Long-term Privacy Profiling through Smartphone Sensors

Ningning Cheng, Shaxun Chen, Parth Pathak, Prasant Mohapatra
Department of Computer Science, University of California, Davis,
Email: {nincheng, sxch, phpathak, pmohapatra}@ucdavis.edu

***Abstract*—Smartphones are closely coupled with their users and smartphone sensors can perceive users' private information. The existing studies in this area focus on user activity recognition and short-term context detection. In this paper, we show that smartphone sensors are able to profile users' long-term privacy and more sensitive information. We present the techniques of discovering users' spending level by merely using smartphone sensors. We do not access users' contacts, calendar, or call log, so that the profiling is performed in a non-intrusive manner. This paper is an alert towards the public that the privacy leakage could be far worse than imagination by just carrying smartphones.**

I. Introduction

Smartphones have become an indispensable part in our daily lives. People use smartphones to interact with friends and family, purchase online merchandise, and search for interested information. On the other hand, today's smartphones are equipped with a rich set of sensors, such as ambient light sensor, proximity sensor, accelerometers, gyroscopes and digital compass. GPS is also referred to as the positioning sensor. These sensors are able to get significant personal information from their owners.

Some existing studies have paid attention to the privacy leakage from smartphone sensors [1]–[3]. For example, Anguita et al proposed a smartphone-based platform that can detect user activities such as sitting, walking, going upstairs and downstairs [2]. Keally et al focused on profiling the environment users currently stay in, such as home, office, and bar [4]. However, they mainly focus on user activity recognition and short-term context detection. Smartphones are much more powerful. It is possible to expose more sensitive information by just habitually carrying them everyday.

In this paper, we present the technique which can profile a user's long-term privacy: spending level. Compared to users' instant activities and current location/environment, spending level is long-term private information and typically more sensitive.

On the other hand, long-term privacy is much more difficult to profile. A transient human activity is physical movements of human body and thus closely related to the motion sensor readings. Short-term context (like having a meeting or staying in a bar) is slightly more complex, because it has higher abstract level. Long-term context, such as spending level to be discussed in this paper, is at an even-higher level, whose relationship to the raw sensor data is more indirect and subtle. Therefore, the profiling of such long-term privacy is highly challenging.

Our method avoids the usage of sensitive smartphone sensors such as camera and microphone. We also do not visit contacts, calendar and call logs in the phone. The GPS access is intentionally minimized as well. As a result, our method performs the profiling in a non-intrusive manner and is battery-efficient. To the best of our knowledge, our paper is the first effort to profile user's spending level merely using smartphone sensors.

The purpose of this paper is to raise an alert towards the users and the public that if a smartphone app is malicious, the privacy leakage could be far worse than imagination. Even if never using a banking/finance app or shopping recommendation app, a user's spending level and financial status could still be leaked. Users usually treat their financial documents in a discreet manner, but if they are unaware of such abilities of smartphone sensors, their privacy can also be in danger.

We conduct extensive experiments to evaluate our proposed method using real-world data. Smartphone sensor readings from 9 volunteers during two months are used in our evaluation. The results show that our method can effectively profile smartphone users' long-term privacy without users' intended input.

The rest of the paper is organized as follows. We survey the related work in Section II. The details of our methods used to profile spending level is described in Section III. Section IV evaluates our work and Section V discusses related issues. Section VI concludes the paper.

II. Related Work

Embedded with multiple motion sensors, smartphones have been intensively utilized for detecting a variety of daily activities such as sitting, standing, walking, and running, going upstairs/downstairs [5], [6]. Anguita et al introduced a light weighted SVM on smartphones for activity recognition [2]. Kwapisz et al used three data mining techniques, decision tree (J48), logistic regression and multilayer neural networks, to detect daily activities [7].

Walking detection, as a subset of activity recognition, has been explored extensively [8]. The techniques include peak detection [9], dynamic time warping [10], frequency analysis [11] and machine learning [2] [7]. Generally speaking, machine learning based methods tend to have higher accuracy but are more computational expensive and battery consuming. Driving recognition is relatively less studied. Johnson et al proposed to use accelerometer and gyroscope to detect aggressive driving [12].

Short-term context detection mainly focus on users' surrounding environment [3], including physical working environment recognition [13], and home/office detection [4] [14]. Some researchers are interested in inferring social interactions such as discover an individual or groups of people [15]. The *Reality Mining* utilizes the bluetooth signals to discover proximity social groups and friendship relationships [15].

A number of studies have been done to record users' traces using their smartphones. *Google Now* logs the trace by utilizing Android's locations service when "report location" is switched on. In *LifeMap*, a location context provider utilizes aggregated information from accelerometer, digital compass, WiFi, and GPS [1] to provide room-level traces. However, these studies record all the traces without differentiating the purpose of the trip.

Our work is substantially different from aforementioned work in that: first, we aim at long-term user profiling instead of instant activities or environment recognition; second, we avoid the use of sensitive sensors/information such as calendar,

call log, camera or microphone and limit the access of GPS, which makes the profiling process less intrusive and more power-efficient; third, we propose *getting-off-car detection* and *loading-merchandise detection*, which is able to identify shopping trips out of other user traces.

III. User Spending Level Profiling

A. Overview

In this paper, we define *spending level* as the average price level of the merchandise that a user purchases. Informally speaking, a user who has high spending level tends to buy upscale products, while a user with low spending level prefers to buy less expensive commodities. User spending level is very sensitive privacy information. People may be aware of their privacy leakage when the information is shown explicitly, such as using financial apps, shopping apps or banking apps, and hence use them in a discreet manner. However, few has raised alert that their financial status could also be leaked from their routine behaviors simply by carrying the smartphone, even without installing any apps mentioned above.

The purpose of this section is to quantify users' spending level by merely using smartphone sensors in a non-intrusive manner. Our method works as follows. First, we try to detect the user's action of getting off the car, based on which we learn the supermarkets, grocery stores and shopping centers the user frequently go to. Then, we match these places to a review system (e.g., *Yelp*) to get their price level information. Combining the places, visiting frequency and price level as input, we are able to determine the spending level of the user.

B. Getting-Off-Car Detection

Getting-off-car detection is achieved by identifying the transition from driving to walking using smartphones' motion sensors. Only when such a transition is detected, we activate the smartphone's location service.

Existing works on users' trace profiling or location discovering are based on continuous GPS readings, some of which also utilize smartphone's WiFi hotspot based location service [1]. In our method, instead, we first detect the *getting-off-car* event through motion sensor readings, and then detect the location where the user get off the car.

We use this method for two reasons. First, the power consumption of sampling motion sensors is much lower than accessing GPS, which is very important in covert profiling (see Section V-B). In our method, we only need to activate the location service for a very short period of time once the *getting-off-car* event is detected. Second, in U.S., most people drive their cars for intended shopping. Indiscriminately recording user's trace is not very helpful to determine the shopping places, because people may just go for a walk and it happens to be close to or inside a mall. Besides, we prefer to use grocery purchase, especially relatively big grocery purchases to perform the profiling (Grocery purchases are essential for all the users, thus can better reveal the spending level. Big purchase also helps rule out outliers). Therefore, in our method, we make use of the transportation method, combining with stay duration and loading detection (which will be discussed later), to help detect the events of shopping more accurately.

Getting-off-car is detected when a driving pattern is followed by a walking pattern. It is based on the fact that most users walk after getting off their cars. A walking pattern is a series of stepfalls which results in a periodic change of the motion sensor signal. Fig. 1 shows an example of the walk signal read from 3 axes of the smartphone accelerometer. In the figure, x-axis is the index of sensor readings (sampling rate is 20Hz) and y-axis shows the acceleration. When driving a car, periodic signals are less significant, which can stem from pavement patterns, engine vibration, imperfection of car wheels and other factors. A typical accelerometer reading from driving is shown in Fig. 2.

First, we apply a low pass filter to the sensor data:

$$\vec{O}_{t_i} = \alpha \vec{A}_{t_i} + (1-\alpha)\vec{A}_{t_{i-1}} \tag{1}$$

where $\vec{A}$ is the accelerometer vector which has three components x, y and z, standing for accelerations along three axes respectively (we read them from Android TYPE_LINEAR_ACCELERATION, in which gravity has already been removed), $\vec{O}$ is the filter's output, t_i ($i \in Z^+$) are temporal index of sampled sensor data. Here we use exponentially-weighted moving average as a low-pass filter. α in Equation 1 is a smoothing factor ($0 \leq \alpha \leq 1$) defined as

$$\alpha = \frac{\Delta t}{T + \Delta t} \tag{2}$$

where Δt stands for $t_i - t_{i-1}$. T is a time constant. The larger T is, the more high-frequency components are filtered out. The best T will be determined by experiments (please refer to Section IV-A).

The output of the filtered walking signal and driving signal is shown in Fig. 3 and Fig. 4 respectively (here T is set to $7\Delta t$). We can see that the walking signal still exhibits significant periodic pattern while the driving signal becomes close to a straight line. Inspired by it, we use the following heuristics to identify the state of driving. If the original accelerometer signal has non-negligible value, but after filtering, it is close to zero, we assume the user is driving. We carefully investigate two months' sensor data from 9 volunteers, among all the cases that have near-zero low-frequency component (such as sitting, standing still, driving and when the phone is not carried by the user), only driving has non-negligible high frequency components. Therefore, this heuristics is simple but effective.

For the walking detection, in order to tolerate smartphone angle change during users' actions, we first calculate the acceleration magnitude by Equation 3 instead of using the value of a single axis.

$$O_{t_i} = \sqrt{(\overrightarrow{O}^x_{t_i})^2 + (\overrightarrow{O}^y_{t_i})^2 + (\overrightarrow{O}^z_{t_i})^2} \tag{3}$$

where $\overrightarrow{O}^x_{t_i}$ is the *x* component of $\overrightarrow{O}_{t_i}$, and similarly for $\overrightarrow{O}^y_{t_i}$ and $\overrightarrow{O}^z_{t_i}$. Then we take derivative of *O*. As we know, *O* reaches its local minimum or maximum when *O*'s derivative equals zero. If for all *i*, $t_i - t_{i-1}$ is a constant (i.e. sensor sampling rate does not change), we can use the following approximation to calculate the derivative of *O* [16]:

$$O'_{t_i} = \frac{1}{8}[2O_{t_i} + O_{t_{i-1}} - O_{t_{i-3}} - 2O_{t_{i-4}}] \tag{4}$$

O'_{t_i} is a discrete function of *t*, which can be presented as a sequence of points on the plain. We use line segments to join all the adjacent point pairs and denote the new function as $\mathcal{O}'_t$. Define H_n as the n^{th} non-negative root of the equation $\mathcal{O}'_t = 0$, such that $|O(H_n)| > 3m/s^2$. In fact, H_n indicates the values of *t* making *O* reaches its local minimum (trough) and maximum (crest), given that small wave troughs and crests are filtered out. Then we calculate:

$$S_j = H_j - H_{j-2} \tag{5}$$

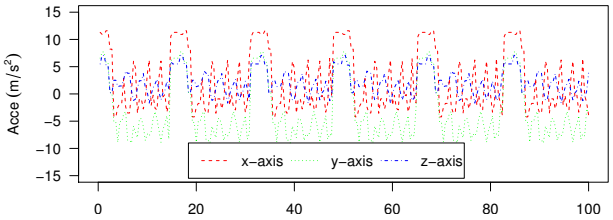

Fig. 1. The x-axis, y-axis and z-axis of acceleration data during walking.

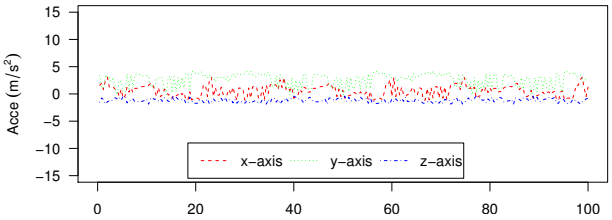

Fig. 2. The x-axis, y-axis and z-axis of acceleration data during driving.

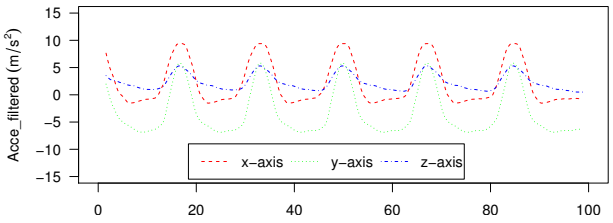

Fig. 3. The x-axis, y-axis and z-axis of filtered data during walking.

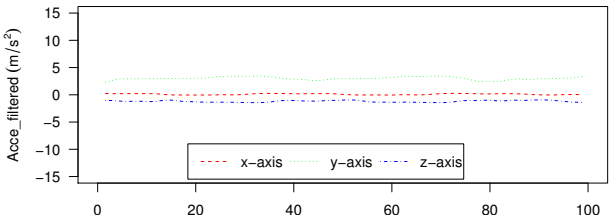

Fig. 4. The x-axis, y-axis and z-axis of filtered data during driving.

If the signal is from a walking user, S_j presents the time duration of one step. We compare the difference of five consecutive step durations. If their differences are within a threshold T_s and each S_j falls into [0.4, 0.7] (Inman et al [17] investigated a large number of adults, and pointed out that most of them has a step frequency of 100 to 120 per minute, which translates to 0.5s to 0.6s per step), the state of walking is assumed. The choice of T_s is determined in Section IV-A.

We have discussed the detection of driving and walking respectively. If the state of driving is followed by walking, a getting-off-car event is reported. The method we use in this subsection is intuitive and based on numerical analysis of motion sensor readings. Compared with machine learning methods [2] [7], our method provides comparable accuracy but is much lower in computational complexity, thus saves battery power (Section V-B discusses the importance of battery efficiency for covert privacy profiling).

C. Location Mapping & Shopping Detection

As soon as a getting-off-car event is detected, we activate the location service of the smartphone. The current location (a $(latitude, longitude)$ pair) is recorded. We note this location as L^i. Then the location service is turned off to save energy.

Based on L^i, point of interest (POI) can be determined through location mapping services. Take *Google Places* for example, using $(latitude, longitude)$ as input, it will return the best match or a list of the POI. Through the *type* field of the response, we can determine the type of POI. If it is not a business zone (e.g., $type$ is park, residence community, etc.) or it happen to be the user's home or working location (we will discuss the determination of users' home and working location in Section V-A), it will not be used in spending level profiling. Otherwise, the response is recorded as R^i for future use, and the loading detection will be performed as follows.

Starting from 15 minutes and ending at 120 minutes after a getting-off-car event is observed, we detect *loading event* using the motion sensors of the smartphone. The *loading event* is the repetitive action that a user takes the items from the shopping cart and puts them into the car trunk. As mentioned previously, for user spending level estimation, we are interested in grocery shopping, especially relatively big grocery purchase (see Section III-B). In these cases, it is very likely a user will use a shopping cart and then take the merchandise home in their car trunk. Therefore, we propose to detect loading events to help identify users' shopping behavior. We do not detect loading events in the first 15 minutes (after getting-off-car event) because quick shopping or relatively small purchases are not of our interest.

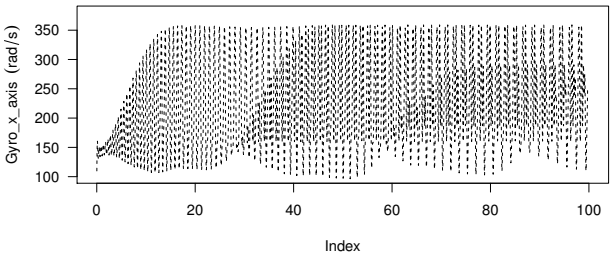

Fig. 5. The x-axis of gyroscope data when loading grocery.

In a loading event, users usually bend over to take the items from the cart and then turn over to put them into the trunk, which will typically repeat multiple times. Figure 5 shows the reading of the gyroscope when a user is loading merchandise to his car trunk (x-axis is sensor reading index and y-axis stands for angular acceleration). We can see a periodic pattern of the signal envelope in the plot. However, the loading action can vary from user to user. Different user may have their own habitual movements. Besides, the place of the smartphone (e.g., in pants pocket or in chest pocket) and the vehicle type may also affect patterns extracted from sensor readings. Since loading event is much more complex than walking and driving, we will detect it using a SVM (support vector machine).

As mentioned before, machine learning based methods are more energy expensive than directly performing numerical analysis on sensor readings. However, we would like to point out that in our method, the getting-off-car detection needs to be performed all day long, while the loading detection is only activated for a maximum of 105 minutes after each

getting-off-car event. Therefore, the efforts above to reduce the power consumption of walking and driving detection are still important. For the loading detection, we sacrifice a little battery life to achieve satisfactory accuracy.

Due to space limit, we do not detail the implementation of our SVM but only present its input and output. In our approach, the input of the SVM are $\overrightarrow{A}^x$, $\overrightarrow{A}^y$, $\overrightarrow{A}^z$, $\overrightarrow{O}^x$, $\overrightarrow{O}^y$, $\overrightarrow{O}^z$, $\overrightarrow{G}^x$, $\overrightarrow{G}^y$, $\overrightarrow{G}^z$ and δC, where $\overrightarrow{A}^x$ is the *x* component of accelerometer reading, $\overrightarrow{O}^x$ is low-pass filtered $\overrightarrow{A}^x$, $\overrightarrow{G}^x$ is the *x* component of gyroscope reading (similarly for *y* and *z* components), δC is the change of compass reading. We put emphasis on gyroscope and compass readings in that the loading process contains significant bending and turning actions. For each aforementioned entry, we sample 10 times a second, and use 2.5 seconds' data as one instance. That is, for each instance, it contains 250 attributes (10*25). In the training phase, each instance has a label, which is either *loading* or *not-loading*. In the prediction phase, sensor readings collected in real-time will be input into the SVM. Two consecutive samples are overlapped by 50 percent. That is, there are 8 inputs in every ten seconds. For each input instance, a label will be assigned, which is the output of the SVM. In our settings, within each ten-second time slot, if more than 50% of the predicted labels are *loading*, we assume the user is loading merchandise within this period of time. The accuracy of our method will be evaluated in Section IV-B.

After a loading event is assumed, we keep detecting until during any continuous 30 seconds, the *loading* labels are below 20% of all the predictions. If a driving event is detected or the 120-minutes limit has been reached, the loading detection also terminates. All the detected loading time is added together, noted as T^i (loading time spent in location L^i) for future use.

D. Spending Level Calculation

Based on the information obtained above, we now describe our spending level estimation method. For each getting-off-car event, if it is not followed by a loading event (within a period of time), we simply omit it. Otherwise, the loading time is noted as T^i and the response from the location mapping service is noted as R^i (please refer to Section III-C); we use them to estimate user spending level.

If R^i only contains a single POI, we denote its price level as P^i. If R^i contains a list of POIs, we pick up the stores in the list with the $type$ grocery or $type$ supermarket and calculate their average price level as P^i. The rationale is that in a single mall, there usually will not be many grocery stores or supermarkets, and it is most likely the user has shopped in a grocery store or supermarket because repeated loading actions are detected. If the list length is more than one and contains no grocery store or supermarket, we use the average price level of all the businesses in R^i to determine P^i. We do this based on the fact that co-located stores tend to have close price levels [18]. The price level of a business can be directly retrieved from the *price level* field of *Google Places* response. However, the fact is that, in *Google Places*, this field is blank for quite a few businesses. In our method, we retrieve the *location name* field from *Google Places*, and then match the name in *Yelp* and get its *price range* value, because *Yelp* has more complete price information. The spending level is calculated as follows:

$$SL = \frac{\sum_i T^i \cdot P^i}{\sum_i T^i} \tag{6}$$

where SL stands for spending level. i is the i^{th} getting-off-car event detected from a certain user. The basic idea is that the number of items bought should be approximately proportional to the loading time, and we use the number of items as the weight of each store. If no loading action is detected after a getting-off-car event, T^i is 0. Given that P^i is a variable between 1 and 4 (corresponding to $ and $$$$ in *Yelp*), the calculated SL is also a value between 1 and 4. The larger SL, the higher the user's spending level is. When i increases (i.e., sensor data is collected for a sufficiently long time), SL can report the user's spending level more accurately.

IV. Evaluation

In this section, we use real-world data to evaluate our proposed method. We develop an Android app which can collect smartphone sensor readings in the background. The accelerometer, gyroscope and digital compass readings are sampled at the rate of 20Hz, and GPS reading is only logged when *getting-off-car event* is detected.

We have nine volunteers, all of whom use an Android smartphone as their primary cellphone. The models include Samsung Nexus S, Galaxy Note 2, Galaxy S4, HTC One, and LG G2. Android version varies from 4.0.2 to 4.4.2. We installed the app on their phones and it runs on startup. The volunteers tagged all the important activities such as *getting-off-car event* and $loading$ event (wrote down the time and type of the event). Some of the walking, running, sitting and bicycling states are also tagged. They also agree to share the names of the grocery stores they shopped at and the number of items they bought from these stores during these two months.

A. Accuracy of Getting-off-car Detection

Getting-off-car detection is an important step in our spending level profiling mehod. It is also the key of the energy-efficient shopping place discovery. In this part, we first evaluate the accuracy of our method on walking detection, and then we present the result of getting-off-car detection.

We cut 210 ten-second fragments of sensor readings from 9 volunteers' (approximately 24 fragments from each), in which 90 are tagged as walking, and the rest 120 are other activities (such as sitting, bicycling, loading and driving, etc.). We apply our approach (T is set to $7\Delta t$) and vary the parameter T_s (time difference between consecutive steps, see Section III-B). The ROC plot of our walking detection method is shown in Figure 6. Here, y-axis shows true positive rate(TPR), which is defined as: $TPR = (True\,Positive)/(True\,Positive + False\,Negative)$, and x-axis stands for false positive rate (FPR), which is $FPR = (False\,Positive)/(False\,Positive + True\,Negative)$.

In this test, true positives stand for the data instances that are in fact walking and our method reports correctly; false negatives are the walking data instances our method fails to report; In an ROC plot, the dots in upper left corner implies good accuracy. The different values of T_s are labeled besides the dots in the plot (unit: millisecond). As we can see, when T_s is small, false positive rate is low but we miss quite a few real walking instances. When T_s is relatively large, we can detect most of walking cases, but also include more false positives. Combining the results, we choose T_s = 200ms (TPR = 0.9, FPR = 0.083). Its detection accuracy is 91%, which can be derived as: $ACC = (TPR + (1 - FPR) \cdot N/P)/(1 + N/P)$, where N/P is the ratio of the number negatives over the number of positives in the original data set. We will set T_s = 200ms hereinafter unless otherwise specified.

Next, we test the accuracy of our getting-off-car detection method. There are 78 positives and 159 negatives in our test data set (evenly from 9 volunteers). We vary the value of T,

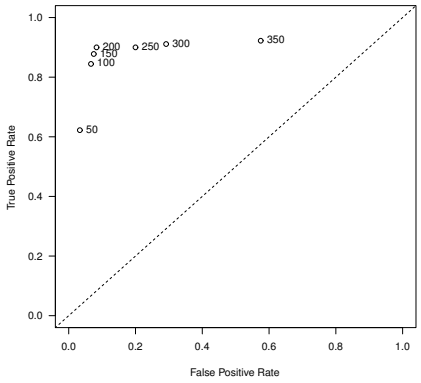

Fig. 6. The accuracy of walking detection.

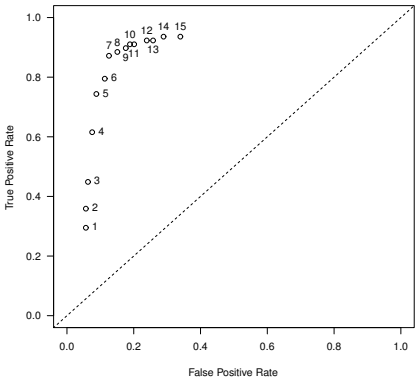

Fig. 7. The accuracy of getting-off-car detection.

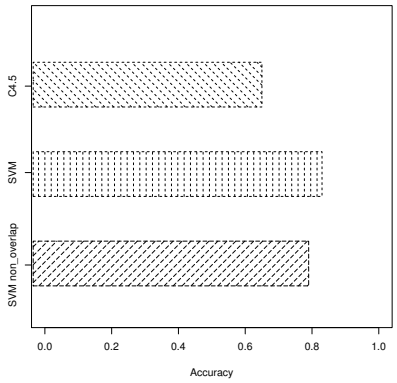

Fig. 8. Loading accuracy comparison.

which indicates the cutoff frequency of the low-pass sensor filter (see Section III-B), and the ROC plot is shown as Figure 7. The number besides the dots in the plot is the multiples of Δt. For example, 5 means $T = 5\Delta t$. When T has a small value, due to unfiltered random acceleration and sensor noise, it is difficult to successfully recognize walking and driving, which results in a low true positive rate of getting-off-car detection. When T is large, signals tend to be oversmoothed, so that more negative signals will be recognized as driving, which increases the false positives. We choose $T = 7\Delta t$ for our method to balance two effects hereinafter. When $T = 7\Delta t$, TPR is 0.867 and FPR is 0.093, so the accuracy will be 88.6%.

From the experiment results above, we can see that although our method is based on straightforward numerical analysis on raw sensor readings, it can achieve a satisfactory detection accuracy, which is sufficient for our succeeding processing.

B. Performance of Spending Level Profiling

Now we move on to evaluate the overall performance of our spending level profiling method. We first take a look at the accuracy of our SVM-based loading detection (see Section III-C). We collect first 3 weeks' data from our 9 volunteers (data set I). Among these data, we have 40 tagged loading events, which are used as the training samples of our SVM. In the following five and a half weeks (data set II), there are 82 tagged loading events. Our trained SVM successfully detected 68 of them on-the-fly. The accuracy is plotted as the second bar in Figure 8.

As comparison, we also perform two off-line experiments. First, we train a C4.5 decision tree using the same training data and same attributes as our SVM. The result (prediction accuracy) is shown as the first bar in Figure 8. We can see that our SVM has superior performance over the decision tree. Second, we also use 40 loading events to train our SVM, but all of these 40 events are from 3 volunteers. Then we use this trained SVM to detect the loading events from other 6 volunteers. The accuracy is shown as the third bar in Figure 8, which is only slightly lower than the second bar. This result illustrates that our method is not user-sensitive. It does not need to be trained for every single individual.

In our method, after a getting-off-car event, if a loading event is detected, we assume the user is shopping (probably at a grocery store), and the GPS reading when getting off the car is used for store mapping. Table I lists the most frequent stores our volunteers shop at. The first column is the name of store as well as its price level (retrieved from *Yelp*). The second column lists the number of visits (tagged by volunteers) and the third column is the number of correct detections reported by our method. We can see that, in most cases, our method can precisely identify the store a user is shopping at.

TABLE I
GROCERY SHOPPING DETECTION

Store Name	Visiting Times	Detected Times
Save Mart ($)	12	10
Safeway ($$)	9	8
Walmart ($)	9	9
Trader Joe's ($$)	8	5
Target ($$)	7	7
Whole Foods ($$$)	5	3

Our method also detects the loading time (if a loading event is found) in order to infer the number of bought items. Figure 9 plots the detected loading time (y-axis) vs. the number of items bought (x-axis). Each dot in the figure represents a detected loading event. We did not ask our volunteer to record their real loading time, because it is too troublesome and the timing action may introduce extra noise to sensor readings (of the loading action). Moreover, the rationale behind our method is that we assume the user spending level has positive correlations to stores' price level, and the weight of each store is determined by the number of items bought in this store. So we are more interested in the relationship between the detected loading time and the number of item bought. In Figure 9, we can see that the dots are distributed along the diagonal and exhibits an approximate positive linear relationship between two variables mentioned above.

Finally, we calculate SL for each volunteer using our method (by Equation 6), and then calculate each volunteer's *real spending level* (also use Equation 6, but substitute the number of items for T^i, and count in all the shop visits reported by the volunteers instead of only the detected ones). The average estimation error between SL and *real spending level* is 19%, where the error is defined as $\frac{|SL - real\ spending\ level|}{real\ spending\ level}$. This result demonstrates that the accuracy of our spending level profiling is satisfactory, especially considering that we do not access any banking, billing, email or message information.

We would like to mention that in data set II, there are in total 436 getting-off-car events, and only 82 of them are grocery shopping. For the rest 354, most of them occur in the volunteers' home or working locations which do not trigger the loading detection (see Section III-C). The average time that the loading detection algorithm runs is only 21 minutes per getting-off-car event. So energy-wise, walking and driving detection which runs all day long matters much more than the loading detection algorithm.

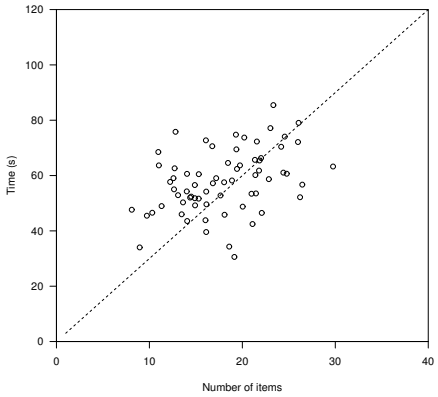

Fig. 9. The result of loading time estimation.

V. Discussion

A. Detection of Home and Working Location

In our shopping profiling, home and working locations are excluded to save energy or improve accuracy (see Section III-C). In fact, the detection of home/working locations is relatively easy. Google Now can perform such detection automatically, in which users' historical traces and WiFi access point information are used. If Google Now is turned off, we can also detect home/working locations based on our getting-off-car detection technique. Each time a getting-off-car event is detected, we turn on the GPS and log its reading. Since users' visits to these two places should be far more often than other places, we can compare the getting-off-car locations and pick them out. We only need to find two most frequent places and exclude them, without differentiating which is home and which is the working location.

B. Feasibility of Covert Privacy-Profiling

In an Android device, an app can run in the background sampling sensor readings. In order to collect sensor data all day long, it needs to run on device startup, which is easy in Android. The Android memory management system sometimes kills background apps to release memory. What the app needs to do to prevent from being killed is showing a message (any content) in the Android notification bar (drop-down area). Therefore, it is relatively easy for a malicious app to pretend as a good one and collect sensor readings continuously in the background.

When a user installs an app, a list of access requests will be prompted for users' approval. If an app requests to access users' identity, calendar, contacts, call log or messages, it may raise users' alert. But our method does not access these sensitive information, as well as the microphone and camera. We only use access motion sensors and location service, which is very normal in the app store.

If a smartphone's battery drains much faster than normal when an app is running in the background, the user may also raise awareness. Therefore, in our method, we minimize the usage of GPS, which is very power-consuming. We also use lightweight algorithms to detect walking and driving, instead of machine learning based approaches, in order to save energy.

In Section IV, we mention that all the sensor data are logged and stored. However, it is for the purpose of performance evaluation. When used for privacy profiling, our method does not need to store all sensor readings. Walking, driving and loading detection are all performed on-the-fly. Only the detection results and timestamps are stored (or sent out wirelessly).

C. Privacy Profiling through Website Access Pattern

It is also possible to profile users' spending level through the analysis of users' website access pattern. However, in order to obtain users' website access information, we need to either sniff smartphone's network traffic or visit web browsing history, both of which are not accessible by third party apps. On the contrary, our sensor-based method, as discussed above, does not require any special access privilege, and is easy to perform but difficult to be detected.

VI. Conclusion

In this paper, we propose to profile users' spending level through smartphone sensors. We do not access smartphones' microphone, camera, users' call log, calendar, contacts or messages; GPS usage is also minimized. Therefore, our method is performed in a non-intrusive manner and is energy-efficient. Our work demonstrates that users' long-term privacy can be invaded through smartphone sensors in a way that users are completely not aware of.

References

[1] J. Chon and H. Cha, "Lifemap: A smartphone-based context provider for location-based services," *IEEE Pervasive Computing*, vol. 10, no. 2, pp. 58–67, 2011.

[2] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine," in *Ambient Assisted Living and Home Care*. Springer, 2012, pp. 216–223.

[3] S. A. Hoseini-Tabatabaei, A. Gluhak, and R. Tafazolli, "A survey on smartphone-based systems for opportunistic user context recognition," *ACM Computing Surveys (CSUR)*, vol. 45, no. 3, p. 27, 2013.

[4] M. Keally, G. Zhou, G. Xing, J. Wu, and A. Pyles, "Pbn: towards practical activity recognition using smartphone-based body sensor networks," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2011, pp. 246–259.

[5] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," in *AAAI*, vol. 5, 2005, pp. 1541–1546.

[6] T. Choudhury, S. Consolvo, B. Harrison, J. Hightower, A. LaMarca, L. LeGrand, A. Rahimi, A. Rea, G. Bordello, B. Hemingway *et al.*, "The mobile sensing platform: An embedded activity recognition system," *Pervasive Computing, IEEE*, vol. 7, no. 2, pp. 32–41, 2008.

[7] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2010.

[8] A. Brajdic and R. Harle, "Walk detection and step counting on unconstrained smartphones," in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. ACM, 2013, pp. 225–234.

[9] R. Libby, "A simple method for reliable footstep detection in embedded sensor platforms," 2008.

[10] L. Rong, D. Zhiguo, Z. Jianzhong, and L. Ming, "Identification of individual walking patterns using gait acceleration," in *2007 1st International Conference on Bioinformatics and Biomedical Engineering*, 2007, pp. 543–546.

[11] P. Barralon, N. Vuillerme, and N. Noury, "Walk detection with a kinematic sensor: Frequency and wavelet comparison," in *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*. IEEE, 2006, pp. 1711–1714.

[12] D. A. Johnson and M. M. Trivedi, "Driving style recognition using a smartphone as a sensor platform," in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*. IEEE, 2011, pp. 1609–1615.

[13] T. Stiefmeier, D. Roggen, G. Ogris, P. Lukowicz, and G. Tröster, "Wearable activity tracking in car manufacturing," *IEEE Pervasive Computing*, vol. 7, no. 2, pp. 42–50, 2008.

[14] A. Rai, Z. Yan, D. Chakraborty, T. K. Wijaya, and K. Aberer, "Mining complex activities in the wild via a single smartphone accelerometer," in *Proceedings of the Sixth International Workshop on Knowledge Discovery from Sensor Data*. ACM, 2012, pp. 43–51.

[15] N. Eagle and A. Pentland, "Reality mining: sensing complex social systems," *Personal and ubiquitous computing*, vol. 10, no. 4, pp. 255–268, 2006.

[16] H. Ying, C. Silex, A. Schnitzer, S. Leonhardt, and M. Schiek, "Automatic step detection in the accelerometer signal," in *4th International Workshop on Wearable and Implantable Body Sensor Networks (BSN 2007)*. Springer, 2007, pp. 80–85.

[17] V. T. Inman, H. J. Ralston, and F. Todd, *Human walking*. Williams & Wilkins, 1981.

[18] S. J. Hoch, B.-D. Kim, A. L. Montgomery, and P. E. Rossi, "Determinants of store-level price elasticity," *Journal of marketing Research*, pp. 17–29, 1995.