for such application. In such case, using the GPS periodically in each data collection round can be a better option. In the following, we derive the condition when using sLoc can be beneficial. Suppose, the data collection window of an application is Δ_{app}, average time to retrieve a stable location update D_{app}, the average interval between GPS usage

by sLoc is Δ_{sLoc}, the average duration GPS is used each time is D_{sLoc}, the power consumption by the GPS to retrieve location is E^s_{GPS}, total energy saved by GPS in time T can be written as

$$E^s_{sLoc} = \left(\frac{D_{app}}{\Delta_{app}} - \frac{D_{sLoc}}{\Delta_{sLoc}}\right).T.E^s_{GPS} \tag{6}$$

For sLoc to be effective, the right side of Equation 6 needs to be positive. Therefore, we can say,

$$\frac{D_{app}}{\Delta_{app}} > \frac{D_{sLoc}}{\Delta_{sLoc}} = \eta \tag{7}$$

where η is the percentage of GPS usage (as defined in 4.1.2) by sLoc. In our observation, time required to get a stable location D_{app} can be easily 3-5 seconds or more (it is not always same as receiving the first location, since the first received location usually comes with a large uncertainty region). Therefore, based on the data collection interval T_{app} of the monitoring application and Equation 7, we can decide if sLoc can yield energy savings over a simple periodic GPS polling.

5.2. Effect of data reporting window on energy savings

In Section 5.1, we have derived the condition when sLoc can bring in additional savings over simply using the GPS. Besides, there are also other factors that contributes in the energy savings. Now we will derive an expression to represent the energy savings of the PLUS method compared to the MDC method. We can say that, energy savings by PLUS (E) = energy savings by sLoc for localization (E_1) + energy required by MDC method to send location and sensory data (E_2) - energy required by PLUS to send sensory data (E_3). Now according to Equation 6 E_1 can be written as,

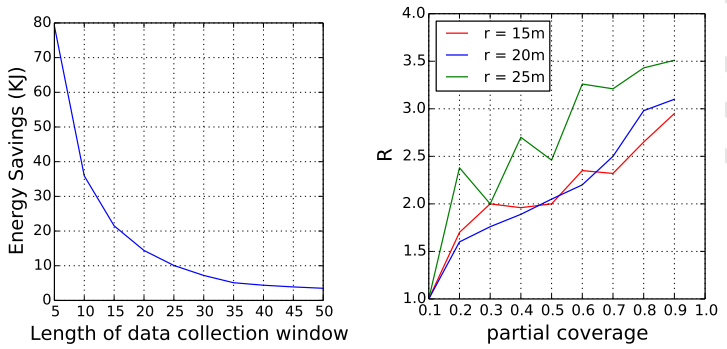
$$E_1 = \frac{N.E^s_{GPS}}{\Delta_{sLoc}}.\frac{T}{\Delta_{app}}\left(D_{app}.\Delta_{sLoc} - D_{sLoc}.\Delta_{app}\right) \tag{8}$$

where, N is the number of participants. Note, the minimum value of E_1 can be 0 basically when sLoc is replaced by a method with periodic GPS usage. Now we can write the following expression for E_2,

$$E_2 = \frac{T}{\Delta_{app}}\left(N.E^c_{GPS} + N_{MDC}\left(E^s_{dat} + E^c_{dat}\right)\right) \tag{9}$$

where N_{MDC} is the number of data samples required by the optimal method, E^c_{GPS} is the energy required to send a location data to the server, E^s_{dat} and E^c_{dat} are the energy required to collect a sensory data sample and to send it the server respectively. Similarly E_3 can be written as,

$$E_3 = \frac{T}{\Delta_{app}}\left(N_{PLUS}\left(E^s_{dat} + E^c_{dat}\right)\right) \tag{10}$$



(a) (b)

Figure 9: (a) Energy savings for varying data reporting window, (b) Ratio between sample requirement by PLUS and MDC (R)

With the help of Equation 8, 11 and 10, we can calculate the value of E. Figure 9a exhibits the impact of varying data reporting window on the energy savings over the MDC method to achieve 70% coverage in a block with 60 users for a duration of 10 minutes. Data reporting window was varied from 5s to 50s. From the figure we observe that the energy savings is quite high for smaller values of data reporting window (or high data collection frequency) as it experiences advantages from both fewer GPS usage by sLoc and reduced communication with the server. With increasing size of data reporting window the benefit from sLoc reduces. However, our method is still able to save a good amount of energy because of the reduced communication overhead. So, we can say that even though our approach is most effective for continuous data collection, it still can be used for application with lower data collection frequency to achieve limited energy efficiency.

5.3. Conservative estimation of necessary participant density

As we have seen in Figure 8b, our proposed data collection approach leverages the high density of the participants in the monitored area to reduce energy consumption. Now we will try to derive a conservative condition for minimum participant density under which our method can operate energy efficiently. As per Equation 11 and 10 (from Section 5.2), we can say that if $E_2 - E_3$ is positive, our data collection approach can operate energy efficiently. Thus the necessary condition becomes

$$N.E^c_{GPS} - (E^s_{dat} + E^c_{dat})(N_{PLUS} - N_{MDC}) > 0 \tag{11}$$

By assuming $E^s_{dat} + E^c_{dat} = L.E^c_{GPS}$ and $\frac{N_{PLUS}}{N_{MDC}} = R$, we get

$$\frac{N}{N_{OPT}} > L(R-1) \tag{12}$$

Here, L depends on the sensor being used for data collection and the amount of sensed data to send to the server. According to the energy model we used, in

most cases the value should be between 1.5 to 2. On the other hand, R depends on the block size, the desired level of coverage and the sensing radius of the sensor. Figure 9b shows the values of R by varying coverage requirement and for different sensing radius in a $100m \times 100m$ block where we observe that R stays between 2 to 3.5 in most cases. Thus, we can have an idea of when our approach works energy efficiently irrespective of the data collection frequency.

5.4. Impact of localization error on data collection

In our framework the achievable coverage is directly proportional to the number of participants present in a block in a given time window, hence having an accurate idea on the participant density is important for successful data collection. Recent research shows that the population density can be accurately determined even for small area such as $100m \times 100m$ blocks with temporal effect by using mobile call data collected from cellular towers [49]. Therefore determining the population density is out of the scope of this paper. However, we want to discuss the impact of localization error on the data collection process. For example, if a user is in block b1 but the localization scheme detects it as b2, the user would upload data sample with wrong probability and tag the data with wrong block ID. One simple approach could be increase the data upload probability so that the effect of the erroneous data samples are minimized. One could also apply an anomaly detection based technique to filter out the erroneous data.

5.5. Continuous monitoring application and cell phone battery life

Continuous monitoring applications have a lot of prospect and applicability especially to know fine-grained and time-sensitive information. For example, how can we know the fine grained temperature of a city? The authorities may publish some average results based on a few major points. But what if we want to know the temperature in the shade by a waterfront where the temperature could be much lower, or temperature in the subway station right now which can be much higher. Or how can we know the pollen count around me right now? Or how much it is around my office where I am headed to? How many opportunistic contacts are available around me? With the help of continuous participatory monitoring, we can find answers for this type of extremely difficult questions. However, deploying such an application can put an adverse affect on the battery life. Note, continuous localization is an extremely challenging problem itself since it consumes a lot of energy. And real-time continuous participatory monitoring applications includes an additional energy consuming task of frequent sensing along with continuous localization. Our work makes an effort to increase the energy efficiency of such data collection application. No matter how much optimization a technique (like ours) does, it still has to awake the smartphone to collect sensory data and location information somewhat frequently, hence stopping the smartphone to go into prolonged sleep mode. Some recent work were proposed to address this challenge where an additional lightweight processor is used to perform this type of sensing tasks and allows the main processor to go

to sleep mode [57, 58]. We would advocate that such a technique would help our data collection approach to be more efficient in terms of battery life.

6. Conclusion

In this paper, we presented PLUS, an energy efficient framework for localization and coverage in participatory urban sensing. Using PLUS, a continuous monitoring application can specify a desired partial coverage requirement of data. The framework performs data collection intelligently to achieve the required coverage while minimizing the energy consumption for localization and communication with the server. We also presented a localization scheme called sLoc to determine the block level location of the user in an energy efficient way. Real world experiments showed that sLoc can save a good amount of energy when a user follows regular routes. Finally, by emulating a continuous monitoring application, we showed that PLUS can save significant amount of energy of the mobile devices compared to traditional methods. Followed by this, another work has been presented in

Although the proposed framework is novel and energy efficient, it still has scope for further improvements. For example, our work excludes indoor event monitoring. How to address such monitoring application? Also how to define coverage metric for a multi-storied indoor environment and collect data to ensure the coverage? One possibility is to define and ensure coverage metric for each floor. For localization in indoor monitoring, one possibility is to use a low cost Wi-Fi signal strength matching based approach. Another possible research direction is to explore the possibility of using passive locations retrieved by other apps to improve energy efficiency. Someone could also investigate how to collect data samples from vehicular users. Specifically how the coverage metric and localization scheme need to be modified in such scenario? In future, we plan to investigate some of these issues.

References

[1] Apple Watch, `https://www.apple.com/watch/` (retrieved March, 2015).

[2] Samsung Gear, `http://www.samsung.com/us/mobile/wearable-tech` (retrieved March, 2015).

[3] iHealth Edge, `http://www.ihealthlabs.com/fitness-devices/ihealth-edge/` (retrieved March, 2015).

[4] O. D. Incel, M. Kose, C. Ersoy, A review and taxonomy of activity recognition on mobile phones, BioNanoScience 3 (2) (2013) 145–171.

[5] J.-K. Min, A. Doryab, J. Wiese, S. Amini, J. Zimmerman, J. I. Hong, Toss'n'turn: smartphone as sleep and sleep quality detector, in: Proceedings of the 32nd annual ACM conference on Human factors in computing systems, ACM, 2014, pp. 477–486.

[6] N. D. Lane, M. Mohammod, M. Lin, X. Yang, H. Lu, S. Ali, A. Doryab, E. Berke, T. Choudhury, A. Campbell, Bewell: A smartphone application to monitor, model and promote wellbeing, in: 5th international ICST conference on pervasive computing technologies for healthcare, 2011, pp. 23–26.

[7] S. Reddy, A. Parker, J. Hyman, J. Burke, D. Estrin, M. Hansen, Image browsing, processing, and clustering for participatory sensing: lessons from a DietSense prototype, in: Proceedings of the 4th workshop on Embedded networked sensors, ACM, 2007, pp. 13–17.

[8] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, M. B. Srivastava, Participatory sensing, ACM Sensys World Sensor Web Workshop, 2006.
[9] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, J. Reich, Mobiscopes for human spaces, IEEE Pervasive Computing.
[10] P. Baier, H. Weinschrott, F. Durr, K. Rothermel, MapCorrect: automatic correction and validation of road maps using public sensing, in: LCN, 2011.
[11] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, A. Sharma, PRISM: platform for remote sensing using smartphones, in: MobiSys, 2010.
[12] S. S. Kanhere, Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces, in: 2011 12th IEEE International Conference on Mobile Data Management (MDM), Vol. 2, 2011, pp. 3–6.
[13] T. Yardibi, E. Karasan, A distributed activity scheduling algorithm for wireless sensor networks with partial coverage, Wireless Networks 16 (1) (2010) 213–225.
[14] Y. Mao, Z. Wang, Y. Liang, Energy aware partial coverage protocol in wireless sensor networks, in: IEEE International Conference on Wireless Communications, Networking and Mobile Computing (WiCom), 2007., 2007, pp. 2535–2538.
[15] M. Di Francesco, S. K. Das, G. Anastasi, Data collection in wireless sensor networks with mobile elements: A survey, ACM Transactions on Sensor Networks (TOSN).
[16] A. Ghosh, S. K. Das, Coverage and connectivity issues in wireless sensor networks: A survey, Pervasive and Mobile Computing 4 (3) (2008) 303–334.
[17] W. Choi, G. Ghidini, S. K. Das, A novel framework for energy-efficient data gathering with random coverage in wireless sensor networks, ACM Transactions on Sensor Networks (TOSN).
[18] M. Di Francesco, S. K. Das, G. Anastasi, Data collection in wireless sensor networks with mobile elements: A survey, ACM Transactions on Sensor Networks (TOSN) 8 (1) (2011) 7.
[19] S. K. A. Imon, A. Khan, M. Di Francesco, S. K. Das, Rasmalai: A randomized switching algorithm for maximizing lifetime in tree-based wireless sensor networks, in: IEEE INFOCOM, 2013, pp. 2913–2921.
[20] S. Imon, A. Khan, M. Di Francesco, S. Das, Energy-efficient randomized switching for maximizing lifetime in tree-based wireless sensor networks, IEEE/ACM Transactions on Networking PP (99). doi:10.1109/TNET.2014.2331178.
[21] S. K. A. Imon, A. Khan, S. K. Das, Effect: An energy efficient framework for data compression in tree-based wireless sensor networks 1–9.
[22] X. Lu, D. Li, B. Xu, W. Chen, Z. Ding, Minimum cost collaborative sensing network with mobile phones, in: 2013 IEEE International Conference on Communications (ICC), IEEE, 2013, pp. 1816–1820.
[23] X. Sheng, J. Tang, W. Zhang, Energy-efficient collaborative sensing with mobile phones, in: IEEE INFOCOM, 2012.
[24] P. Baier, F. Durr, K. Rothermel, PSense: Reducing Energy Consumption in Public Sensing Systems, in: IEEE 26th Conf. on Advanced Information Networking and Applications (AINA), 2012.
[25] P. Rudman, S. North, M. Chalmers, Mobile pollution mapping in the city, in: UK-UbiNet Workshop on eScience and Ubicomp, 2005.
[26] H. Weinschrott, F. Durr, K. Rothermel, StreamShaper: Coordination algorithms for participatory mobile urban sensing, in: IEEE MASS, 2010.
[27] A. Khan, S. K. A. Imon, S. K. Das, An energy efficient framework for localization and coverage in participatory urban sensing, in: IEEE 39th Conference on Local Computer Networks (LCN), 2014, pp. 193–201.
[28] W. Choi, S. K. Das, A novel framework for energy-conserving data gathering in wireless sensor networks, in: IEEE INFOCOM, 2005.
[29] W. Z. Khan, Y. Xiang, M. Y. Aalsalem, Q. Arshad, Mobile phone sensing systems: A survey, Communications Surveys & Tutorials, IEEE 15 (1) (2013) 402–427.
[30] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, A. T. Campbell, A survey of mobile phone sensing, IEEE Communications Magazine.
[31] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, A. T. Campbell, The Jigsaw continuous sensing engine for mobile phone applications, in: Proceedings of the 8th ACM Conference

on Embedded Networked Sensor Systems, ACM, 2010, pp. 71–84.
[32] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, K. Aberer, Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach, in: Wearable Computers (ISWC), 2012 16th International Symposium on, Ieee, 2012, pp. 17–24.
[33] S. Nath, ACE: exploiting correlation for energy-efficient and continuous context sensing, in: Proceedings of the 10th international conference on Mobile systems, applications, and services, ACM, 2012, pp. 29–42.
[34] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, People-centric urban sensing, in: Proceedings of the 2nd annual international workshop on Wireless internet, ACM, 2006, p. 18.
[35] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, A. T. Campbell, BikeNet: A mobile sensing system for cyclist experience mapping, ACM Transactions on Sensor Networks (TOSN) 6 (1) (2009) 6.
[36] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, A. T. Campbell, Sensing meets mobile social networks: the design, implementation and evaluation of the CenceMe application, in: ACM Conf. on Embedded Network Sensor Systems, 2008.
[37] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, W. 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, ACM, 2010, pp. 105–116.
[38] A. Balasubramanian, R. Mahajan, A. Venkataramani, Augmenting mobile 3g using wifi, in: Proceedings of the 8th international conference on Mobile systems, applications, and services, ACM, 2010, pp. 209–222.
[39] M. Shahriar, Y. Liu, S. K. Das, Hipcv: A history based learning model for predicting contact volume in opportunistic network, in: A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on, IEEE, 2015.
[40] N. D. Lane, Y. Chon, L. Zhou, Y. Zhang, F. Li, D. Kim, G. Ding, F. Zhao, H. Cha, Piggyback crowdsensing (pcs): energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities, in: Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, ACM, 2013, p. 7.
[41] D. Zhang, H. Xiong, L. Wang, G. Chen, CrowdRecruiter: selecting participants for piggyback crowdsensing under probabilistic coverage constraint, in: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, ACM, 2014, pp. 703–714.
[42] E. Miluzzo, M. Papandrea, N. D. Lane, A. M. Sarroff, S. Giordano, A. T. Campbell, Tapping into the vibe of the city using vibn, a continuous sensing application for smartphones, in: Proceedings of 1st international symposium on From digital footprints to social and community intelligence, ACM, 2011, pp. 13–18.
[43] E. Kanjo, S. Benford, M. Paxton, A. Chamberlain, D. S. Fraser, D. Woodgate, D. Crellin, A. Woolard, MobGeoSen: facilitating personal geosensor data collection and visualization using mobile phones, Personal and Ubiquitous Computing 12 (8) (2008) 599–607.
[44] H. Weinschrott, J. Weisser, F. Durr, K. Rothermel, Participatory sensing algorithms for mobile object discovery in urban areas, in: Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on, IEEE, 2011, pp. 128–135.
[45] G. Tychogiorgos, C. Bisdikian, Selecting relevant sensor providers for meeting "your" quality information needs, in: Mobile Data Management (MDM), 2011 12th IEEE International Conference on, Vol. 1, IEEE, 2011, pp. 200–205.
[46] M. Shahriar, S. K. Das, Connect: Consociating opportunistic network neighbors for constructing a consistent and connected virtual backbone, in: 2014 IEEE WoWMoM, pp. 1–9.
[47] A. Khan, S. K. A. Imon, S. K. Das, Ensuring energy efficient coverage for participatory sensing in urban streets, in: IEEE International conference on Smart Computing (Smartcomp), 2014.
[48] Airsage, `http://www.airsage.com` (retrieved July, 2013).
[49] P. Deville, C. Linard, S. Martin, M. Gilbert, F. R. Stevens, A. E. Gaughan, V. D. Blondel, A. J. Tatem, Dynamic population mapping using mobile phone data, Proceedings of the

National Academy of Sciences 111 (45) (2014) 15888–15893.
[50] I. F. Akyildiz, W. Wang, The predictive user mobility profile framework for wireless multimedia networks, IEEE/ACM Transactions on Networking (TON).
[51] L. Song, D. Kotz, R. Jain, X. He, Evaluating location predictors with extensive wi-fi mobility data, in: IEEE INFOCOM, 2004.
[52] J. Paek, J. Kim, R. Govindan, Energy-efficient rate-adaptive gps-based positioning for smartphones, in: Proceedings of MobiSys, 2010.
[53] A. Barto, P. Anandan, et al., Pattern-recognizing stochastic learning automata, IEEE Transactions on Systems, Man and Cybernetics.
[54] V. Gullapalli, A stochastic reinforcement learning algorithm for learning real-valued functions, Neural networks.
[55] V. Gullapalli, Associative reinforcement learning of real-valued functions, in: IEEE Intl. Conference on Decision Aiding for Complex Systems, 1991.
[56] K. Lin, A. Kansal, D. Lymberopoulos, F. Zhao, Energy-accuracy trade-off for continuous mobile device location, in: MobiSys, 2010.
[57] B. Priyantha, D. Lymberopoulos, J. Liu, Littlerock: Enabling energy-efficient continuous sensing on mobile phones, Pervasive Computing, IEEE 10 (2) (2011) 12–15.
[58] P. Georgiev, N. D. Lane, K. K. Rachuri, C. Mascolo, DSP. Ear: leveraging co-processor support for continuous audio sensing on smartphones, in: Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems, ACM, 2014, pp. 295–309.