Unlocking the smartphone’s sensors for smart city parking

Jean-Gabriel Krieg a, Gentian Jakllari a,*, Hadrien Toma b, André-Luc Beylot a

a Department of Telecommunications and Networks, ENSEEIH, University of Toulouse, France
b ENSICAEN, France

1. Introduction

Someone named Bob is cruising around downtown looking for a place to park. Using an application on his smartphone, he identifies a free spot in the next block, saving him valuable time. Indeed, a Parsons Brinckerhoff study [2] of New York City’s Chinatown showed that on weekends 41% of drivers spend more than 20 min looking for on-street parking. This figure increases to 54% on weekdays. Using this application also reduces Bob’s carbon footprint – in congested urban areas 30% of the traffic is due to drivers cruising for a parking space [3,4]. Once parked, Bob leaves the vehicle and goes about his business. The application keeps track of his movement and can detect automatically when he returns to the vehicle and pulls out of the parking space. It computes the time Bob spent parking, charges his account and then marks the respective parking spot as available again. This saves Bob the inconvenience of having to anticipate how long he will be away, go to the ticket machine to pay, and then place the receipt on the dashboard. At the same time, it saves the city a significant amount of money. Cincinnati, for example, is owed about $12 million from unpaid parking tickets dating back to 2005 [5]. What is more, as the application relies on the smartphone’s sensors and infrastructure already available, cities will no longer have to invest in building and maintaining an extensive parking payment infrastructure. Finally, the application has a negligible effect on the battery.

* Corresponding author.
E-mail addresses: krieg@enseeiht.fr (J.-G. Krieg), jakllari@enseeiht.fr (G. Jakllari), hadrien.toma@ecole.ensicaen.fr (H. Toma), beylot@enseeiht.fr (A.-L. Beylot).

1574-1192/© 2017 Elsevier B.V. All rights reserved.
Unfortunately, despite some efforts [6], no such application has gained wide adoption. The reason is twofold: First, providing real-time parking availability information requires knowing in real-time when a user vacates a parking space. However, automatically identifying an unparking event with almost 100% accuracy while relying on smartphone sensors remains an open problem. Second, such an application needs a broad user base to be successful and requiring a city-wide roll-out of a smartphone application creates a high barrier to entry. The SF-Park pilot project [7] addresses both challenges by installing dedicated in-ground parking sensors on approximately 7000 on-street parking spaces. This required $18 million as start-up cost, or roughly $2500 per space [8] – a prohibitive cost for many cities. One might be tempted to think that users should just notify when vacating a parking space. However, the Google Open Spot experiment has shown that most users neglect to inform the system when they vacate a parking space [9]. Smartphone solutions based on GPS have been proposed to automatically detect unparking events [10–12] and shown to be accurate. However, relying on GPS can quickly drain the battery, a non-starter for most smartphone users. ParkSense [3] can detect unparking events using only the smartphone’s wireless interface, reducing energy consumption. However, in order to do so it relaxes the requirement for accuracy. Furthermore, it does not address the high barrier-to-entry issue.

We present SmartPark, a smartphone-based system that addresses the dual challenge facing the automatic on-street parking management. It can detect unparking events with accuracy reaching 100% and introduces an approach for overcoming the barrier-to-entry issue, all with a negligible impact on battery life.

**Challenge 1 – Detecting unparking events.** SmartPark divides the problem into two parts: identifying that the user is indeed inside a vehicle of the same type as the one she parked with, and identifying that the user is inside the same vehicle as the one with which she parked. Identifying that a user is inside the same type of vehicle is challenging as, once parked, people can use a variety of transportation modes. Thus, SmartPark introduces an approach that can detect 9 transportation modes – walking, bicycle, bus, car, subway, motorbike, train, tram, airplane – covering virtually all transportation available. It leverages the smartphone sensors to generate a large number of features and uses a supervised learning approach based on random forests [13] to classify sensor readings into one of the possible transportation modes in real-time. To identify that a user is inside the same vehicle she parked with, SmartPark introduces a novel similarity coefficient for matching two physical locations using Wi-Fi and cellular base station signals.

**Challenge 2 – High barrier to entry.** To avoid the need for a city-wide roll-out, SmartPark introduces an approach that enables an incremental deployment, starting with as little as 10% of the total users. Using information from the fraction of users using SmartPark, it can compute analytically the probability that a user without SmartPark has parked on a given parking space. Thus, SmartPark can provide reliable parking availability information even with a small initial adoption, paving the way for an incremental deployment. A city could incentivize a small percentage of parking users to adopt SmartPark with the expectation that its initial success will lead to a spontaneous large-scale adoption by users.

In summary, our contributions can be summarized as follows:

- We introduce the first system capable of detecting virtually every mode of transportation with accuracy approaching 100% while relying exclusively on the sensors available on modern smartphones (Section 3).
- We introduce an algorithm for location matching, which relies on the readings from Wi-Fi and cellular base stations (Section 4). Combined with the transportation mode detection, it enables SmartPark to detect unparking events with accuracy approaching 100% while having a negligible impact on battery life.
- We introduce an analytical approach for estimating parking availability even when only a fraction of users uses SmartPark (Section 5).
- We evaluate SmartPark in two ways: using simulations and in the wild (Section 6). Simulations (Section 6.1) using a variety of configurations show that with as little as 10% of users adopting it, SmartPark can estimate parking availability with accuracy above 80%. The accuracy reaches over 90% when 20% or more of the users adopt SmartPark. Over 30 h of experiments in the wild (Section 6.2) using 7 different smartphones with the help of 12 volunteers from 3 different cities show that: (a) SmartPark correctly detects unparking events 97% of the time while triggering zero false positives, and (b) SmartPark has a minimal impact on the battery consumption. Running SmartPark on a fully charged LG Google Nexus 5 for 5 h straight caused only about 4% drop in battery charge.

2. **System overview**

SmartPark enables the automatic detection of a user vacating a parking space by leveraging the smartphone’s sensors and the ubiquitous Wi-Fi and cellular infrastructure. At the heart of SmartPark is the ability to mine readings from seven different sensors so as to accurately detect virtually any transportation mode available in modern city centers. This large number of sensors is necessary to overcome the inherent erroneous nature of the inexpensive sensors with which modern smartphones are equipped. Once the transportation mode is detected, SmartPark can tell whether a user is back in the same kind of vehicle as when she parked. With a similarity matching algorithm using the Wi-Fi and cellular signals received, SmartPark can distinguish a user getting back in her vehicle and pulling out of the parking spot, making it available again, from a user getting a ride in a different vehicle. Fig. 1 shows the architecture of SmartPark. SmartPark can be in one of the four following states:

---

1 A user selects on the application a parking space among those available, making the problem of knowing when a place is occupied trivial. Dealing with malicious behavior, a user selecting a place but parking at another, is left as future work.
Fig. 1. SmartPark architecture for real-time parking availability. When a user parks, SmartPark creates a location profile using readings from Wi-fi and cellular base stations and a vehicle profile using the Transportation Mode detection module. Once the user leaves the vehicle, SmartPark checks periodically the transportation mode so as to detect when she is again the same kind of vehicle. If that is the case, it runs a location matching and, if positive, and the user starts pulling away, it knows the user has vacated the parking spot.

1. The user is looking for parking. SmartPark detects automatically the user’s vehicle type and sends a request to the SmartPark server via the Wi-Fi or cellular network asking for the available parking spots in the vicinity.
2. The user parks. SmartPark scans the Wi-Fi and cellular base stations available to create a location profile and saves it for future use.
3. The vehicle is parked. The user can go about her business: walking around, taking the bus, subway, tram or any other transportation mode. SmartPark, in the mean time, triggers a transportation mode detection every 10 s.
4. The user is vacating the parking spot. Smartphone detects this event because: (a) the user is using the same transportation mode as when she parked, and (b) she is back at the same location. Once the user pulls out of the parking spot, SmartPark sends a message to the SmartPark server, which will mark the respective parking spot as available.

Note that the number of the available parking spots depends on the size of the vehicles and how people park. SmartPark proposes to work with cities and adopt their regulations regarding the size of parallel parking spots [14]. Smartphone relies on two major functionalities to deliver on its promise:

- Transportation mode detection.
- Location matching.

We describe each in detail in the next sections.

3. Transportation mode detection

In this section, we present SmartPark’s module for identifying the transportation mode using a smartphone. Needing only 2 s of sensor readings, it can decide with accuracy reaching 100% whether a user is using one of the following transportation modes: walking, bicycle, bus, car, subway, motorbike, train, tram, airplane. This unprecedented granularity, speed and accuracy is achieved thanks to two innovations: One, while previous approaches rely on a single sensor, usually the accelerometer [3], SmartPark builds an approach capable of exploiting the multitude of sensors available on modern smartphones. Two, SmartPark introduces an approach based on Random Forests [13] for classifying sensor readings into transportation modes in real time.

3.1. Exploiting smartphone’s multiple senses

A key insight underlying SmartPark is that all sensors with which modern smartphones are equipped can provide useful information as to the user transportation mode. Sound and light levels while riding a motorbike, for example, can be very different from those inside a car or bus; pressure levels in a subway usually running underground may be different from those inside a tram. At the same time, these are inexpensive and inherently erroneous sensors and cannot be the lone source for transportation mode detection.

To corroborate our intuition, we perform large scale experiments with the help of 12 volunteers using 7 different smartphone models (the experimental setup is described in detail in Section 6.2). Fig. 2 shows an analysis of the sound
level readings collected while the volunteers were using one of the 9 transportation modes SmartPark supports. The data shows that there is some correlation between the sound sensor readings and the transportation modes. For example, the readings while riding a motorbike, Fig. 2(a), have the highest median value and a tight box. On the other hand, Fig. 2(b) shows that sound levels collected from pedestrians are difficult to analyze and classify, probably due to the fact that a pedestrian is exposed to all kinds of sound noises in city centers.

A similar phenomenon is observed with the frontal acceleration readings show in Fig. 3. Here the subway produces the highest median value and a tight box, consistent with the acceleration one experiences repeatedly at every station. Nonetheless, measurements show significant overlap in the frontal acceleration readings of the different transportation modes.

To overcome the limitations of using a single sensor, SmartPark builds an approach capable of leveraging several of the sensors available on off-the-shelf smartphones. As Fig. 4 shows, using multiple sensor readings can significantly improve the capability to distinguish different transportation modes. Using readings from two sensors – the frontal acceleration and sound levels – and without using any sophisticated classification algorithms enables detecting three transportation modes: car, airplane and train. This is something the experimental data showed it cannot be done while using only one sensor.

As a result, SmartPark uses several of the sensors available on modern smartphones to detect the transportation mode. Specifically, it uses the following:

- Accelerometer (three axes).
- Gyroscope (three axes).
- Orientation (three axes).
- Magnetic field sensor (three axes).
- Luminosity sensor.
3.2. Feature extraction

Table 1 shows the summary statistics we compute for the temporal, frequency and wavelet transform of every sensor reading. We have erred on the side of being exhaustive and selected the majority of the commonly used summary statistics.
Table 1
Statistics extracted from each sensor reading to create the classification features.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Temporal, Frequential, Wavelets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location statistics</td>
<td>Mean, Median, Harmonic mean</td>
</tr>
<tr>
<td></td>
<td>Geometric mean, RMS</td>
</tr>
<tr>
<td>Spread statistics</td>
<td>STD, Var, 1st-Q, 3rd-Q</td>
</tr>
<tr>
<td>Shape statistics</td>
<td>Kurtosis, Skewness, Entropy</td>
</tr>
<tr>
<td>Integral and derivative</td>
<td>Surface, Growth rate</td>
</tr>
<tr>
<td>5 Peaks and 5 Valleys</td>
<td>Value, Magnitude, FHWM, 2 Inter-distances</td>
</tr>
</tbody>
</table>

for a total of 64. We collect 15 sensor readings for which we calculate the FFT and the wavelet transform. For each of the three versions of the signal – temporal, frequency and wavelet transform – we compute the 64 summary statistics. This gives a total of $15 \times 3 \times 64 = 2880$ features. In Section 3.3, we present a supervised learning approach that uses these features for identifying the 9 transportation modes SmartPark supports.

3.3. Classification

Several classification models have been proposed in literature with the Partial Least Squares (PLS) [15] (itself a major evolution of Principal Component Analysis (PCA)) one of the more popular. It tries to find a linear regression model by creating a new plane from the feature and class variables [16]. PLS components are selected so that their covariance with the class is maximized. Thus from a list of predictors, PLS extracts the linear combinations of the predictors — also called latent factors. PLS regression is particularly efficient in cases such as the one considered here in which the feature space is significantly bigger than the class space. However, as PLS is based on a linear approach it needs to compute many linear combinations of the features so as to predict one or more transportation modes [17]. Given the variety of features SmartPark uses, this would result in a very large number of computations, making it a poor choice for a smartphone-based solution. Therefore, we opted for a tree-based classifier.

Classification trees are hierarchical structures with leaves representing class labels and branches conjunctions of features leading to those class labels. We use a training set to grow a classification tree. To split an internal tree node, we look for the feature and threshold for this feature that partitions the training set into two branches such that the disorder in each branch is minimized. To quantify the disorder, we use the entropy, calculated at each node, $i$, as follows:

$$D_i = - \sum_{m \in \text{modes}} \frac{n_{m,i}}{N_i} \log\left(\frac{n_{m,i}}{N_i}\right)$$

where $N_i$ is the number of training set elements in the $i$th node and $n_{m,i}$, the number of training set elements corresponding to the $m$th class in the $i$th node. Fig. 5 shows an example of a classification tree section produced by SmartPark.

While a single tree classifier can work well, it is known to suffer from overfitting, especially when it uses a large number of features. To minimize this effect, we use the Random Forest (RF) [13] method, an ensemble learning approach combining multiple tree predictors. Each tree is grown using the same training set and a randomly selected subset of the available features. We use this approach and compute 500 such trees offline. The trees are loaded into SmartPark which lets each tree make a classification decision and uses majority voting for the final decision – a step well within the computation capabilities of modern smartphones.

4. Location matching: Is this MY vehicle?

When a user parks her vehicle (car/motorbike/bicycle), she enters the parking spot number in the SmartPark application. Automatically, SmartPark creates a location profile by leveraging the ubiquitous Wi-Fi and cellular infrastructure. In particular, the location profile is computed as follows:

$$S_1 = (S_1(1), S_1(2), \ldots, S_1(a))$$
$$W_1 = (W_1(1), W_1(2), \ldots, W_1(a))$$
$$C_1 = (C_1(1), C_1(2), \ldots, C_1(b))$$
$$Wc_1 = (Wc_1(1), Wc_1(2), \ldots, Wc_1(b))$$

where $S_1$ represents the SSIDs of the $a$ access points, $W_1$ the RSSIs of the corresponding APs, $C_1$ the Cell IDs of the $b$ stations and $Wc_1$ the RSSIs of the corresponding stations.

When SmartPark detects that the user is using the same transportation mode as the one she did when she parked, it checks whether the location is also the same.2 For this, it computes a profile of the current location and compares it to

2 After a user parks the car there is the possibility she meets a friend offering her a ride back.
the one stored at the time of parking. The comparison, however, is not straightforward. RADAR [18], for example, uses the Euclidean distance in RSSI space whereas SensLoc [19] uses the Tanimoto coefficient [20]. ParkSense [3], observing that WiFi signals usually come from indoors and, therefore, are very unreliable, simply compares the number of access points in common between two location profiles. Our experience, however, has shown that although the WiFi signals are often weak outdoors, the signal strength remains a strong indicator of distance. Therefore, SmartPark introduces a new approach for comparing two location profiles built on three principles:

- **Proportion of APs found**: we take into account the proportion of APs in common between two location profiles.
- **Compare raw RSSIs values**: instead of a binary model, we compare the RSSI values of the two respective profiles.
- **Favor the highest RSSI values**: we add a weight to favor the APs/BSs with the highest RSSI values. The idea being that APs/BSs with stronger RSSI are more likely to be closer to the actual parking location.

Relying on these principles, SmartPark compares two location profiles, 1 and 2, by using a similarity coefficient, $C$, computed as follows:

$$
C = \frac{1}{\sum_{i=1}^{a} Ws_1(i)} \sum_{i=1}^{a} Ws_1(i) (1 - |\frac{Ws_1(i) - Ws_2(i)}{Ws_1(i)}|) \\
+ \frac{1}{\sum_{i=1}^{b} Wc_1(i)} \sum_{i=1}^{b} Wc_1(i) (1 - |\frac{Wc_1(i) - Wc_2(i)}{Wc_1(i)}|)
$$

where $Ws_1(i) - Ws_2(i)$ and $Wc_1(i) - Wc_2(i)$ are the differences in RSSI received from the same access point, $i$, and cellular base station, $i$, respectively, present in both location profiles. Two location profiles sharing no WiFi access points and no cellular base stations will have zero as similarity coefficient. Two location profiles with similarity coefficient above a given threshold are considered to represent the same physical location. In Section 6.2.4 (Fig. 12), we show that using 0.5 as similarity threshold SmartPark exhibits almost perfect behavior.

5. Smart parking with hidden users

In this section, we consider the realistic case in which not all parking users use SmartPark and introduce an approach enabling SmartPark to still deliver useful information regarding on-street parking availability. This would allow an incremental deployment of SmartPark: a particular city incentivizes a number of early testers to adopt SmartPark with the expectation that a successful deployment will trigger a wide adoption by parking users themselves.

While SmartPark is running, there will be two categories of parking users in a particular area: those that use SmartPark and those who do not. We refer to the latter as hidden users. In the early stages, we expect hidden users to outnumber SmartPark users, with the roles being reversed as a bigger number of users adopt SmartPark. However, the existence of some hidden users can never be excluded, unless a particular city makes SmartPark mandatory for using its parking spaces.

Therefore, SmartPark computes and offers its users an occupancy probability for every parking space instead a zero or one answer. To calculate this probability, we start with the two basic detectable events:
1. A SmartPark user moving her vehicle from a parking space makes that space available.
2. A SmartPark user parking her vehicle makes the respective parking space occupied.

In the presence of hidden users, the challenge for SmartPark is to compute the occupancy of a parking space between the two detectable events.

Let $t_0$ be the time at which a SmartPark user unparks her vehicle from a particular parking space, $n$. Let $P_n^{\text{busy}}(t_i)$ be the occupancy probability of parking space $n$ at a later time, $t_i$. We have:

$$P_n^{\text{busy}}(t_i) = P(\text{occupied at time } t_i | \text{released at } t_0)$$

Which is equivalent to:

$$P_n^{\text{busy}}(t_i) = P(\text{a user has parked before time } t_i | \text{released at } t_0)$$

Let us assume for a moment that the parking duration, $T_{park}$, is constant; we will relax this assumption later. If the parking space is free, it means there was no parking event between $t_i - T_{park}$ and $t_i$. Assuming user arrivals follow a Poisson distribution with event rate $\lambda$, the probability that $k$ users arrive before $t_i$ is:

$$\left(\frac{\lambda t_i}{k!}\right) e^{-\lambda t_i}.$$ Thus, the probability that the parking space is free is:

$$P_{n,\text{constant}}^{\text{free}}(t_i) = \begin{cases} e^{-\lambda T_{park}} & \text{if } t_\Delta > T_{park} \\ e^{-\lambda t_\Delta} & \text{if } 0 < t_\Delta < T_{park} \end{cases}$$

where $t_\Delta = t_i - t_0$.

To relax the assumption that $T_{park}$ is constant, we assume that it follows an exponential distribution with rate $\mu$. Thus, we have:

$$P_{n,\text{constant}}^{\text{free}}(t_i) = \int_0^\infty P_{n,\text{constant}}^{\text{free}}(t_i) dT_{park}$$

Finally, the probability that a parking place, $n$, is occupied at time $t_i$, $t_\Delta$ after the time a SmartPark user left the particular place is:

$$P_n^{\text{busy}}(t_i) = 1 - P_n^{\text{free}}(t_i) = U * (1 - e^{(-\lambda + \mu) t_\Delta})$$

where $U$ is the stationary probability:

$$U = \frac{\lambda}{\lambda + \mu}$$

Solving Eq. (3) requires knowledge of $\lambda$ and $\mu$.

Calculating an estimate of $\mu$ can be done by relying on the parking time of SmartPark users, the assumption being that the parking time of hidden users is similar. SmartPark maintains a variable, $t_p$, that it updates based on the parking time of SmartPark users during an observation window, $w_{obs}$. It then computes an estimate of $\mu$ as follows:

$$\hat{\mu} = \frac{1}{t_p}$$

Estimating $\lambda$ is far more challenging since SmartPark, by definition, has no knowledge as to their activity. However, if we were to know what part of users have SmartPark installed, which we call the monitored fraction, we could infer the number of hidden user arrivals. Let $n_{\text{smart}}$ and $n_{\text{hidden}}$ be the number of SmartPark and hidden user arrivals, respectively, in a time window $w_{obs}$. Let $f_m$ be the monitored fraction. We have:

$$n_{\text{hidden}} = \frac{n_{\text{smart}}(1 - f_m)}{f_m}$$

SmartPark estimates $n_{\text{smart}}$ by looking at the average SmartPark user arrivals across the parking area it manages during the observation time window. To estimate $f_m$, the monitored fraction, we adopt an approach from [8].

Finally, we can estimate $\lambda$ as follows:

$$\hat{\lambda} = \frac{n_{\text{hidden}}}{w_{obs}}$$

6. Performance evaluation

We evaluate SmartPark in two ways: First, we use a simulator written in Python to test SmartPark in a large number of scenarios. Second, we evaluate SmartPark in the wild using an Android implementation on off-the-shelf smartphones.
Table 2
Simulation parameters for the metered on-street parking scenario.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>User arrivals</td>
<td>Inter-arrival exponential law (12 users per hour)</td>
</tr>
<tr>
<td>Total number of users</td>
<td>3000</td>
</tr>
<tr>
<td>Monitored fraction</td>
<td>0.1–0.5</td>
</tr>
<tr>
<td>Parking capacity</td>
<td>80–200</td>
</tr>
<tr>
<td>Average parking time</td>
<td>15 min</td>
</tr>
<tr>
<td>Simulation time</td>
<td>10 h</td>
</tr>
</tbody>
</table>

6.1. Simulations

We implemented a parking simulator in Python 3.5.2. Parking users arrive following a parametrizable distribution and a fraction of them, the monitored fraction, utilizes SmartPark. Users choose uniformly at random among the parking spaces available and stay parked for a time that is randomly chosen following the exponential distribution.

For each simulation, we calculate SmartPark’s accuracy by keeping track of all its decisions regarding the availability of parking spaces and comparing to the ground truth. Specifically, let $P_i^{busy}(t)$ be SmartPark’s calculation of the occupancy of parking space $i$ at time $t$ (calculated using Eq. (3)), $S_i(t)$, a binary variable showing the ground truth, $C$, the number of parking spaces and $t_{end}$, the simulation time, we have:

$$\text{Accuracy} = \frac{C}{\sum_{i=1}^{C} \int_{0}^{t_{end}} (|P_i^{busy}(t) - S_i(t)|) dt}$$

(4)

6.1.1. Metered on-street parking

We first consider metered on-street parking, the most popular form of parking management in busy downtowns.

Setting: Table 2 summarizes the simulation parameters. We use the exponential distribution to simulate the inter-arrival of the parking users with an average of 12 new users per hour. A total of 3000 users is used. While the user arrival is kept constant, we vary the number of spaces available so as to create scenarios representative of parking demand in different city sections: overcrowded, highly crowded/high churn, moderately crowded/medium churn and lightly crowded/low churn areas.

Performance metric: The metric of interest is accuracy as defined by Eq. (4). SmartPark’s accuracy depends on two parameters: the monitored fraction, that is, the percentage of users having SmartPark, and the size of the observation window, $w_{obs}$. Therefore, we evaluate the accuracy as function of these two parameters.

Results: Fig. 6 shows the occupancy and number of parking events for the overcrowded (50 parking spaces), high-churn (80 spaces), medium-churn (100 spaces) and low-churn (200 spaces) parking-demand scenarios. While the number of parking events is (by design) the same across all scenarios, the parking occupancy is very different. Fig. 6(a) models the case in which there are less parking spaces available than users wishing to park. A user will look between 5 and 10 min for a parking space before abandoning. In this scenario 33% of users end up abandoning.

For all the scenarios evaluated, Fig. 7 shows SmartPark’s accuracy is over 80% with only 10% monitored fraction, exceeding 90% when the monitored fraction is set to 50%.

For a more in-depth understanding of the results, Fig. 8 shows the occupancy of a particular parking space selected at random for each of the low, medium and high-churn scenarios. The monitored fraction is set to 50%. The data shows that SmartPark’s occupancy estimate closely follows the ground truth, with average accuracy of approximately 92%, 91% and 83% for the high, medium and low-churn scenarios, respectively. SmartPark relies on its users for estimating the occupancy rate of the hidden users, explaining why the high-churn scenario generates the best accuracy.

Finally, Fig. 9 shows the impact of the observation window, $w_{obs}$, on SmartPark’s accuracy. The data shows that SmartPark achieves the best accuracy when the observation window is between 10 and 50 min. Smaller window values tend to miss some parking-user arrivals and/or departures, underestimating the number of hidden users. On the other hand, big window values can make the estimates of parking times and parking-user arrivals insensitive to local changes and effects.

6.1.2. Municipal garage

For the second set of simulations, we consider a municipal garage, usually the second most used form of city parking.

Setting: Table 3 summarizes the simulation parameters. The main difference with the metered on-street parking is that the average parking time now is 10 h, consistent with the long term parking allowed in a garage. As we did in Section 6.1.1, we evaluate SmartPark’s accuracy as function of the monitored fraction, $f_m$.

Results: Fig. 10(a) shows the parking events and garage occupancy during the 10 h simulation. The data shows that users arrive between 8:00–9:00 a.m. and leave between 4:00–7:00 p.m., corresponding to the working time schedule. Fig. 10(b) shows the parking occupancy of a parking space where a hidden user happened to park. SmartPark, using data
(a) The overcrowded scenario presents an occupancy near 100%. Some users have to abandon.

(b) The high churn scenario presents an occupancy near 100%.

(c) The medium churn scenario presents an occupancy near 80%.

(d) The low churn scenario presents an occupancy near 30%.

**Fig. 6.** The number of parking spaces occupied and that of parking events during the simulation for four parking-demand scenarios.

Finally, **Fig. 10(c)** shows that with only 10% of the parking users installing SmartPark, its accuracy is above 80%, reaching over 90% when 20% or more of the users adopt it. This data is very encouraging as it shows an incremental deployment of SmartPark is possible.
6.2. Deployment

In this section, we evaluate SmartPark in the wild using an Android implementation and with the help of 12 volunteers. The evaluation is divided into three parts: In the first part, we evaluate SmartPark’s transportation mode detection module. In the second part, we evaluate SmartPark’s location matching module. Finally, in the third part, we evaluate SmartPark as a whole.

**Part I: Transportation Mode Detection**

**Method and data collection:** Twelve volunteers are given one of the 7 smart devices, Sony Xperia E3, LG Google Nexus 5, LG Google Nexus 6, LG G2, Huawei Honor 4X, Samsung Galaxy S5, Samsung Galaxy S3 mini, and are asked to enter regularly...
Fig. 9. Accuracy of per spot occupancy estimation for different observation windows, $w_{obs}$. SmartPark adopts 20 min as the default value.

(a) Garage occupancy across time. (b) Occupancy of a parking space where a hidden user parks. SmartPark accurately tracks the occupancy using the data it collects from monitored users elsewhere in the garage.

(c) Accuracy for different values of the monitored fraction, $f_{m}$.

Fig. 10. Municipal garage scenario.

their transportation mode as they go about their normal daily lives. SmartPark records the 15 sensor readings, as described in Section 3, with the highest frequency rate available, often 200 Hz, computes the explanatory variables, and makes its own decisions about the transportation mode using 500 pre-computed classification trees.\footnote{The trees are trained using a different set of data.} The experiment runs for a total of
Table 4
Random Forest with 500 trees achieves 98.72% accuracy in transportation mode detection.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>Kappa</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLS</td>
<td>0.69</td>
<td>0.67</td>
</tr>
<tr>
<td>Single tree</td>
<td>0.54</td>
<td>0.53</td>
</tr>
<tr>
<td>Random forest</td>
<td>0.99</td>
<td>0.98</td>
</tr>
</tbody>
</table>

Table 5
SmartPark system allows to largely increase the performance of current transportation detection system, without involving a high latency.

<table>
<thead>
<tr>
<th>Mode</th>
<th>SmartPark [21]</th>
<th>[22]</th>
<th>[23]</th>
<th>[12]</th>
<th>[10]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bicycle</td>
<td>99%</td>
<td>81%</td>
<td>87%</td>
<td>66%</td>
<td>89%</td>
</tr>
<tr>
<td>Bus</td>
<td>98%</td>
<td>78%</td>
<td>58%</td>
<td>67%</td>
<td>88%</td>
</tr>
<tr>
<td>Car</td>
<td>99%</td>
<td>72%</td>
<td>96%</td>
<td>86%</td>
<td>88%</td>
</tr>
<tr>
<td>Subway</td>
<td>96%</td>
<td>65%</td>
<td>53%</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Motorbike</td>
<td>98%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Pedestrian</td>
<td>100%</td>
<td>95%</td>
<td>81%</td>
<td>95%</td>
<td>89%</td>
</tr>
<tr>
<td>Train</td>
<td>100%</td>
<td>68%</td>
<td>-</td>
<td>-</td>
<td>98%</td>
</tr>
<tr>
<td>Train</td>
<td>99%</td>
<td>84%</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Airplane</td>
<td>99%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Motorized</td>
<td>99%</td>
<td>74%</td>
<td>54%</td>
<td>95%</td>
<td>76%</td>
</tr>
<tr>
<td>Latency</td>
<td>2 s</td>
<td>4 s</td>
<td>8 s</td>
<td>1 s</td>
<td>2 s</td>
</tr>
</tbody>
</table>

Table 6
Confusion matrix for SmartPark.

<table>
<thead>
<tr>
<th>Transport</th>
<th>Bicycle</th>
<th>Bus</th>
<th>Car</th>
<th>Subway</th>
<th>Motorbike</th>
<th>Pedestrian</th>
<th>Train</th>
<th>Tram</th>
<th>Airplane</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bicycle</td>
<td>802</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>99.50%</td>
</tr>
<tr>
<td>Bus</td>
<td>0</td>
<td>837</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>47</td>
<td>0</td>
<td>0</td>
<td>97.51%</td>
</tr>
<tr>
<td>Car</td>
<td>3</td>
<td>8</td>
<td>1747</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>99.04%</td>
</tr>
<tr>
<td>Subway</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>583</td>
<td>0</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>95.57%</td>
</tr>
<tr>
<td>Motorbike</td>
<td>3</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>835</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>98.00%</td>
</tr>
<tr>
<td>Pedestrian</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>973</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100.00%</td>
</tr>
<tr>
<td>Train</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>830</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100.00%</td>
</tr>
<tr>
<td>Train</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1506</td>
<td>0</td>
<td>0</td>
<td>99.54%</td>
</tr>
<tr>
<td>Airplane</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>1037</td>
<td>0</td>
<td>98.76%</td>
</tr>
</tbody>
</table>

about 28 h during which SmartPark makes 9282 transportation-mode decisions. To measure the accuracy, we compare what the user entered with the transportation mode computed by SmartPark.

**Basis for Comparison:** We compare SmartPark to 5 state-of-the-art approaches, namely Reddy [23], Stenneth [10], Zheng [12], Wang [22] and Peaks [21].

6.2.1. Classification approach

First we investigate whether choosing Random Forest over PLS or a single tree is the right decision. To exclude the element of luck from the results, we use Cohen’s kappa criteria:

\[ \kappa = \frac{p_o - p_e}{1 - p_e} \]

where \( p_o \) represents the observed agreement between the real and predicted transportation mode and \( p_e \) represents the probability of randomly making the correct decision.

Table 4 shows that Random Forest, as implemented by SmartPark, is by far the most accurate approach, almost doubling the accuracy of a single tree and PLS.

6.2.2. Classification accuracy

Next, we evaluate the accuracy of SmartPark and compare it to five state-of-the-art approaches. Table 5 shows that: One, SmartPark is by far the most accurate. Two, SmartPark detects by far the most modes of transportation. The closest in terms of modes of transportation supported are Peaks and Wang which support 5 to SmartPark’s 9.

Finally, we focus on SmartPark for a more detailed analysis of its classification performance. The confusion matrix, Table 6, shows that SmartPark detects every transportation mode with accuracy that never drops below 95%. Equally important, the number of false positives is negligible.

6.2.3. Classification accuracy using a subset of sensors

The experiments so far show that SmartPark is highly accurate when using most sensors available on modern smartphones. However, due to privacy concerns, certain users may be reluctant to adopt an application which collects readings
from sensors such as the microphone, for example. Considering that SmartPark utilizes 7 (Section 3.1) sensors, it is unrealistic to provide measurement data for all possible combinations. Instead, we focus on 4 informative combinations: (a) Accelerometer, sound, as the two best sensors; (b) Barometer, accelerometer, gyroscope, as the top three sensors when excluding the microphone; (c) Gyroscope, light, sound, accelerometer; (d) All the sensors except light, the most energy consuming, and sound, for privacy concerns.

Fig. 11 shows that, as expected, using less sensors reduces SmartPark’s accuracy. However, the impact is limited. Even with only 2 sensors, SmartPark’s accuracy stands at 82%, reaching 94% when excluding only the microphone and light sensors.

**Part II: Identifying Own Vehicle**

**Method and data collection:** A volunteer is given one of the smart devices and asked to perform a large number of parking and unparking events with her vehicle in the course of several days. Then she is asked to perform a large number of what we call “false” parking/unparking events: she parks her vehicle, goes about her business then takes a ride back in a friend’s vehicle. To create the ground truth, she inputs on her phone the kind of event (parking or unparking) she performs every time. By the end of the experiment, the user performed 29 real parking/unparking and 27 false parking events.

We collect all the sensor readings generated during this period, use it to make parking/unparking decisions offline and compare them to the ground truth. To make the parking decisions we use SmartPark, ParkSense (the two versions proposed in [3]), the traditional and weighted version of Tanimoto index [24] and the Jaccard Index [25]. For two locations, $S_1$, $S_2$, the Jaccard Index, $J$, is computed as follows:

$$J = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

**6.2.4. Overall performance**

Fig. 12 shows the ROC curves$^4$ for different values of the similarity threshold (Eq. (2), Section 4). The data shows that SmartPark outperforms all the other approaches. In particular, using 0.5 as similarity threshold SmartPark exhibits almost perfect behavior: it correctly identifies 97% of the real parking/unparking events while triggering zero false positives.

**6.2.5. Impact of distance**

SmartPark identifies a user’s own vehicle by performing location matching (Section 4), which was shown in Section 6.2.4 to perform very well in a real-world scenario. In this experiment, we take a more targeted approach and try to deceive SmartPark in an effort to determine its limits. A user parks her car, goes about her business and, instead of going back to her car, is offered a ride by someone else in a second car of a similar model - thereby generating false unparking events all the time. The second car is placed at different distances from the user’s car, as shown in Fig. 13(b).

Fig. 13(a) shows that, as expected, when the second car is placed at virtually zero distance from the user’s car, SmartPark cannot distinguish the two. However, as the distance is increased so is SmartPark’s accuracy. Once the distance between the two cars reaches 20 m, SmartPark’s accuracy reaches 100%.

**Part III: Overall System Performance**

$^4$ True detection rate = True Positives/(True Positives + False Negatives), False Positive Rate = False Positives/(False Positives + True Negatives).
Fig. 12. ROC curves for all the methods considered. Using 0.5 as similarity threshold, SmartPark correctly identifies 97% of the real parking/unparking events while triggering zero false positives.

(a) False Unparking Detection Rate. For distances above 20 m SmartPark correctly identifies false unparking events 100% of the time. (b) CDF of the false unparking events.

Fig. 13. Detecting false unparking events: A user parks the car, goes about her business and then and gets a ride back in a second car, deliberately generating false unparking events 100% of the time. The second car is placed at various distances from her car.

6.2.6. Parking accuracy
We ask 7 volunteers to use SmartPark for a period of a few weeks; the only action asked of them is to signal on their phones whenever they park and unpark their vehicles so as to create the ground truth. Five volunteers are regular car users while the other two usually bike.

At the end of the experiment, the data showed that the volunteers signaled 27 parking and unparking events while using a car and 2 parking and unparking events while using a bicycle. SmartPark managed to correctly detect the parking and unparking 100% of the time, with zero false positives.

6.2.7. Energy consumption
Fig. 14 shows the phone battery level when a user starts with a full charge and uses SmartPark. For comparison, we also include the effect of using the GPS and Wi-Fi continuously. The data shows that SmartPark has a negligible effect on the battery and significantly smaller than using just the Wi-Fi, for example.

To elucidate why SmartPark consumes so little energy, we analyze the energy needs of its two core functionalities: detecting the transportation mode and location matching.

Transportation Mode Detection: SmartPark uses 2 s of sensor readings for detecting the transportation mode, a process triggered every 10 s when the car is parked. We measured the energy consumed by the sensors SmartPark utilizes and that required to calculate the summary statistics from this 2 s sample. We found it to be around 197 mA for the 7 smartphones used in our experiments. This represents about 1615 mJ every 10 s.
Fig. 14. After 5 h of using SmartPark, the battery level has dropped by only 4%.

*Location Check:* We measured the energy consumption of the Wi-Fi and network passive scans on the 7 smartphones used in this study and found it to be around 117 mA for a period of 1.07 s, representing about 513 mJ per location check. Note that the location checks are performed only when SmartPark detects that the user is using the same transportation mode as when she parked her vehicle.

7. Related work

SmartPark brings together ideas from different disciplines so an exhaustive state of the art is beyond the scope of this work. In the following, we describe some representative examples.

7.1. Transportation mode detection

The emergence of the smartphone as a powerful computing and sensing device has prompted a lot of work on leveraging results from decades of research in learning and data mining [26] for analyzing people's daily activities. Hemminki et al. [21] propose using the smartphone's accelerometer to detect the user's transportation mode. It uses the popular Adaboost classifier [27] to detect five transportation modes: car, bus, train, tram and metro with accuracy between 67–90%. In contrast, SmartPark delivers virtually 100% accuracy for a larger number of transportation modes, albeit using more sensors. Yu et al. [28] introduce a new classifier that can significantly reduce power consumption while still maintaining good accuracy. However, it is geared towards non-motorized activities, such as walking and running and it groups all motorized transportation under a single category, which they refer to as vehicle. GPS has been considered for detecting transportation mode [12], however, it has been shown to be inaccurate and energy hungry. [10] introduces an approach combining GPS and GIS (geographic information system) data that can detect transportation modes with accuracy reaching 93.5%, a 17% improvement over GPS-only solutions. However, this comes at the cost of needing high-granularity GIS data such as real-time information regarding bus locations, rail and bus stop information, etc. [23] and [11] combine accelerometer and GPS to achieve 94% accuracy but, just as [28], these solutions are geared towards non-motorized activities and use the umbrella “motorized” label to group all motorized activities.

7.2. Location matching

Using Wi-Fi access points as a reference for localization has become a very popular tool since first being introduced in [18]. The Euclidean distance in signal-strength space is used by RADAR [18] for matching two locations and by NearMe [29] for distance estimation. SensLoc [19] uses the Tanimoto coefficient [20] to compare signal-strength vectors for identifying places. BeaconPrint [30], also targeted at recognizing places, uses the unique identifiers of Wi-Fi access points and GSM towers to create space signatures. None of these approaches, however, was targeted at the specific needs of detecting unparking events. Most are designed for indoors where Wi-Fi signals are usually strong while those designed for outdoors are usually aimed at coarse-grained place recognition. Parksense [3], observing that Wi-Fi signals outdoors are often severely attenuated, uses the beacon reception ratio instead of signal strength to define the signature of a place. SmartPark strikes a different compromise. It still uses the signal strength because it is more closely related to distance but it gives priority to access points from which it receives the strongest signals. Furthermore, it takes advantage of cellular base stations which, unlike the Wi-Fi access points, are installed primarily for outdoor access, resulting in stronger and more stable received signals.

7.3. Real-time parking information

Most new indoor parking facilities install custom infrastructure for providing real-time parking availability. Providing the same service for on-street parking has become a major challenge for cities worldwide [6,31,32,4]. San Francisco has put
in place of one of the most sophisticated and expansive systems for street parking management [7]. At the cost of thousands of dollars per parking spot, however, it is beyond the means of many cities. A far more affordable solution can be built by taking advantage of the increasingly sophisticated sensors installed on our smartphones. ParkSense [3] is one of the first parking availability solutions that relaxes the requirement for custom infrastructure by relying exclusively on WiFi. It is a significant improvement but it can only detect if a user is walking or aboard a motorized vehicle, assuming the latter means the user is unparking. However, after a user parks her vehicle she can very well use other forms of motorized transportation, such as public buses. Furthermore, ParkSense requires that its users are always in the reception area of several WiFi access points. UPDetector [33] relaxes the assumption for always-on WiFi by relying on the smartphone sensors. However, it still detects only two states: walking and driving. SmartPark addresses this weakness by being able to detect 9 transportation modes using the smartphone sensors. CrowdPark [34] represents a different set of solutions, which rely on crowdsensing instead of sensing. To address the biggest challenge of any such system, user participation, it introduces a marketplace and shows that, by choosing payment and incentive parameters carefully, the parking service can be profitable. While CrowdPark and SmartPark follow very different approaches, they can be complementary. SmartPark is designed for a light-weight deployment and, once successful, it can be complemented with something like CrowdPark, enabling a more sophisticated interaction between parking users.

8. Conclusions and future work

We presented SmartPark, a system that relies on smartphone sensors and the ubiquitous WiFi/cellular infrastructure to provide real-time parking availability information. SmartPark addresses the challenge of detecting unparking events by solving the automatic transportation mode detection and location matching problems. The solution is based on an exhaustive statistical analysis of the sensor readings, generating 2880 features, and a Random Forest based approach for classifying the sensor readings into one of the 9 supported transportation modes in real time on a smartphone. To reduce the initial deployment risk, SmartPark introduces an analytical approach for estimating parking occupancy even when a small fraction of users adopts SmartPark. We evaluated SmartPark in simulation and the wild and showed that it can identify unparking events with 97% accuracy while triggering zero false positives.

Despite its accuracy, SmartPark has limitations, especially when considering that such applications can have financial implications. For example, a user can park her vehicle and then hand the keys to someone else to unpark it without necessarily giving them her smartphone: SmartPark will fail to detect the unparking event in this case. A potential solution is allowing a user to declare a new driver when handing over the keys and forward them the parking location profile. Experiments in Section 6.2.5 revealed another limitation: When two cars are separated by a distance of at least 20 m, SmartPark can tell them apart with 100% accuracy. However, when the distance is 10 m or less the accuracy drops to 50%. A better location matching approach is needed before SmartPark can transition to a product. A potential solution is to use the fact that more and more cars are equipped with Bluetooth, enabling a fine-grained pairing between car and driver.

To address the issues that could potentially arise from a large-scale deployment, we plan to negotiate with city halls interested in SmartPark to allow us to perform testing and data collection on voluntary basis starting with a small user base.

References