BluePark: Tracking Parking and Un-parking Events in Indoor Garages

Sonia Soubam
IIIT Delhi
sonias@iiitd.ac.in

Dipyaman Banerjee
IBM Research Lab, India
dipyaban@in.ibm.com

Vinayak Naik
IIIT Delhi
naik@iiitd.ac.in

Dipanjan Chakraborty
Mosix Technologies
dipanjan.chakraborty@gmail.com

ABSTRACT

Finding a parking spot in a busy indoor parking lot is a daunting task. Retracing a parked vehicle can be equally frustrating. We present BluePark, a collaborative sensing mechanism using smartphone sensors to solve these problems in real-time, without any input from user. We propose a novel technique of combining accelerometer and WiFi data to detect and localize parking and un-parking events in indoor parking lot. We validate our approach at the basement parking of a popular shopping mall. The proposed method outperforms Google Activity Recognition API by 20% in detecting drive state in indoor parking lot. Our experiments show 100% precision and recall for parking and un-parking detection events at low accelerometer sampling rate of 15Hz, irrespective of phone's position. It has a low detection latency of 20s with probability of 0.9 and good location accuracy of 10m.

CCS Concepts

•**Networks** → *Sensor networks;* •**Computing methodologies** → Cluster analysis;

Keywords

Smartphone sensing; Parking

1. INTRODUCTION

Shopping malls in India are one of the largest shopping and entertainment zones with about 50 Million footfalls per month [1]. Several parking facilities in these malls are underground and have heavy traffic. It is difficult to detect free parking spots as driver have limited visibility range and parking lots are quite large. Advanced parking systems have limited adoption in India due to several practical issues such as cost, maintenance, mischievous users tampering with sensors, etc. Furthermore, for retrieving parked cars, the only aid available is marked pillars. Users often forget to note down manually the pillar numbers leading to a frustrated experience. We propose *BluePark*, a smart parking system for indoor parking facility, that can detect when and where a car is parked and un-parked using smartphone sensing in real-time with "zero" cognitive load on the user.

Any parking or un-parking event, almost all the time, is associated with a change of locomotive state of the user. Parking events imply a transition from a $drive$ state to a $walk$ state of the user, whereas un-parking events imply the opposite transition [7]. We use the same definitions in this paper. Unlike outdoor environment, detecting $drive$ state in indoor parking facility is challenging due to low vehicle speed and unavailability of GPS. We propose a novel approach of combining accelerometer and WiFi data to accurately detect $walk$ and $drive$ states in indoor parking environment. We further design a light-weight algorithm that quickly detects the state transitions, which translates to parking and un-parking event.

Localization is done with the help of an external server, which is a one time request and it is not power hungry. BluePark was deployed in the indoor parking area of a popular shopping mall in New Delhi. It utilized the existing WiFi APs deployed in the facility.

The primary contributions of this paper are as follows:

1. An algorithm, for detecting drive and walk states in indoor parking areas, which outperforms Google Activity Recognition API's accuracy by more than 20%. Our approach is independent of how the phone is held. We achieve this accuracy by way of proposing
   (a) A rule-based fusion of WiFi and accelerometer data
   (b) Adaptive DBSCAN, a modification of traditional DBSCAN algorithm, that allows unsupervised real-time detection of states using accelerometer data
   (c) A WiFi-based metric that differentiates between moving and not moving in indoor environments
2. Based on our state-detection algorithm, we propose a mechanism for detecting and localizing parking and un-parking events that has 100% precision and recall at sampling frequency of 15Hz or higher, detection latency of 20s with probability of 0.9 and parking location accuracy of 10m.

2. RELATED WORK

Smartphone based parking solutions exists in the literature [6, 8, 7]. However, they detect only parking event [6], or assume the location of parked car to be known from manual parking payment

ICDCN '16, January 04-07, 2016, Singapore, Singapore
© 2016 ACM. ISBN 978-1-4503-4032-8/16/01…$15.00
DOI: **http://dx.doi.org/10.1145/2833312.2833458**

application [8]. PocketParker [7] used transitions between user states to detect parking and un-parking for outdoor parking. The main focus of their work is to create a prediction model, based on parking and un-parking events of user and how drivers not using PocketParker affects the parking estimation. The focus of our work is on accuracy and latency of the event detection.

Accelerometer is a popular choice for transportation mode detection [5]. The existing accelerometer based classifiers are supervised classifiers, which require extensive training to account for dependence of accelerometer data on user and phone position. Moreover, detecting drive state using accelerometer in indoor is difficult due to low driving speed. Transportation mode detection using GPS information have been proposed. Our design scenario is an indoor parking system and therefore, GPS information cannot be used. We present a simple approach of combining WiFi and accelerometer information to detect drive, walk and still states of the user.

3. ARCHITECTURE OF THE SYSTEM

An overview of BluePark's system architecture is shown in Figure 1. It has two components: a mobile client and a cloud service.

The mobile client, an Android application running on the smartphone, is responsible for the state detection and the event detection. WiFi signals received by users' smartphones at the time of event detection is sent to the cloud service to identify location of the parked or un-parked vehicle.

The cloud service provides event localization and parking occupancy status. WiFi fingerprint information of the various parking spots are stored in the service's database. Given a WiFi signature, the cloud service returns the most probable parking spot. The cloud service maintains occupancy status of each parking spot in a *Parking Entries* log using the parking and un-parking events reported by the mobile client.

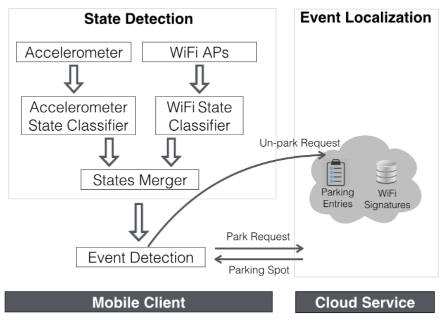

Figure 1: System Architecture of BluePark

4. APPROACH

In this section, we describe the state detection, event detection, and event localization modules in detail.

4.1 State Detection

The state detection module has three parts: Accelerometer State Classifier, WiFi State Classifier, and State Merger.

4.1.1 *Accelerometer State Classifier*

We developed an online unsupervised learning approach to identify $walk$ states of a user from accelerometer data. We propose *Adaptive DBSCAN*, a modified version of DBSCAN [4], which can cluster accelerometer data in real time. The Accelerometer State Classifier uses a threshold value on each cluster's centroid α to determine if the accelerometer traces in that cluster correspond to a $walk$ or $\neg walk$ state.

Algorithm 1 AdaptiveDBScan

```
INPUT: Accelerometer feature values, $V_i = \{v_1, v_2, ...v_n\}$
INITIALIZE: $NeighborPts \leftarrow \{\}$, list of potential cluster points
procedure ADAPTIVEDBSCAN($v_i$)
    if $NeighborPts$ is empty then
        append $v_i$ to $NeighborPts$;
        $\alpha \leftarrow v_i$
    else
        if REGIONQUERY($v_i, \alpha$) then
            append $v_i$ to $NeighborPts$;
            $\alpha \leftarrow mean$(NeighborPts);
            if $sizeof(NeighborPts) \geq MinPts$ then
                $A(i) \leftarrow getState(\alpha)$;
                declare $A(i)$;
        else
            if $sizeof(NeighborPts) < MinPts$ then
                $A(i) \leftarrow unknown$;
                declare $A(i)$;
            empty $NeighborPts$;
            append $v_i$ to $NeighborPts$;
            $\alpha \leftarrow v_i$;
procedure REGIONQUERY($v_j$,$\alpha$)
    if $v_j > \tau$ then $\epsilon \leftarrow \theta * \alpha$ ;
    else
        $\epsilon \leftarrow \epsilon_0$;
    if $|v_j - \alpha| < \epsilon$ then return true;
    else
        return false;
```

DBSCAN [4] is one of the most popular density based clustering algorithm. It requires complete data set to perform clustering and hence, is not appropriate for clustering real-time data. For BluePark we need a classifier which can process incoming data in real time. Moreover, the accelerometer data for different user states is observed to have large density differences, which DBSCAN cannot cluster. Algorithm 1 describes the steps of *Adaptive DBSCAN*. Input to the *AdaptiveDBSCAN* are features that are computed on accelerometer traces collected over non-overlapping time windows of a fixed duration. It is a sequence of features $\{v_1, v_2, ...v_n\}$, where v_i is the average variance of FFT energy for accelerometer window at time i. At any given time instance, we keep a list of potential cluster points in $NeighborList$ and a current ϵ value. The *regionQuery* function determines if an incoming data point should be included in the current cluster in $NeighborList$. If the incoming data is within ϵ distance of α, the current cluster centroid, it is included in the cluster. A cluster is discarded as noise if it has less points than $MinPts$ as in traditional DBSCAN and the accelerometer state is declared as $unknown$. For each new incoming data point the ϵ value is recomputed as a fraction of that data point. The fractional adapting coefficient is denoted as $0 \leq \theta \leq 1$. This enables us to work on real time incoming data and helps to accommodate large density differences among different user locomotive states. For data points with small values ($\leq \tau$) we use a default threshold ϵ_0. The $getState$ function returns a_{walk} or $a_{\neg walk}$ using a threshold on the value of α.

4.1.2 *WiFi State Classifier*

The WiFi State Classifier determines if a user is changing location continuously. As user changes location, set of WiFi APs (APs) visible on user's phone changes. We define a metric β to measure this change and it is used to determine the WiFi based state $W(t)$ by the WiFi State Classifier.

The WiFi signals are scanned continuously and are grouped into non-overlapping time windows of fixed duration Δ called as *WiFi scan window*. Each *WiFi scan window* consists of multiple WiFi scans. Let $S(t) = \{s_1, s_2, \ldots, s_N\}$, represent a set of unique SSIDs present in *WiFi scan window* starting at time t where s_i is SSID of a unique AP.

ParkSense [8] used Jaccard Index of WiFi APs to determine drive state but does not consider the frequency of their occurrences. It assumes the signal received by a receiver from an AP will disappear as soon it moves away from that AP. As indoor parking areas have multiple reflecting surfaces like walls, ceilings, pillars, etc., a signal from a far away AP can reach the receiver due to multi-path effect. Hence, it will be erroneous if one estimates rate of movement by only comparing the count of APs that are visible from successive positions of a vehicle. Moreover, the same AP may be visible multiple times in the same scanning window due to slow driving speed and restricted driving lanes.

For indoors, we argue, one needs to consider the frequency of occurrence of visible APs. Though multi-path may cause a far away signal to reach the receiver, such a signal traverses over a longer path compared to a signal from the AP, which is in direct line of sight. As a result, APs which are closer to the receiver will occur more frequently during the scanning window than the APs, which are distant. To capture the effect of multi-path, we propose a modified version of the Jaccard Index β, which includes frequency of occurrence of APs in its formulation. We define β for successive *WiFi scan windows* staring at time $t - \Delta$ and t as:

$$\beta(t) = \frac{\sum_{s \in S'} min\{f_t(s), f_{t-\Delta}(s)\}}{\sum_{s \in S'} max\{f_t(s), f_{t-\Delta}(s)\}} \qquad (1)$$

where $S' = S(t-\Delta) \cup S(t)$, and $f_t(s)$ is the number of times AP s occurs in *WiFi scan window*. successive windows. We use a threshold on $\beta(t)$ values to determine the WiFi based states w_{moving} or $w_{\neg moving}$, which are then sent to the States Merger module.

4.1.3 States Merger

States Merger determine current state of user using Accelerometer State Classifier and WiFi State Classifier states. Suppose at any time t, the current Accelerometer state is $A(t)$ and WiFi state is $W(t)$. The State Merger determines the current state of the user $C(t)$ by combining $A(t)$, $W(t)$, and previous state $C(t-1)$. There possible classes for $C(t)$ are: $walk$, $drive$, $still$, and $unknown$.

The States Merger combines $A(t)$, $W(t)$, and $C(t-1)$ are combined using the rules given in Table 1, where '*' denotes any possible state. If WiFi State Classifier detects the user to be in a $moving$ state but the Accelerometer State Classifier does not detect a $walk$ state then we infer the user's current state to be $drive$. Similarly, we conclude a $still$ state if the user is neither $moving$ and nor $walk$. A $walk$ state is detected whenever the Accelerometer State detects a $walk$ irrespective of all other sates whereas the previous state $C(t-1)$ is used when the Accelerometer based state is $unknown$.

4.2 Event Detection

The goal of event detection module is to detect state transitions, which corresponds to parking and un-parking events. For each state transition between *walk* and *drive*, we check if the user states, before and after the transition remains unchanged for more than ν amount of time to mark it as a legitimate transition. If the state changes within ν amount of time, we discard the corresponding state transition. We only look for transitions between *walk* and *drive* states.

$A(t)$	$C(t-1)$	$W(t)$	$C(t)$
a_{walk}	*	*	$walk$
$a_{\neg walk}$	*	$w_{\neg moving}$	$still$
		w_{moving}	$drive$
$a_{unknown}$	$unknown$	*	$unknown$
	$walk$	$w_{\neg moving}$	$unknown$
		w_{moving}	$walk$
	$drive$	$w_{\neg moving}$	$unknown$
		w_{moving}	$drive$
	$still$	$w_{\neg moving}$	$still$
		w_{moving}	$unknown$

Table 1: Rule for combining Accelerometer and WiFi based states to obtain the final state.

4.3 Event Localization

Event Localization module determines parking spot L_{id} for the WiFi signature sent by the mobile client. We used a WiFi based indoor localization system which was hosted on the cloud service. It is based on the well known Bayesian Inference technique called Horus [9]. WiFi based indoor localization system is popular due to its ubiquity and low cost. Moreover, BluePark already uses WiFi for locomotive state detection, this choice allows us to use the same for localization without any additional sensor, thereby eliminating other overheads. We use pillar numbers adjacent to parking spaces to identify a parking spot where a vehicle is parked.

5. EVALUATION

We compare BluePark's state detection with Google's Activity Recognition API (GARA) [2], which is the most popular activity detection service on Android. It is a part of the Google Play Service and is capable of detecting STILL, ON_FOOT, IN_VEHICLE, TILTING, ON_BICYCLE, and UNKNOWN states. We used Google Play Services version 6.1, with an update interval of 3s.

5.1 Setup for Experiments

We conducted our experiments in an underground parking area (172.5m×165m) of a popular shopping mall in New Delhi. We collected WiFi fingerprints for 56 parking spots. A total of 53 unique APs are visible from these parking spots.The standard size of a parking spot is 2.5m×5m. For localization accuracy, distance between estimated and ground truth location is computed using this standard size and map of the basement parking area. Five accelerometer sampling rates: 5Hz, 10Hz, 15Hz, 50Hz, and 99Hz were used. We tested BluePark on three on-body phone positions, in hand, in trouser pocket, and in backpack. GARA has its internal way of deciding sampling frequency. The parameters values of BluePark is given in Table 2 which were empirically derived.

	Parameters	**Classifier Threshold**
ASC	window size = 6s $MinPts = 2$ $\theta = 0.35$	$\alpha < 2$, then $A_{state} = a_{\neg walk}$ $2 \leq \alpha \leq 20$, then $A_{state} = a_{walk}$ $\alpha > 19$, then $A_{state} = a_{unknown}$
WSC	sampling interval = 5s window size = 20s	$\beta \geq 0.34$, then $w_{state} = w_{moving}$ $\beta < 0.34$, then $w_{state} = w_{\neg moving}$
ED	$\nu = 10s$	NA

Table 2: List of parameters used in Accelerometer State Classifier (ASC), WiFi State Classifier (WSC), and Event Detection(ED) and their values used for evaluation of BluePark

5.2 Accuracy of State Detection

We compare performance of our state detection module with that of GARA. We consider $drive$ and $walk$ states for comparison. We observe BluePark performs better than GARA for $drive$ detection by more than 20% at all sampling rates. It validates efficacy of our modified Jaccard Index formulation. In trouser pocket and backpack positions, GARA performs better than BluePark, whereas at hand phone position BluePark has better accuracy. The $walk$ detection performance of the two approaches are not significantly different when the sampling rate is 10Hz or higher.

5.3 Accuracy of Event Detection

We measure accuracy of event detection in terms of precision and recall. The precision and recall of GARA is 100% for all phone positions for both parking and un-parking events. For BluePark, there are cases of false positives at 5Hz and 10Hz. At sampling rate of 15Hz or higher, the precision and recall are all 100%. Therefore, we use 15Hz and higher sampling rate for further evaluations.

5.4 Latency in Event Detection

Latency in event detection is computed as time difference between detected time of an event and actual time, manually noted down. Cumulative distribution function of latency in event detection for BluePark in comparison to GARA is shown is Figure 2. BluePark detects parking transition within 20 seconds delay with probability of 0.89, whereas GARA takes around 30 seconds. The higher event detection latency of GARA is due to its low detection accuracy of $drive$ state.WiFi signature seen at the time of parking event is used to determine location of the parking spot. A high detection latency will results in an incorrect WiFi signature, thereby introducing an error in localization. Therefore, latency in event detection directly affects the event location accuracy.

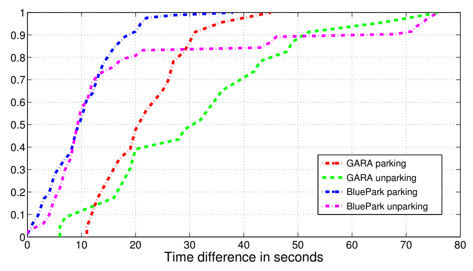

Figure 2: Cumulative distribution of time difference between ground truth and detected time for parking and un-parking activity.

5.5 Accuracy of Event Localization

Event localization accuracy is important for identifying where the vehicle was parked. Apart from detection latency, location accuracy depends on localization approach used. Figure 3 shows comparison of cumulative distribution of distance error. For measuring location accuracy of GARA, we used our modular design by replacing the state detection module with GARA and the rest of the system remains the same. The set of locations considered in computing BluePark and GARA location accuracy is a subset of the locations considered for inherent accuracy calculation. The median location error of BluePark is 10m and that of GARA is 15.4m.

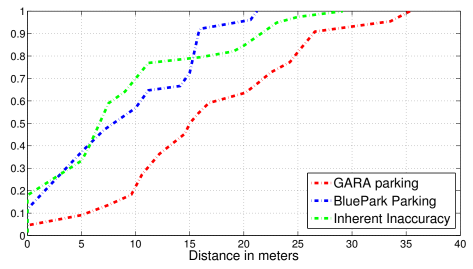

Figure 3: Cumulative distribution of location error in terms of meters.

Due to space constrains, we could not include all the results in detail. For detailed information, please refer [3].

6. CONCLUSION

In this paper, we presented BluePark, a collaborative sensing mechanism, using smartphone sensors, that can detect and localize parking and un-parking of vehicles in indoor parking lots in real time, without any user input. We proposed a combination of accelerometer and WiFi sensors to detect $drive$ and $walk$ states. It is a modular system and its state state detection and localization modules can be replaced by algorithm of user's choice. As our state detection algorithm uses unsupervised training for detection, it does not require per-user calibration. We proposed an event detection time bounded legitimate state transition detection. We presented our own algorithm for state detection and showed that it gives better accuracy, even at low sampling rate, as compared to GARA. The experimental results probe that BluePark is robust to common on-body phone positions.

7. ACKNOWLEDGEMENT

The authors would like to acknowledge the support provided by ITRA project, funded by DEITy, Government of India, under grant with Ref. No. ITRA/15(57)/Mobile/HumanSense/01.

8. REFERENCES

[1] http://outdoormediaplan.com/malls-media-kit.php. **http://outdoormediaplan.com/malls-media-kit.php**Accessed:2015-08-3.

[2] Recognizing the User's Current Activity. **http://goo.gl/5fm314** Accessed : 2014-08-16.

[3] Technical Report on BluePark. **https://goo.gl/1C8fKv**.

[4] Ester, M., peter Kriegel, H., S, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press (1996), 226–231.

[5] Hemminki, S., Nurmi, P., and Tarkoma, S. Accelerometer-based transportation mode detection on smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, ACM (New York, NY, USA, 2013), 13:1–13:14.

[6] Lan, K.-C., and Shih, W.-Y. An intelligent driver location system for smart parking. *Expert Systems with Applications 41*, 5 (Apr. 2014), 2443–2456.

[7] Nandugudi, A., Ki, T., Nuessle, C., and Challen, G. Pocketparker: Pocketsourcing parking lot availability. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14 Adjunct, ACM (New York, NY, USA, 2014), 963–973.

[8] Nawaz, S., Efstratiou, C., and Mascolo, C. Parksense: A smartphone based sensing system for on-street parking. In *Proceedings of the 19th Annual International Conference on Mobile Computing &#38; Networking*, MobiCom '13, ACM (New York, NY, USA, 2013), 75–86.

[9] Youssef, M., and Agrawala, A. The horus wlan location determination system. In *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, MobiSys '05, ACM (New York, NY, USA, 2005), 205–218.