UPDetector: Sensing Parking/Unparking Activities Using Smartphones

Shuo Ma
University of Illinois at Chicago
Chicago, IL, USA
sma21@uic.edu

Ouri Wolfson
University of Illinois at Chicago
Pirouette Software Consulting Inc.
Chicago, IL, USA
wolfson@cs.uic.edu

Bo Xu
University of Illinois at Chicago
Pirouette Software Consulting Inc.
Chicago, IL, USA
boxu@cs.uic.edu

ABSTRACT

Real-time information about vacant parking spaces is of paramount value in urban environments. One promising approach to obtaining such information is participatory sensing, i.e. detecting parking/unparking activities using smartphones. This paper introduces and describes multiple indicators, each of which provides an inconclusive clue for a parking or an unparking activity. As a result, the paper proposes a probabilistic fusion method which combines the output from different indicators to make more reliable detections. The proposed fusion method can be applied to inferring other similar high-level human activities that involve multiple indicators which output features asynchronously, and that involve concerns about power consumption. The proposed indicators and the fusion method are implemented as an Android App called UPDetector. Via experiments, we show that our App is both effective and energy-efficient in detecting parking/unparking activities.

Categories and Subject Descriptors
J.m [**Computer Applications**]: Miscellaneous

General Terms
Algorithms

Keywords
Parking, unparking, sensor, fusion, accelerometer, smartphone, activity.

1 Introduction

Vacant parking spaces are scarce resources in many urban areas, and finding these in crowded urban environments can be very frustrating. In addition, cruising for a vacant parking space slows down traffic, causes traffic jams, and pollutes the environment. It is reported that vehicles searching for parking in downtown Los Angeles created 38 trips around the world, producing 730 tons of carbon dioxide and burning 47,000 gallons of gas in one year [16]. The key to enable a service that navigates drivers to available parking spaces is the detection and collection of real-time parking space availability information. In [23] we have shown that this collection, even when restricted to a very small fraction of the drivers (e.g. 5%), and even in the face of detection errors (up to 25% false positives and negatives), can produce very reliable real-time estimates (up to 90%) of parking availability on city blocks. This reliability is enabled by storing the parking activity history of the participating drivers, and combining it with real-time signals (or observations). These signals are the subject of this paper. Some indicators of parking activities are mentioned in [7, 18], but a comprehensive list and fusion methods are not provided.

Real-time parking space availability information is of great value in alleviating the parking problem. By feeding such information to navigation systems, drivers can be directly led to an available parking space, or to a block with a high-likelihood of availability. Existing approaches of generating/collecting real-time parking spaces availability information can be classified into following categories: (i) infrastructure based; (ii) probe vehicle based; and (iii) participatory sensing based.

An infrastructure based approach requires installing sensors under the pavement (e.g. SFPark project in San Francisco and StreetLine sensors [20]). This is expensive to implement and maintain. For example, the SFPark project costs $23M. Furthermore, the sensors tend to malfunction in adverse weather conditions, e.g. when covered by mud or snow.

A probe vehicle based approach [7] uses vehicles equipped with inexpensive sensors such as ultrasonic sensors to scan the street. To provide real-time availability information, a probe vehicle needs to scan the same street repeatedly; and to cover large areas, multiple vehicles need to scan different streets concurrently. Thus, this approach also incurs a high cost and, if dedicated vehicles are used, introduces additional traffic.

A participatory sensing based approach exploits the sensors in smartphones to detect parking/unparking activities. In this paper, we design and implement an energy-efficient mobile App called Unparking/Parking detector (*UPDetector)* that effectively detects parking and unparking activities by analyzing and fusing the data of multiple sensors embedded in smartphones. Specifically, the contributions of this work include:

1. We propose several indicators, each associated with one or more smartphone sensors, for detecting parking/unparking activities. These indicators cover both paid and free parking

SIGSPATIAL '14, November 04-07 2014, Dallas/Fort Worth, TX, USA
Copyright 2014 ACM 978-1-4503-3138-8/14/11 $15.00
http://dx.doi.org/10.1145/2674918.2674929

scenarios. For the purpose of energy conservation, we distinguish between periodical and triggered indicators.

2. We propose a probabilistic method to fuse features output by the different indicators. These indicators are asynchronous, i.e. they output feature-vectors at different times. The proposed fusion method is proved to have a desirable *reinforcement* property. The fusion method can be applied to inferring other high-level human activities that are characterized by multiple asynchronous indicators. For example, detecting if a driver is fueling at a gas station.
3. The proposed detection method works regardless of the phone placement, e.g. pants pocket, or hand-held.
4. Our design of *UPDetector* reduces the usage of GPS to save power. We evaluate its energy-consumption via experiments.

The rest of this paper is organized as follows. In Sec. 2, we introduce indicators and describe how to fuse features output by different indicators. Next we detail the features and the implementation of individual indicators in Sec. 3. In Sec. 4 we describe the implemented *UPDetector* App and its performance. Related work is presented in Sec. 5, and in Sec. 6 we conclude.

2 Indicators and Indicator Fusion

In this section, we first propose a list of indicators in subsection 2.1. Then, in subsection 2.2, we divide the indicators into two types, the periodical and the triggered indicators. Then we propose a fusion method in subsection 2.3.

2.1 Preliminaries on Indicators

An *indicator* is an event that reveals some hint or clue of a parking or an unparking activity. For example, one indicator for unparking is that a person first walks then drives. But this indicator cannot distinguish a passenger from a driver. From this perspective, a stronger indicator for unparking is that the phone is connected with the car via Bluetooth since in general a passenger is less likely than a driver to connect to the car via Bluetooth. Similarly, Bluetooth disconnection from the car is an indicator for parking. Another exemplary indicator for parking is that a person walks towards a roadside pay box and then walks back to a car. Note that this *pay-at-street-parking* indicator is complex enough to be considered an activity by itself and thus can be decomposed into sub-indictors in order to be implemented. In addition, drivers could be encouraged (via incentives) to manually signal a parking or unparking activity. This user input event is also considered as a special kind of indicator. In this paper we focus on indicators that have to be detected by sensors.

Table 1 gives a list of indicators, where the second column states whether the indicator is for parking, unparking or both activities; and the last column lists the sensors that are required to implement the indicator. Indicators output *vectors*. Each vector consists of multiple scalar values, each of which is called a *feature*. Vectors of the same indicator have the same set of features. For example, the features of the acoustic sound indicator include Zero Crossing, Spectral Flux [8]. Sec. 3 details the features of a subset of the indicators listed here.

Table 1 Example indicators of parking/unparking activities

Indicator	activity	Explanation	Sensors
change in the variance of the acceleration (*CIV*)	both	In parking activities, a person first drives and then walks. Since walking often has a large variance in acceleration while driving has a small variance, this transition leads to a sudden increase in the variance of acceleration. Likewise, in unparking activities, the variance of acceleration usually suddenly decreases.	accelerometer
phone connected or disconnected to the car via Bluetooth	both	The phone is connected/disconnected to a car via Bluetooth. (The App asks the user to identify the car Bluetooth device from a list of available Bluetooth devices. This request is only done once.)	Bluetooth
motion-state transition (MST)	both	A parking activity corresponds to a transition from the *driving* state to the *walking* state; and an unparking activity corresponds to a transition from the *walking* state to the *driving* state.	accelerometer
acoustic signals	both	The sounds of human-vehicle interactions that are typically made only during parking or unparking activities. Example interactions include turn on/off the vehicle engine, and open/close the vehicle doors.	microphone
car backing	both	Backing the car is common in parking/unparking activities. It is detected by sensing a sudden reverse in the direction of acceleration.	accelerometer and gyro
pay at street-parking box	parking	In the paid street parking scenarios, a driver often needs to walk to a pay box to buy a parking ticket and walk back to the car to place the ticket in the car.	accelerometer and GPS
WiFi signature [10]	unparking	If the parking location is known, a Wifi signature can be created for it and then used to detect an unparking activity by periodically comparing the signature of the current location to that of the known parking location.	wireless interface
parking payment mobile App's	parking	Parking payment mobile Apps such as ParkMobile [11] and PayByPhone [12] give a hint of a possible parking activity when such an App is brought to the foreground of the smartphone by a user.	
User-input signal	both	Drivers may manually signal a parking/unparking event due to incentives introduced by any gamification or socialization feature of the App.	

2.2 Periodical and Triggered Indicators

In UPDetector, indicators that rely only on energy-efficient sensors (e.g. the accelerometer) output a vector periodically. Such indicators are referred to as *periodical* indicators. For example, both the *Change-In-Variance* indicator and the *motion state transition* indicator are periodical indicators.

Indicators that involve energy-hungry sensors such as the microphone, are not periodically monitored in order to conserve energy. They are triggered to output a vector only when the parking or unparking becomes the *hypothesis*, i.e. indicated by the periodical vectors as the most likely outcome among the three outcomes, namely *parking*, *unparking*, *none*. We refer to such indicators as *triggered* indicators. For example, the engine-start sound is a triggered indicator. That is, only when the periodical vectors indicate unparking as the most likely outcome, the microphone starts to record for a few seconds and outputs a vector of features of the recorded sound sample. Triggered indicators can be considered auxiliary evidences to verify or refute the hypothesis proposed by the periodical indicators.

Table 2 lists the indicators described in Table 1 with their corresponding category, i.e. periodical or triggered. The table shows the output frequency for the periodical indicators; and for the triggered indicators, it shows which hypotheses, i.e. parking, unparking or both, trigger the indicator.

Table 2 List of categorized indicators

Indicator	**Type**
sudden change in the variance of acceleration	periodical: once every few seconds
phone connected or disconnected via Bluetooth	periodical: frequency at which the smartphone monitors the Bluetooth connection
motion state transition	periodical: once every few seconds
acoustic signals	triggered: by parking and unparking hypotheses
car backing	periodical (only when the user at *in_vehicle* state): once every a few seconds
pay at street-parking	triggered: by parking hypothesis
parking payment mobile App's	periodical: frequency at which the smartphone monitors the foreground App
WiFi signature[10]	periodical: compute the WiFi signature at certain frequency and compare it to the signature of the parking location

2.3 Indicator Fusion

2.3.1 Proposed Fusion Method

Whenever some indicator outputs a vector, we need to calculate the probability for each of the three possible outcomes, i.e. *parking*, *unparking*, and *none*, denoted by O_1, O_2, O_3, respectively. Let d be the average duration of a parking/unparking activity (e.g. one minute). Assume that at time point t a vector V_P of periodical indicator I is generated. Let S be the set of vectors consisting of V_P and the latest vector $V_{P'}$ of every periodical indicator other than I (assuming that vector $V_{P'}$ is generated no earlier than time point t-d). The indicators are assumed independent and thus the vectors in S are considered independent. Define a *fusion set* to be a set of independent vectors to be fused. S is a fusion set. Therefore, we can compute the probability $P(O_i|S)$, i=1,2,3 using Eq. (1) below.

$$P(O_i|S) = \prod_{X \in S} P(X|O_i) * P(O_i)/P(S) \quad (1)$$

The calculation of the term $P(X|O_i)$ and $P(O_i)$ are detailed in subsections 2.3.1.2 and 2.3.1.3 respectively. If *none* is the most likely outcome, then no indicator is triggered and thus no parking or unparking activity will be detected. Otherwise, the most likely outcome (i.e. either parking or unparking), denoted by O_h, becomes the *hypothesis*, and invokes the triggered indicators. Each triggered indicator outputs one vector. Denote such triggered vectors by V_{R_1}, V_{R_2},…, V_{R_m}, where V_{R_i} is generated earlier than V_{R_j} for $i < j$. Then the hypothesis O_h is tested in the following way. Let $\mathcal{R}_j$ the set of vectors including the triggered vector V_{R_j}, all the triggered vectors that are generated before V_{R_j}, and the periodical vector set S that proposes the hypothesis, i.e. $\mathcal{R}_j = S \cup \{V_{R_i} | i \leq j, i \in [1, m]\}$. Whenever a triggered vector V_{R_j} is generated, we use Eq. (1) to calculate the probabilities for all three outcomes, where set S is replaced by set $\mathcal{R}_j$. That is, $\mathcal{R}_j$ is also a fusion set. Then we normalize the calculated probabilities, denoted by $P_N(O_i|\mathcal{R}_j)$, using Eq. (2).

$$P_N(O_i|\mathcal{R}_j) = \frac{\prod_{X \in \mathcal{R}_j} P(X|O_i) * P(O_i)}{\sum_{i=1}^{3} (\prod_{X \in \mathcal{R}_j} P(X|O_i) * P(O_i))}, i = 1,2,3 \quad (2)$$

We set a threshold $T \in (0, 1)$, referred to as the *detection threshold*, such that an activity of the hypothesis outcome, i.e. parking or unparking, is considered *detected* only when the normalized probability of the hypothesis is above T, i.e. $P_N(O_h|\mathcal{R}_j) \geq T$. Note that once a parking or an unparking activity is detected, we say that the hypothesis O_h is verified by vector set $\mathcal{R}_j$. Therefore, there is no need to consider any triggered vector that is generated after V_{R_j}, i.e. V_{R_k}, $k = j + 1, j + 2, \ldots m$.

Multiple detections of the same activity: It is possible that one parking/unparking activity is detected multiple times by some indicator (e.g. the *CIV* indicator) because the activity lasts a period in which the indicator outputs multiple times. To cope with this, we consider all detected activities of the same type (i.e. parking or unparking) within a short period (e.g. half a minute) as a single activity.

2.3.1.1 Localization Process

A parking or unparking activity needs to be associated with the time and location of the activity. To save energy, the App only invokes the localization process when it is necessary. The timing for localization could be either when a hypothesis is proposed (by the periodical vectors) or when a hypothesis is confirmed. If the location

is retrieved at the time when a hypothesis is proposed, then the location is cached upon retrieval, and consumed if later the hypothesis is confirmed.

Define the temporal interval from the time when a parking/unparking activity happens to the time when a location is retrieved for the activity as the *delay of localozation*, or simply, the *delay*. Obviously the smaller the delay is, the closer the retrieved location is to the true location where the activity happens. Thus, it is better to invoke the localization process at the time when a hypothesis is proposed instead of confirmed since it leads to a smaller delay.

The App uses the following localization process. Specifically, one location fix L_1 is retrieved via the smartphone localization API, which intelligently fuses the localization sources GPS, WiFi, and cellular networks. Simultaneously, another location L_2 is requested via *Skyhook* [17], a third-party location provider which uses known Wi-Fi hotspots. The retrieval of L_1 and L_2 takes about two seconds on average based on our experiments, and each one of them is returned with an accuracy estimate. We then choose the location with the higher accuracy as the detected location of the hypothesis. Note that the use of a third-party localization API (e.g. Skyhook) consumes little additional energy since it is invoked occasionally, i.e. only when a parking or unparking hypothesis is proposed. Meanwhile it helps improve the location precision.

If one parking/unparking activity is detected multiple times we use the time and location of the first detection.

2.3.1.2 Calculation of $P(X|O_i)$

In this subsection, we detail how to calculate $P(X|O_i)$, i.e. the probability that vector X occurs given outcome O_i. Let $X = (x_1, x_2, \dots, x_n)$ where each x_i is a feature. We regulate that all features in X are mutually independent. In the case that some features are dependent on each other (two features are dependent if their Pearson's Correlation is over a threshold), only one of them is included in X. Since all the features in X are mutually independent, the term $P(X|O_i)$ can be computed by Eq. (3), where $P(x_k|O_i)$ is the probability that feature x_k has the current value given that the outcome is O_i.

$$P(X|O_i) = \Pi_{k=1}^{n} P(x_k|O_i) \quad (3)$$

The term $P(x_k|O_i)$ is estimated using the following approach. Conduct experiments that generate outcome O_i. From these experiments collect a sample set of x_k's under the O_i outcome. Normalize all collected x_k's into the [0,1] interval. Discretize the range [0, 1] into several bins, allocate the collected samples into the corresponding bins based on the value of x_k, and calculate the frequency of each bin. If the frequencies of the bins approximate a normal distribution, we estimate the $P(x_k|O_i)$ using the normal distribution of which the mean and standard deviation are estimated using the collected x_k samples. Otherwise, $P(x_k|O_i)$ is estimated to be the frequency of the bin in which x_k falls.

Note that some $P(x_k|O_i)$'s are estimated rather than obtained by experiments. For example, for the Bluetooth indicator, we estimate how many drivers have smartphones connected to the car via Bluetooth, instead of conducting experiments to determine it.

2.3.1.3 Estimating $P(O_i)$'s

Prior probabilities $P(O_i)$'s are estimated using the following approach. $P(O_1)$ of a specific user U is equal to (the amount of time spent on parking activities per *statistical window* / the amount of time per *statistical window*). The size, i.e. amount of time, of a *statistical window* is dependent on the location and time of the day. For example, if user U enters a parking structure (detected by using the energy-efficient Wi-Fi signature method [10]), it means that user U is likely to park soon and thus the statistical window may be just a few minutes. For another example, the size of the statistical window is larger during the night than during daytime since generally parking activities are less likely to occur at night than in daytime. In case more than one rule determines the current window size, the smallest window will apply. For example, assuming that the size of the window is eight hours at night and is ten minutes if the car just enters the garage, then the window is ten minutes if a car enters a garage at night. To estimate the amount of time spent on the parking activities during a statistical window for user U, the App needs to count the average number of parking activities and estimate the average time duration that each parking activity takes. At the beginning, when the user just starts to use our App and there are not enough samples to calculate these, default values will be applied.

Similarly, $P(O_2)$ is equal to (the amount of time spent on unparking activities per *statistical window* / the amount of time per *statistical window*). Finally, $P(O_3) = 1 - P(O_1) - P(O_2)$.

2.3.2 Reinforcement Property

Our proposed fusion method has a desirable *reinforcement* property. That is, by combining vectors which occur under the hypothesis (i.e. either a parking/unparking outcome), we obtain a higher confidence in the hypothesis outcome. We prove that the reinforcement property holds when applied to two indicators. We formalize the property as follows. (The formulation uses the parking outcome as the hypothesis, but the same argument holds for the unparking hypothesis.)

Theorem: Given two fusion-set vectors X_1, X_2 such that $P(X_i|O_1) > Max\big(P(X_i|O_2), P(X_i|O_3)\big), i = 1,2$,i.e. both most likely occur under the parking outcome O_1, $P_N(O_1|X_1, X_2) > Max(P_N(O_1|X_1), P_N(O_1|X_2))$.

Intuitively, the theorem indicates that if the probability for O_1 is highest for each of two vectors X_1 and X_2, then the probability of O_1 is higher for the fusion set $\{X_1, X_2\}$ than the probability of O_1 under either one of the vectors X_1 or X_2. This means that in our fusion method these two vectors reinforce each other in concluding O_1. We give the formal proof below.

Proof: Since X_1, X_2 are interchangeable, next we only prove $P_N(O_1|X_1, X_2) > P_N(O_1|X_1)$. While $P_N(O_1|X_1, X_2) > P_N(O_1|X_1)$

can be proved using the same rationale. Based on the definition of normalized probability, i.e. Eq. (2), we have

$$\begin{cases} P_N(O_1|X_1,X_2) = \frac{P(O_1|X_1,X_2)}{P(O_1|X_1,X_2)+P(O_2|X_1,X_2)+P(O_3|X_1,X_2)} \\ P_N(O_1|X_1) = \frac{P(O_1|X_1)}{P(O_1|X_1)+P(O_2|X_1)+P(O_3|X_1)} \end{cases} \quad (4)$$

Thus, substituting Eq. (4) into $P_N(O_1|X_1,X_2) > P_N(O_1|X_1)$, we obtain Eq. (5).

$$\frac{P(O_1|X_1,X_2)}{P(O_1|X_1,X_2)+P(O_2|X_1,X_2)+P(O_3|X_1,X_2)} > \frac{P(O_1|X_1)}{P(O_1|X_1)+P(O_2|X_1)+P(O_3|X_1)} \quad (5)$$

Taking the reciprocals on the both sides of Eq. (5), we get Eq. (6)

$$\frac{P(O_1|X_1,X_2)}{P(O_1|X_1,X_2)} + \frac{P(O_2|X_1,X_2)}{P(O_1|X_1,X_2)} + \frac{P(O_3|X_1,X_2)}{P(O_1|X_1,X_2)} < \frac{P(O_1|X_1)}{P(O_1|X_1)} + \frac{P(O_2|X_1)}{P(O_1|X_1)} + \frac{P(O_3|X_1)}{P(O_1|X_1)} \quad (6)$$

Subtracting one from both sides of Eq (6), we get Eq (7)

$$\frac{P(O_2|X_1,X_2)}{P(O_1|X_1,X_2)} + \frac{P(O_3|X_1,X_2)}{P(O_1|X_1,X_2)} < \frac{P(O_2|X_1)}{P(O_1|X_1)} + \frac{P(O_3|X_1)}{P(O_1|X_1)} \quad (7)$$

A sufficient condition for Eq. (7) is Eq. (8). That is, if Eq. (8) holds then Eq. (7) stands, and thus the theorem is proved.

$$\begin{cases} \frac{P(O_2|X_1,X_2)}{P(O_1|X_1,X_2)} < \frac{P(O_2|X_1)}{P(O_1|X_1)} \\ \frac{P(O_3|X_1,X_2)}{P(O_1|X_1,X_2)} < \frac{P(O_3|X_1)}{P(O_1|X_1)} \end{cases} \quad (8)$$

Next we prove the first equation in Eq. (8), i.e. $\frac{P(O_2|X_1,X_2)}{P(O_1|X_1,X_2)} < \frac{P(O_2|X_1)}{P(O_1|X_1)}$. The other part $\frac{P(O_3|X_1,X_2)}{P(O_1|X_1,X_2)} < \frac{P(O_3|X_1)}{P(O_1|X_1)}$ can be proved using the same rationale. Denote $\frac{P(O_2|X_1,X_2)}{P(O_1|X_1,X_2)} < \frac{P(O_2|X_1)}{P(O_1|X_1)}$ by Eq (9).

$$\frac{P(O_2|X_1,X_2)}{P(O_1|X_1,X_2)} < \frac{P(O_2|X_1)}{P(O_1|X_1)} \quad (9).$$

Using the Bayes rule on the right side of Eq. (9), we have Eq. (10)

$$\frac{P(O_2|X_1,X_2)}{P(O_1|X_1,X_2)} < \frac{P(X_1|O_2)}{P(X_1|O_1)} \times \frac{P(O_2)}{P(O_1)} \quad (10)$$

Using the Bayes rule on the left side of Eq. (10), we have Eq. (11)

$$\frac{P(X_1|O_2,X_2)}{P(X_1|O_1,X_2)} \times \frac{P(O_2|X_2)}{P(O_1|X_2)} < \frac{P(X_1|O_2)}{P(X_1|O_1)} \times \frac{P(O_2)}{P(O_1)} \quad (11)$$

Using the Bayes rule on the second term of the left side of Eq (11), we have Eq. (12)

$$\frac{P(X_1|O_2,X_2)}{P(X_1|O_1,X_2)} \times \frac{P(X_2|O_2)P(O_2)}{P(X_2|O_1)P(O_1)} < \frac{P(X_1|O_2)}{P(X_1|O_1)} \times \frac{P(O_2)}{P(O_1)} \quad (12)$$

Dividing the common factor $\frac{P(O_2)}{P(O_1)}$ from the both sides of *Eq.* (12), we have *Eq.* (13)

$$\frac{P(X_1|O_2,X_2)}{P(X_1|O_1,X_2)} \times \frac{P(X_2|O_2)}{P(X_2|O_1)} < \frac{P(X_1|O_2)}{P(X_1|O_1)} \quad (13)$$

Since indicators are independent, X_1 and X_2 are independent, Eq. (13) is simplified to Eq. (14).

$$\frac{P(X_1|O_2)}{P(X_1|O_1)} \times \frac{P(X_2|O_2)}{P(X_2|O_1)} < \frac{P(X_1|O_2)}{P(X_1|O_1)} \quad (14)$$

Since X_2 most likely occurs under O_1, i.e. $\frac{P(X_2|O_2)}{P(X_2|O_1)} < 1$. Then it is clear that Eq. (14) stands and thus Eq. (8) stands. Therefore, we have $P_N(O_1|X_1,X_2) > Max(P_N(O_1|X_1), P_N(O_1|X_2))$ ▪

Given the above theorem, it is easy to prove the general case, i.e. that $P_N(O_i|X_1,X_2,\dots X_m) > Max(P_N(O_1|X_1),\dots P_N(O_1|X_m))$ given that the X_j's are independent, and that for each *j*, $P(X_j|O_i)$ is the largest among $P(X_j|O_k)$, *k*=1,2,3.

3 Implementation of Individual Indicators

In this section, we detail the features and implementation of only a subset of the proposed indicators. Other indicators, once implemented, can be plugged-and-played into our system through the fusion method described earlier.

3.1 The Change-In-Variance (CIV) Indicator

3.1.1 Preliminaries on Accelerometer

The accelerometer of a smart phone has a coordinate-system consisting of three axes, as shown in Fig. 1. The *X* axis is horizontal and points to the right, the *Y* axis is vertical and points up and the *Z* axis points towards the outside of the front face of the screen. Each reading of the accelerometer contains three values, that is, one value for each axis. The three axes are defined relative to the screen of the phone in its default orientation.

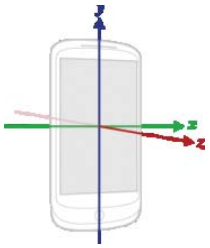

Fig. 1 Axes of mobile phone

3.1.2 Features of the CIV Indicator

A sliding window with a fixed size W seconds is used. The sliding window moves forward N seconds every time it slides. That is, two consecutive windows overlap W-N seconds. We refer to N as the sliding step thereafter. During each window, we calculate the difference between the variances of the acceleration in the second

half and in the first half of the window. Hereafter we refer to this feature as the *VariDiff* of a window. Intuitively, as shown in Fig. 2, an unparking activity results in a window where the first half (corresponding to the *walking* state) has a large variance while the second half (corresponding to the *driving* state) has a small variance. Likewise, the parking activity has a similar sharp difference in variances of the two halves of the window. We have observed via experiments that this variance discrepancy exists no matter where the phone is placed, e.g. in pants pocket, in handbag, etc.

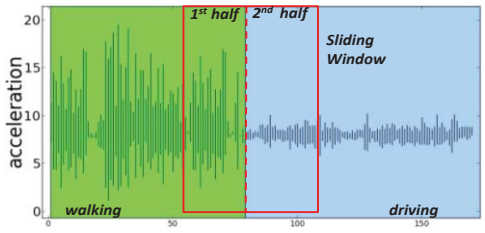

Fig. 2 The sliding window for the CIV indicator

However, the *VariDiff* feature of the current window itself is not sufficient due to noise. For example, the *VariDiff* feature of a window during which the user walks, may be either positive or negative since the acceleration during walking is oscillating. As a result, such a window may be misidentified as a parking or unparking activity. To deal with the noise, we consider using the *VariDiff* feature of all windows within a *scope*, denoted by S. Then each *CIV* vector consists of three features: (i) the *VariDiff* feature of the current window; (ii) the average value of the *VariDiff* feature during the preceding $S/2$ windows; (iii) the average value of the *VariDiff* feature during the succeeding $S/2$ windows. Formally, the *scope* S is the total number of preceding and succeeding windows that are considered in a *CIV* vector. Observe that S does not include the current window. In order to calculate feature (iii), the production of the vector of a window is delayed for $S/2$ windows.

Fig. 3 illustrates the calculation of the *CIV* vectors. Assume that the *VariDiff* feature of 1st, 2nd, 3rd, 4th, 5th window has a value of 0.03, 0.01, 2.6, 1.6, 1.8, respectively, and the scope S equals to 4. Then when the 5th window ends, the *CIV* vector of the 3rd window is computed and equals to (0.02, 2.6, 1.7), where 0.02 is the mean of the *VariDiff* of the 1st and 2nd windows and 1.7 is the mean of the *VariDiff* of the 4th and 5th windows.

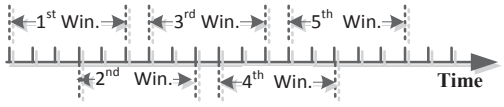

Fig. 3 An example of calculating CIV vectors

3.2 The Bluetooth Indicator

The Bluetooth embedded in the phone is a strong indicator when it is enabled. When a phone is connected to a car (requiring one time human input to indicate the name of the Bluetooth device of the car), there is a good chance that the person is the driver and intends to depart, which corresponds to an unparking activity. Similarly when a phone is disconnected, it is likely due to a person leaving the car, which indicates a parking activity. There is only one feature for the Bluetooth indicator, which takes one of the following three values: i.e. *connected, disconnected, not enabled*.

3.3 The Motion State Transition Indicator

Motion states, such as *waling, driving*, etc., can be classified from raw accelerometer readings. After motion states are classified, the transitions between motion states are identified to signify the parking/unparking activities. The vectors of the motion state transition indicator thus include four features. The first two features are the probabilities of the latest motion state being *walking* and *driving*, respectively. Similarly, the last two features are the probabilities of the motion state before latest being *walking* and *driving,* respectively.

3.4 The Acoustic Indicators

Acoustic indicators refer to the sounds of human-vehicle interactions that are typically made only during parking or unparking activities. Example interactions include turn on/off the vehicle engine, open and close the vehicle doors. Such sounds often have distinct frequency and amplitude from each other.

It has been shown in [13] that the sounds of these particular human-vehicle interactions can be classified with relative high precision and recall using specific audio features [8] e.g. pitch, spectrum. However, in [13] the authors do not consider power consumption and thus the sounds are recorded constantly. In our case, in order to save energy, acoustic sounds are modeled as triggered indicators, as described in 2.2.2. That is, they are only activated and output vectors after the parking or unparking outcome becomes the hypothesis.

In addition to the work of [13], we included the "bus noise" (i.e. the bus engine sound with the background noise) as one of the acoustic sounds, and found out that the "bus noise" sound is highly distinguishable from other sounds such as engine start, or door open/close. This will help distinguish a private car trip from a bus trip.

4 Evaluation

We implemented a prototype system on the Android platform. This section details the experimental methodology and the results. Specifically, we introduce our experimental setting in subsection 4.1. Then we show the performance of the App in subsection 4.2.

4.1 Experimental Methodology

4.1.1 Mobile App Implementation

UPDetector can be implemented on all mobile platforms, such as Android, Apple iOS and Windows mobile systems. We implement a prototype on the Android platform using the Samsung Galaxy S3, which has a 1 GB RAM and quad-core 1.4 GHz Cortex-A9 processor. The sampling frequency of the accelerometer ranges about 10~30 Hz. Fig. 4 shows some screenshots of the implemented prototype.

We have implemented the following indicators in the *UPDetector* App: the *Bluetooth* indicator, the *CIV* indicator and the *MST* indicator. We have also implemented the acoustic indicator (i.e. the audio analysis part) on the laptop; unfortunately, we cannot port the function to the App since Android system currently does not support the audio feature extraction library. The Bluetooth indicator is highly reliable by itself and may work independently of other indicators. Therefore, here we restrict attention to vehicles that do not have Bluetooth devices and present the detection results using the *CIV* and the *MST* indicators.

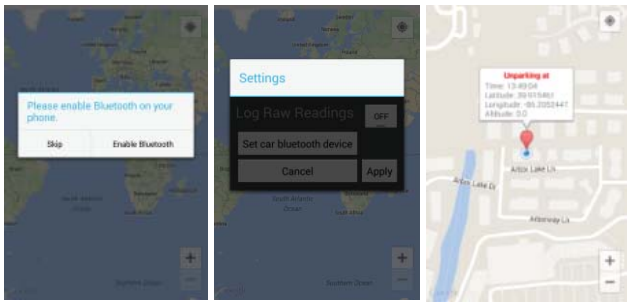

Fig. 4 *UPDetector* implementation screenshots

4.1.2 Data Collection

We have had three people carry a smartphone with the *UPDetector* application running during multiple days. During these periods of time, the smartphone was placed in different positions including front pant leg pocket, coat pocket, handbag/knapsack and held in hand. The parking/unparking activities in the experiment were perpendicular outdoor parking in a living community. The time (in seconds) of each parking/unparking activity was manually recorded as the ground truth. The time of an unparking activity is the second when the vehicle starts to move from a parked state; and the time of a parking activity is the second when the vehicle reaches the still state.

The collected data is split into one training set and one test set. The training data set contains 40 parking activities and 40 unparking activities. The test data set contains 60 parking activities and 60 unparking activities. The training set is used to learn the conditional probability of features under different outcomes, i.e. $P(x_k|O_i)$'s. Via experiments, we observe that $P(x_k|O_i)$'s in the training set approximate normal distributions. So we estimate the parameters of the normal distributions using the training data set and then use them for the test data set. The test data is used to evaluate the performance of the detection of parking/unparking activities.

4.1.3 Detection Methods

We evaluate five detection methods. These methods are categorized into two groups. The first group consists of three methods that use a single indicator: (i) the method that only uses the *Change-in-Variance* (*CIV*) indicator, referred to as the *CIV* method hereafter; (ii) the method that only uses the *Motion State Transition* (*MST*), where the motion state classifier is implemented using the features described in [5] (multiple classification methods, including meta-classification methods such as AdaBoost and RandomForest, are tried to train the model and the best classification method, i.e. RandomForest, is chosen), referred to as the *MST-CL1* method hereafter; (iii) the method that only uses the *Motion State Transition* (*MST*), where the motion state classifier is provided by Google Activity Recognition *(GAR)* API [1], referred to as the *MST-CL2* method hereafter. The *GAR* API returns a distribution over five possible states including *driving*, *walking, still, tilting,* and *unknown*. We have observed that the *GAR* API outputs the *unknown* state as the most likely state frequently. To better utilize the API's results, we modify the most likely state for the following case. If the *GAR* API outputs *unknown* as the most likely state, and if the second most likely state S has a likelihood that is larger than the sum of the likelihoods of other three states except *unknown* and S, we treat S as the most likely state.

The second group consists of two methods that fuse multiple indicators. The first method, referred to as the *CIV-MST_CL1* method, combines the *CIV* indicator and *MST-CL1* using the probabilistic based fusion algorithm described in Sec. 2.3.

The second method, referred to as the *CIV-MST_CL1_CL2* method, combines the *CIV-MST_CL1* method with the *MST_CL2* method. Specifically, each activity A_d detected by the *CIV-MST_CL1* method is considered a hypothesis (thus a location is retrieved and cached when A_d is detected). We have learned from experiments that the *MST_CL2* method is reliable but suffers from a large delay (see Sec. 4.2.1). Therefore, we consider that A_d is confirmed if later *MST_CL2* also detects the same type (i.e. parking/unparking) of activity as A_d. If multiple A_d's of the same activity type have been output by *CIV-MST_CL1* when *MST_CL2* outputs a detection, we use the location retrieved for the first A_d. Note that here we use a simple "and" logic to combine the detection results of *CIV-MST_CL1 and MST_CL2* because *MST_CL2* is highly reliable (but unfortunately has a long delay) and thus a simple "and" logic is sufficient. Otherwise, we would have used the proposed probabilistic fusion algorithm to combine *CIV-MST_CL1* and *MST_CL2*.

4.1.4 Matching Detected Activities with the Ground Truth

A detected parking (unparking) activity A_d is matched to a ground truth parking (unparking) activity A_g if the time difference between A_d and A_g is smaller than five seconds[1]. For a detected activity A_d, the matching ground truth activity A_g can always be uniquely identified because any two consecutive ground truth activities are at least minutes away from each other, and thus there is no confusion in the matching.

[1] except for the *MST_CL2* method; since it suffers a long delay, its value is one minute

Note that a ground truth activity may be detected multiple times (i.e. matched to several detected activities that are temporally consecutive and close to each other). This only happens when a sliding window is used in the indicator (e.g. in the *CIV* indicator) and the window slides in a way such that the two consecutives windows overlap. For example, consider the example shown in Fig. 2. The red window in the figure represents a detected unparking activity. If we slide the red window slightly to the right, apparently, the new window may still represent a detected unparking activity that is matched to the same ground truth activity. When multiple detected activities are matched to the same ground truth activity A_g, we use the first matched detected activity and ignore the rest of the detected activities that are matched to A_g.

4.1.5 Performance Measures

The performance is measured by the *precision* and *recall*. Eq. (15) gives the definition, where *tp*, *fp*, *fn* are the numbers of true positives, false positives, and false negatives, respectively. A detected parking (unparking) activity A_d is a true positive if it matches a ground truth parking (unparking) activity; otherwise it is a false positive; and a ground truth activity A_g is considered a false negative if no detected activity is matched to it.

$$\begin{cases} precision = tp / (tp + fp) \\ recall = tp/ (tp + fn) \end{cases} \quad (15)$$

Another measure is the delay of localization. (see def. in Sec.) Denote by t_g the timestamp of the ground truth activity A_g and by by t_l the time when the location is received. The *delay*, denoted by D, can be calculated using Eq. (16).

$$D = t_l - t_g. \quad (16)$$

4.2 Evaluation Results

4.2.1 Detection Accuracy and the Delay

The *MST_CL1* classifier is implemented according to [5]. The *MST_CL2* uses the activity recognition API provided by Google. The API provides one parameter that adjusts the update frequency. This parameter is set to zero [1] so that the updates are obtained at the highest possible frequency.

Table 3 lists the values for parameters for the *CIV* method and the detection threshold. In this paragraph we discuss these parameters. Many previous works (e.g. [5, 14]) suggest that the window size should be large enough to include a few hundred samples but not too large to increase the delay. Based on the accelerometer sampling frequency in Android (i.e. about 10~30 Hz), 10 seconds is a reasonable window size (other window sizes are also tried and 10 seconds show the best results).

A small sliding step helps capture the sudden change in the acceleration. Intuitively, a small scope helps decrease the delay. We conducted experiments to learn the impact of the *scope* parameter on the precision and recall. The experiments suggest that as the scope S increases, the precision and recall first increases then decreases. This is because the scope only helps when it includes recent past samples; and it starts to hurt the performance as it continues to increase and includes samples from a more remote past. The results suggest that a scope of six windows achieves the best precision and recall.

Table 3 Default values of parameters

Notation	Meaning	Value
W	window size of the *CIV* indicator	10 seconds
N	sliding step of the *CIV* indicator	3 seconds
S	scope of the *CIV* indicator	6 windows
T	detection threshold	0.7

Table 4 shows the performance of the detection methods described in 4.1.3. Note that the average delay in the table refers to the average delay of the true positives, i.e. the detected activities that are matched to the ground truth activities.

In the first group (i.e. methods that use only one indicator), *MST_CL2* gives the best precision and recall but it suffers from a large delay, especially for unparking activities. For this reason the *MST_CL2* method cannot be used alone (i.e. if used a alone the location of the parking/unparking activity cannot be accurately identified). Note that the delay for unparking activities is much larger than that for parking activities for *MST_CL2*. This is due to the fact that the *GAR* API outputs *driving* state with a much larger delay than the *walking* state. In comparison, the *CIV* method have a much smaller delay but with a slightly lower recall and a much lower precision. The *MST_CL1* method has the poorest precision and recall among the three methods. Note that the *MST_CL1* uses the features described in [5], where the authors report a much higher precision and recall for human activities classification. But in [5], the phone has a fixed position (i.e. the front leg pocket) while here the phone is placed in various positions. In addition, [5] does not include *driving* as an activity. In general, *driving* is much harder to be correctly classified than on foot activities such as *walking* or *jogging* since *driving* is easily confused with *still* or *standing*. (This may also explain why the *GAR* API outputs *driving* activity with a much larger delay than *walking* activity.) The results of the first group demonstrate that no individual indicator is good enough.

Table 4 Detection performance on the testing data set

Detection Methods		Parking Activities			Unparking Activities		
		Recall	Precision	Avg. Delay (secs)	Recall	Precision	Avg. Delay (secs)
Methods that use only one indicator	*CIV*	86.2%	29.7%	10.68	87.9%	45.1%	14.43
	MST_CL1	60.3%	18.6%	20	70.6%	22.2%	14.17
	MST_CL2	94.8%	88.7%	17.75	89.6%	89.6%	46.18
Methods that fuse multiple indicators	*CIV-MST_CL1*	91.3%	23.8%	10.3	96.5%	24.3%	15.72
	CIV-MST_CL1_CL2	93.1%	90.4%	9.98	81.8%	93.1%	14.36

In the fusion method group, the *CIV-MST_CL1* has a higher recall than that of both the *CIV* and *MST_CL1* method. However, the method's precision remains unsatisfying. This is because the fusion process enhances the detection confidence when both the *CIV* and the *MST_CL1* method correctly detects the same type activity (i.e. parking/unparking) with a low confidence and thus helps improve the recall. However, when both the *CIV* and the *MST_CL1* method mistakenly detect the same type activity with a low confidence, the fusion also boosts the confidence, and as a result the precision of the *CIV-MST_CL1* method may be lower than the largest precision of the constituting methods.

As the integration of the *CIV-MST_CL1* method and the *MST_CL2* method, the *CIV-MST_CL1_CL2* method inherits all the merits: it has a higher precision than both its constituting methods; it has a fairly high recall while keeps a small delay.

4.2.2 Energy Consumption

We employ PowerTutor [24] to measure the power consumption. For the purpose of localization, GPS is enabled when *UPDetector* is running. But it is in the stand-by mode and consumes little energy (about 0.8 mw) during most of time. GPS only enters the energy-hungry searching mode (about 220 mw) once for each parking/unparking activity. Since there are at most a few parking/unparking activities during a day, the power consumption for localization is negligible.

Most power consumption of the App attributes to CPU usage caused by the computation during the fusion of periodical vectors. When *UPDetector (running the CIV-MST_CL1_CL2 method)* is the only App running and phone activities (such as call, sms) are avoided, the corresponding battery life is about 20.3 hours. The battery life with no app running and no phone activities is around 25 hours. That is, *UPDetector* costs 4.7 hours of the battery life. It is possible to further reduce this cost by decreasing the output frequency of the periodical indicators such as the *CIV, and* by special hardware [4].

5 Related Work

5.1 Parking Spaces Detection

In the past, on street parking slot detection is usually performed by sensors embedded in the pavement, e.g. the SFPark project, or in vehicles [7]. However, these efforts require a significant investment and are too expensive to fully cover a large city. Given the proliferation of mobile devices, smartphone applications such as ParkMobile and PaybyPhone, that allow drivers to pay for parking by mobile phones, are emerging. Such App's can be used by our method as an indicator for parking. [10] proposes a novel method which leverages WiFi beacons in urban environments to detect unparking. This method can be integrated into our work as an indicator for unparking activities. This method by itself is not always applicable since WiFi signature works only when the parking location is covered by multiple WiFi access points.

5.2 Activity Recognition

There have been works on detecting motion activities based on readings from sensors in smartphones. Generally, a motion activity detection algorithm is a classifier which reads raw sensor data (e.g. from GPS [2, 19, 25], from accelerometer [3, 22], from both [15], or from WiFi/GSM [9]), processes it to extract features, and then classifies and outputs the motion activity such as *still, walking, running*. These existing works are not motivated by detection of parking/unparking activities and thus bear the following problems when applied to our problem directly: (i) some of them use GPS heavily and thus are not applicable to our energy-constrained scenario; (ii) many works that use accelerometer to classify everyday human activities do not include *driving* as an activity; and they often assume that the accelerometer is placed on a fixed position with the user. Furthermore, as our study demonstrates, the motion state transition method by itself is not a reliable indicator for parking/unparking.

5.3 Classifier Fusion

In [21], the authors survey existing methods of combing multiple classifiers. These methods include i) ensemble methods that combine multiple homogenous classifiers (i.e. classifiers that are learned using the same set of features and the same classification algorithm), such as Bagging and Boosting; and ii) non-ensemble methods that combine heterogeneous classifiers, such as the majority voting (e.g. voting based on either the number of each class or the aggregated confidence in each class). We apply the ensemble methods, i.e. the Boosting method via Weka, to implement individual indicators such as *MST*. But the ensemble methods do not handle the asynchronous data problem. That is, in our application scenario, different indicators output vectors of different feature sets at different frequencies. Additionally, we aim to save energy, a consideration that is missing in prior work on classifier fusion. Cost-sensitive boosting [6] methods may be applicable, but it is not clear how to incorporate energy consumption into cost functions. As pointed out by the authors of [21], none of the non-ensemble methods are shown to be superior to others, neither theoretically nor empirically. Our proposed fusion method can be considered a non-ensemble method that is motivated by and designed for the unparking/parking detection application, and potentially applied to other applications with the asynchronous data problem.

6 Conclusion and Future Work

We presented the design and implementation of a parking/unparking activities detection system called *UPDetector*. We described several indicators used by *UPDetector,* and their corresponding features. *UPDetector* uses a probabilistic fusion method which combines features output by multiple indicators to derive parking/unparking activity detection results. We described this method in the paper. We evaluated the *UPDetector* prototype via experiments, and demonstrated its effectiveness and energy consumption.

Using the implemented *Bluetooth* indicator, the current App we implemented has a certain capability of distinguishing a driver from a passenger. This capability can be further enhanced via

incorporating other indicators, e.g. the pay-at-street-parking box indicator and the parking-payment-mobile-App indicator listed in Table 2. In addition, acoustic indicators can be used to distinguish buses from private cars.

Currently the probabilities $P(x_k|O_i)$'s are estimated using limited experimental data collected by ourselves. In the future, via the means of gamification or socialization, crowdsourcing may help gather more data and thus provide a more precise estimation of those probabilities.

Acknowledgments: This research was supported in part by the US Department of Transportation National University Rail Center (NURAIL); Illinois Department of Transportation (METSI); and National Science Foundation grants IIS-1213013, CCF-1216096, DGE-0549489, IIP-1315169.

7 References

[1] ActivityRecognitionClient | Android Developers: http://developer.android.com/reference/com/google/android/gms/location/ActivityRecognitionClient.html.

[2] Chu, D., Lane, N.D., Lai, T.T.-T., Pang, C., Meng, X., Guo, Q., Li, F. and Zhao, F. 2011. Balancing energy, latency and accuracy for mobile sensor data classification. Proceedings of the *9th ACM Conference on Embedded Networked Sensor Systems - SenSys* '11. (2011), 54.

[3] Dernbach, S., Das, B., Krishnan, N.C., Thomas, B.L. and Cook, D.J. 2012. Simple and Complex Activity Recognition through Smart Phones. 2012 *Eighth International Conference on Intelligent Environments*. (Jun. 2012), 214–221.

[4] Haichen Shen, Aruna Balasubramanian, Eric Yuan, Anthony LaMarca, D.W. Improving Power Efficiency Using Sensor Hubs Without Re-Coding Mobile Apps.

[5] Kwapisz, J., Weiss, G. and Moore, S. 2011. Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*. 12, 2 (2011), 74–82.

[6] Masnadi-Shirazi, H. and Vasconcelos, N. 2011. Cost-sensitive boosting. *IEEE transactions on pattern analysis and machine intelligence*. 33, 2 (Feb. 2011), 294–309.

[7] Mathur, S. and Jin, T. 2010. Parknet: drive-by sensing of road-side parking statistics. Proceedings of the *8th international conference on Mobile systems, applications, and services*. (2010).

[8] McEnnis, D., McKay, C., Fujinaga, I. and Depalle, P. 2006. jAudio: Additions and Improvements. ISMIR. (2006).

[9] Mun, M., Estrin, D., Burke, J. and Hansen, M. 2008. Parsimonious mobility classification using GSM and WiFi traces. Proceedings of the *5th Workshop on Embedded Networked Sensors*. (2008), 1–5.

[10] Nawaz, S., Efstratiou, C. and Mascolo, C. 2013. ParkSense: A Smartphone Based Sensing System For On-Street Parking. In Proceedings of the *19th ACM International Conference on Mobile Computing and Networking* (MOBICOM 2013). (2013).

[11] Parkmobile: http://us.parkmobile.com/.

[12] PayByPhone.: http://www.paybyphone.com/how-it-works/.

[13] Rababaah, A. 2011. Event Detection, Classification And Fusion For Non-Stationary Vehicular Acoustic Signals. *International Journal of Science of Informatics*. 1, 1 (2011), 9–20.

[14] Ravi, N., Dandekar, N., Mysore, P. and Littman, M. 2005. Activity recognition from accelerometer data. *AAAI*. (2005).

[15] Reddy, S., Mun, M., Burke, J. and Estrin, D. 2010. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks (TOSN)*. 6, 2 (Feb. 2010), 1–27.

[16] Shoup, D. 2005. The High Cost of Free Parking. American Planning Association.

[17] Skyhook Inc.: http://www.skyhookwireless.com/.

[18] Stenneth, L., Wolfson, O., Xu, B. and Yu, P.S. 2012. PhonePark: Street Parking Using Mobile Phones. 2012 *IEEE 13th International Conference on Mobile Data Management*. (Jul. 2012), 278–279.

[19] Stenneth, L., Wolfson, O., Yu, P.S., and Xu, B. . 2011. Transportation Mode Detection using Mobile Phones and GIS Information. Proceedings of the *19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. (2011).

[20] Streetline, Inc.: www.streetline.com.

[21] Tulyakov, S. and Jaeger, S. 2008. Review of classifier combination methods. *Machine Learning in Document Analysis and Recognition*. Figure 1 (2008), 1–26.

[22] Wang, Y., Lin, J. and Annavaram, M. 2009. A framework of energy efficient mobile sensing for automatic user state recognition. Proceedings of the *7th international conference on Mobile systems, applications, and services*. (2009).

[23] Xu, B., Wolfson, O., Yang, J., Stenneth, L, Yu, P.S., and Nelson, P. Real-time Street Parking Availability Estimation. MDM 13: Proceedings of the *14th International Conference on Mobile Data Management*.

[24] Zhang, L., Tiwana, B., Qian, Z. and Wang, Z. 2010. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. Proceedings of *the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*. (2010).

[25] Zheng, Y., Liu, L., Wang, L. and Xie, X. 2008. Learning transportation mode from raw gps data for geographic applications on the web. Proceedings of the *17th international conference on World Wide Web*. 49 (2008).