ParkGauge: Gauging the Occupancy of Parking Garages with Crowdsensed Parking Characteristics

Jim Cherian* Jun Luo* Hongliang Guo* Shen-Shyang Ho* Richard Wisbrun†
*School of Computer Engineering, Nanyang Technological University, Singapore
†Traffic Management, Traffic Assistance, BMW Group, Germany
Email: *{jcherian, junluo, guohl, ssho}@ntu.edu.sg †richard.wisbrun@bmw.de

***Abstract*—Finding available parking spaces in dense urban areas is a globally recognized issue in urban mobility. Whereas prior studies have focused on outdoor/street parking due to a common belief that parking garages are capable of delivering real-time occupancy information, we specifically target at (indoor) parking garages as this belief is far from true. This problem is very challenging as all the infrastructure supports (e.g., GPS and Wi-Fi) assumed by existing proposals are not available to parking garages, so counting how many vehicles are using a parking garage by crowdsensing can be extremely difficult.**

To this end, we present ParkGauge, a method to gauge the occupancy of parking garages, along with a reference system prototype for performance evaluation; it infers parking occupancy from crowdsensed parking characteristics instead of counting the parked vehicles. ParkGauge adopts low-power sensors (e.g., accelerometer and barometer) in the driver's smartphone to determine the driving states (e.g., turning and braking). A sequence of such states further allows the inference of driving contexts (e.g., driving, queuing and parked) that in turn yield temporal parking characteristics of a parking garage, including time-to-park and time-in-cruising/queuing. Mining such mobile data opportunistically collected from a crowd of drivers arriving at various garages yields a good measure of their occupancies and hence useful recommendations can be generated (in real-time) to inform drivers coming toward these venues. Through extensive experiments, we demonstrate that our method fully explores these parking characteristics to efficiently infer occupancies of parking garages with high accuracy.

I. Introduction

With a rapid economic growth, the vehicle population in dense urban areas has been witnessing a drastic rise around the world; this has led to a higher demand on parking infrastructure especially in mega-cities. While on-street parking has led to serious traffic congestion due to vehicles cruising in searching for parking places [1], multi-storey *car parks* (aka, *parking garages*), which are the major urban parking facilities for mega-cities, can also produce serious on-road congestion because of long queues of vehicles waiting to enter them, even though their total capacity may be sufficient to meet the parking demands. A reason for this paradox is that the occupancy information measured at the garages (if any) is mostly displayed only locally, rather than published online and available to drivers in advance. As illustrated in Figure 1, our study on the *central business district* (CBD) of Singapore during the year 2014 shows that, out of 360 garages, 28% have no occupancy monitoring system installed and 93% do not publish the real-time occupancy information online, which can cause waiting queues as shown in Figure 2.

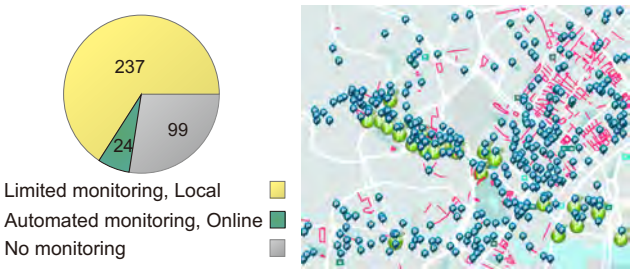

Fig. 1: Parking garages in the CBD area of Singapore, with percentages of the types of occupancy monitoring shown on the left. Garages with online occupancy information and on-street parking are marked in green and purple, respectively.

Vehicles waiting to get parked also cause environmental issues, as their emissions produced during cruising/braking result in serious air pollution [2], [3]. Many cities try to control the problem by establishing road-side parking guidance systems, adding occupancy sensor infrastructure, and even enforcing special policies (e.g., a dynamic parking pricing scheme) to discourage people from driving into a CBD, but they all need heavy investments to get fully deployed and some of them actually increase the overall carbon footprint. In fact, the majority part of the parking problem can be tackled if the occupancy information is available to drivers in advance, enabling them to drive towards less congested venues or re-plan their trips. Our studies reveal that some popular parking garages may remain full for several hours with acute queuing during peak hours, while close-by ones possess surplus, as shown by the two garages A and B shown in Figure 3.

Fig. 2: Queuing in front of parking garages: examples from Singapore on the left and Dortmund, Germany on the right.

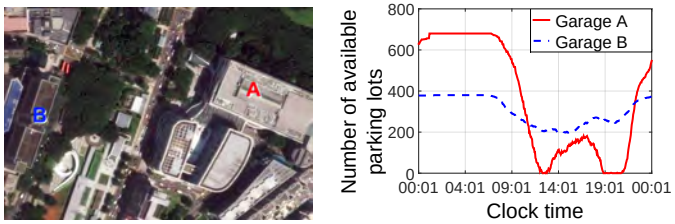

Fig. 3: Parking availabilities for two close-by parking garages A and B during a day.

Existing infrastructure-based systems deployed to collect parking occupancy data often require occupancy sensors and wireless beacons at parking lots (e.g., SFpark [4]) or extra sensors on vehicles (e.g., ParkNet [3]), making them expensive and not very scalable. Consequently, it would be very helpful, from both economic and environmental points of view, if the crowdsensing ability [5], [6] of the driving population can be exploited to tackle the parking problem in an infrastructure-free manner. Inertial sensing has been widely used for human gesture and transportation mode detection (e.g., [7], [8], [9], [10], [11], [12]) without depending on any infrastructure support, but it is extremely hard to infer occupancy information from direct inertial sensor readings: inertial sensing may well suggest driving states such as braking and turning, but such events are seemingly unrelated to parking occupancy unless certain inference mechanisms are in place to establish the necessary connections.

In this paper, we propose ParkGauge, a method to gauge the occupancies of parking garages in urban areas. In order to apply inertial sensing for this purpose, ParkGauge innovates in using (temporal sequences of) driving states (e.g., braking and turning) to infer parking characteristics (e.g., time-to-park, time-in-queuing) that are in turn exploited to deduce garage occupancies. To the best of our knowledge, ParkGauge is the first crowdsensing method [1] that uses temporal information to efficiently and effectively indicate parking occupancy for indoor parking garages. The primary contributions of this work are summarized as follows:

- We propose a fundamentally new idea to measure the occupancy of parking garages, exploiting seemingly irrelevant but readily available mobile sensing data.
- We develop a full sensing framework in order to complete the transformation from (mostly) inertial sensor readings to temporal parking characteristics.
- We identify, combine and re-engineer machine learning algorithms in a hierarchical manner, aiming to guarantee a highly effective inference process on the diverse datasets typically obtained through a crowdsensing system.
- We demonstrate the efficiency and effectiveness of ParkGauge with extensive experiments in several parking garages, using a prototype implemented on Android.

It is worth noting that, unlike existing crowdsensing-based parking systems that directly count the available parking lots, ParkGauge does not require a "crowd" (hence high penetration of the application) for sensing individual parking garages. Instead, a minimal amount of sensing data acquired from a small number of users for each parking garage would be sufficient for ParkGauge to deliver useful information, whereas the "crowd" is needed only for covering many parking garages across a large urban area.

The rest of our paper is organized as follows. We first discuss the related proposals that lead us to the design of ParkGauge in Section II, followed by an overview of the architecture and key concepts in Section III. We further present the technical details of ParkGauge methodology in Section IV and our extensive experiment results in Section V. Finally, we discuss potential extensions and conclude the paper.

II. Related Works

In this section, we motivate ParkGauge by showing the mismatches between existing literature on parking occupancy inference/activity recognition and the actual challenges of the indoor parking problem.

A. Parking Occupancy Inference

ParkNet [3] initiates the idea of crowdsensing-assisted parking availability detection, but its reliance on extra sensors (i.e., ultrasonic range finders) installed on ParkNet vehicles has tagged it with a more infrastructure-dependence nature. Instead, a few of the recent parking systems, PhonePark [14], ParkSense [15] and PocketParker [16], have taken more advantage of the driver-vehicle crowdsensing ability: they combine smartphone sensors and localization techniques to detect parking and unparking events and to infer the availability of parking spaces. ParkSense [15] relies on the ubiquitous presence of Wi-Fi beacons in urban areas to detect unparking events, while the detection procedure is initiated by a phone-based parking payment system and keeps working until an unparking event is detected. Focusing on outdoor parking areas (surface lot), PocketParker [16] requires GPS or Wi-Fi to roughly locate a parking vehicle and further applies a probabilistic inference mechanism to account for non-participating vehicles. It utilizes accelerometer to deduce transitions between driving and walking and thus to detect parking and unparking events. PhonePark [14] proposes a similar approach to build up a historical parking availability profile, while additionally exploiting the pay-by-phone parking system and bluetooth sensor to detect parking and unparking events.

Though relying on GPS, Wi-Fi, or a payment system is plausible for outdoor parking, it is often not feasible for indoor parking: while both GPS and Wi-Fi are often not accessible, a payment in garage typically happens at the departure time. Also, ParkSense [15] requires the system to keep Wi-Fi sensing between a pair of parking and unparking events, potentially increasing the overall energy consumption of the smartphone. Moreover, inferring occupancy from a subset of parked vehicles (as suggested by PocketParker [16]) may not work well in a garage, simply due to its much

[1] A poster abstract of this work [13] explores the potential of using Simple Linear Regression to infer discrete occupancy levels from time-to-park.

larger capacity than an outdoor parking area. Crowdsourcing applications are available for drivers to report the occupancy information (e.g., Google OpenSpot). Such approaches often lack proper incentives to stimulate driver participations [15], because a driver would deem it too troublesome to input data manually. In addition, we note that existing vehicle navigation systems provide a useful feature called *estimated time of arrival* (ETA) at destination. However, if a parking system only delivers the number of available parking lots, a driver has insufficient information at his disposal to plan the arrival at the final destination.

B. Activity recognition

Existing literature on human activity and transportation mode detection is bountiful [7], [8], [17], [9], [10], [11], [18]. These proposals often have a low-power profile due to the use of inertial sensing techniques. However, it is almost impossible to directly apply inertial sensing to derive occupancy information. Fortunately, recent advances in machine learning and data mining techniques allow indirect inference to be performed when proper logical correlations can be identified [19]. Previous studies [20] have also explored the feasibility of using hierarchical Bayesian non-parametric methods for mobile context discovery from raw sensor data. Therefore, it is potentially feasible to avoid the sensing-prohibitive parking lot counting and still gauge the occupancy information with seemingly irrelevant sensor readings.

III. Background / Overview

This section explains the design rationale of ParkGauge, followed by an overview of the architecture.

A. Design Rationale

Whereas counting the available parking lots is not feasible for indoor parking garages, it is not effective either because the crowdsensing application has to be adopted by most (if not all) drivers. Therefore, ParkGauge aims to gauge the occupancy evaluation from a different perspective. In particular, if we deem a parking garage as a queueing system, the occupancy determines its service rate, which is functionally related to the waiting time in the queue. In the parking context, we term this waiting time time-to-park and consider it as a temporal *parking characteristic*; it is composed of other parking characteristics such as time-in-cruising and time-in-queuing. While time-to-park is defined as the time span from a vehicle's arrival at a parking garage till it gets parked, time-in-cruising and time-in-queuing measure the time spent in looking for a free parking lot and that for waiting in a stationary state, respectively. From the data that we have collected shown in Figure 4, there exist strong correlations between occupancy and time-to-park.

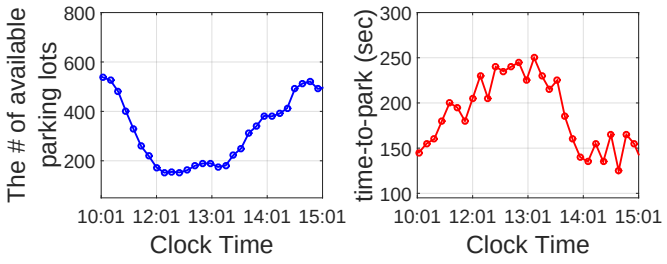

Fig. 4: Parking occupancy and time-to-park at a popular shopping mall: a negative correlation is obvious.

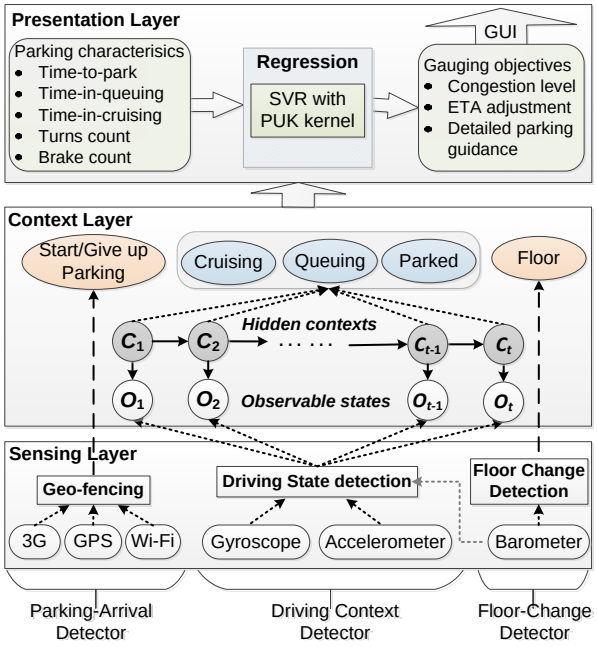

Fig. 5: Architecture of ParkGauge.

Using parking characteristics such as time-to-park to infer occupancy can also satisfy our other design objectives. First of all, these characteristics can be detected mostly by inertial sensing techniques (if properly used), making the system both autonomous and energy-efficient. Secondly, it is scalable to very large metropolitan areas: since only a few tens of samples (so as to reach certain statistic significance) need to be obtained for each parking garage, a small number of ParkGauge equipped drivers/vehicles would be able to cover a very large area. Thirdly, the data used for inferring occupancy can extend to serve other purposes, including a more accurate ETA indication. Finally, the system is self-incentivizing: if drivers wish to get the occupancy information for their respective destinations, they would enable the ParkGauge application to share their data in the first place.

B. System Architecture

ParkGauge runs as an application in a driver's smartphone that is assumed to be taken along with the driver upon completing a parking. The architecture of ParkGauge has a 3-layer presentation shown in Figure 5, so that connections between the seemingly unrelated sensing data and our gauging objectives can be made through a multi-stage inference. Sensing functions are performed at the lowest layer (the Sensing Layer), where ParkGauge collects data using several sensors embedded in the smartphone, including mainly gyroscope,

accelerometer, and barometer for continuous sensing in a parking garage, as well as 3G, GPS, and Wi-Fi only for determining the starting context of a parking process. Features extracted from the sensor data collected within a sliding window are fed into a Random Forest-based classifier that estimates the *driving state*. The outputs of this layer are the *driving state*, such as accelerating, braking and turning.

The Context Layer applies a *Hidden Markov Model* (HMM) to represent the relation between the input (*driving state*) and the output (*driving context*), where *driving state*s are treated as observable states and the parking-relevant *driving context*s are the hidden ones to be inferred. As the HMM factors in the temporal correlations between consecutive driving contexts, it may potentially counteract the errors in classifying the driving states. Note that a few driving contexts require direct sensor inputs, resulting in direct connections that bypass the HMM.

At the Presentation Layer, parking characteristics are derived from driving contexts. For example, the time span between start-parking and parked is counted as time-to-park, whereas a brake count counts the number of brakings. Some of these characteristics can be used directly (e.g., to improve ETA estimations), but inferring the occupancy of a parking garage requires an independent learning module based on regression. Using the historical data, we build a regression model to represent the relation between occupancy and the derived parking characteristics, so that ParkGauge can infer the occupancy in real-time. To perform regression on our diverse crowdsensed data, we employ Support Vector Regression (SVR) with a universal kernel function based on Pearson VII function (PUK) [21]. As all the learning models are trained offline, the online inferring process run by ParkGauge client involves only low complexity arithmetic operations, making it both time and energy efficient to be hosted in commercially-available smartphones.

IV. Methodology

We explain the key aspects of our methods in this section. We first present two direct-sensing components of ParkGauge, namely parking-arrival and floor-change detectors. Subsequently, we focus on the driving context detection and explain how we learn the HMM model from observed sensor data and infer the driving contexts. We finally discuss the novel aspects on utilizing the various parking characteristics to infer occupancy of a parking garage.

A. Parking-Arrival Detector (PAD)

As the contexts to set and reset a gauging process, start-parking and give-up-parking need to be captured in order for ParkGauge to decide whether to start or to end. ParkGauge relies on a low-energy 3G-based location detector to roughly indicate if the geo-fence (e.g., a 500m circle around a parking garage near the destination is entered. If true, ParkGauge starts the GPS-based localization to capture either start-parking or give-up-parking context. If the vehicle further approaches the garage within a small circle (e.g. 50-75m, empirically very close to the entry point), the start-parking context is signaled and ParkGauge starts its gauging process. The current time is recorded as the candidate $Parking\ Arrival\ Time$, and the Driving Context Detector (DCD) is invoked. Note that the thresholds (50-75m, 500m, 60s) are configurable for individual locations or garages.

Under a normal situation, the false-positive rate of detecting a start-parking is low, as the chance of a vehicle slowly approaching a parking garage but bearing no intention of entry is very small. However, we have observed a few vehicles driving near a parking garage but hit by a traffic jam, causing a false alarm. Fortunately, ParkGauge is robust to such anomalies, as the recorded $Parking\ Arrival\ Time$ is only a candidate; it can be canceled if a give-up-parking is signaled, either due to the vehicle exiting the geo-fence or as a result of the driving context signaled by DCD (*c.f.* Sec. IV-C).

B. Floor-Change Detector (FCD)

This detector primarily operates on the barometer sensor data to indicate relative height variations and thereby detects floor changes during a vehicle's parking process. Detecting floor-change context allows ParkGauge to reinforce its belief in an ongoing parking process, and it also facilitates deriving parking characteristics for individual floors within a parking garage (e.g., time-in-cruising for basement 2), so that ParkGauge can provide fine-grained parking characteristics desirable for users who, for example, prefer to park at a floor close to their favorite shops.[2]

While the gradual increase in barometer reading is the consequence of the vehicle's cruising along a spiral ramp, the overall variations clearly indicates the floor changes: the altitude variation is inversely proportional to that of the pressure with a slope of roughly 0.12mp/m. The patterns are consistent for each parking garage and that helps ParkGauge offer repeatable floor detection performance even in the presence of long-term pressure fluctuations [22]. ParkGauge obtains barometer readings at 1Hz frequency and it applies a 10s window to check if the readings are sufficiently stable, i.e., variance below a threshold (e.g., 0.1mb). Whenever stable readings are identified, their average value is then compared with that of the previous one. If the resulting altitude difference falls within a pre-determined height interval applicable for the parking garage structure (normally around 3.3m) or its multiples, a floor-change context is signaled.

C. Driving Context Detector (DCD)

DCD is the core of ParkGauge for inferring the driving contexts of a vehicle. We first briefly introduce the basic models. Then we explain the feature selection and driving state classification at the sensing layer. Finally, we present the learning process for the HMM (context layer).

[2]Such profile information can be of interests to other mobile applications such as advertisement pushing or coupon distribution.

1) Modeling a Parking Process: We use O and C to represent *observation* (or driving state[3]) and driving context respectively. The time-granularity of inference is quantified using a window of size W. We use the subscript t to denote the time index of the state and context for a window. Except for the 3 direct-sensed contexts discussed in Sec. IV-A and IV-B, the relation between observation (driving state) and driving contexts are modeled as an HMM with standard first-order Markov assumptions made for the temporal sequence of driving contexts. This can be specified by: (1) stationary context distribution, $\Pr(C_1)$, (2) context transition model, $\Pr(C_t|C_{t-1})$, and (3) observation model, $\Pr(O_t|C_t)$.

When a stream of driving states $[O^1, O^2, \cdots]$ are estimated by the sensing layer, ParkGauge takes them for inferring the t-th driving context C_t, where the correlation between consecutive contexts has been implicitly considered by involving the previous context as an input. Table I lists all the driving contexts and observations considered by ParkGauge.

TABLE I: List of driving states and contexts.

Driving state	Driving context
a. proceeding	**i.** driving
b. accelerating	**ii.** cruising
c. decelerating	**iii.** parked
d. turning	**iv.** queuing
e. braking	**v.** start-parking
f. pausing	**vi.** give-up-parking
g. walking	**vii.** floor-change

2) Feature Extraction and Classification of Driving States: As shown in Figure 5, we mainly use the accelerometer and gyroscope for driving context detection. Whereas barometer may be used for this purpose with even lower energy consumption [23], our current implementation avoids using it due to its longer detection latency. Next, we describe how features suitable for ParkGauge are extracted and used to train a classifier that estimates driving states.

ParkGauge uses a sampling frequency of 20Hz to collect sensor readings through Android SensorManager API. The driving state is determined by collecting these readings in an interval of W seconds and feeding it to a classifier. By default, $W = 10$s and we justify this choice through experiments in Sec. V-D. Raw acceleration values are pre-processed by applying a low-pass filter to estimate the gravity component that is removed to obtain linear acceleration values. These values are further smoothed (to mitigate noise) by applying a 5-term moving average, and the results are processed to extract the time-domain and frequency-domain features mentioned in Table II. Similarly, raw gyroscope readings are also smoothed and processed to extract the specific angular time-domain features listed. Figure 6 shows examples of raw sensing readings corresponding to the driving states considered by ParkGauge. Clearly, the aforementioned features characterize these states sufficiently.

[3]We shall hereafter use these two terms in an interchangeable manner.

TABLE II: Features used in classifying driving state.

Acceleration-based		Gyroscope-based
Time-domain	Frequency-Domain	Time-domain
Mean	Peak Frequency	Mean
Variance	FFT-Peak	Variance
Range	FFT-0,1,2,3,4Hz	Range
Gradient	Energy	Peak Count

Using a dataset obtained from our experiments (*c.f.* Sec. V-A), we train a Random Forest based classifier [24] offline that enables ParkGauge to efficiently and accurately estimate the driving state from sensor data readings at runtime. The training dataset consists of the aforementioned features extracted for a sliding window of size W and 50% overlap. Each data record is labeled with corresponding actual driving states. The Random Forest algorithm fits randomly selected samples from the labeled training data to multiple decision trees using the principle of bootstrap-aggregating. It further predicts the class labels for unseen data by taking a majority vote among the trained decision trees and thereby offers better generalization performance over a single decision tree. The ParkGauge implementation incorporates a rather modest set of 10 trees with a maximum depth of 3 each.

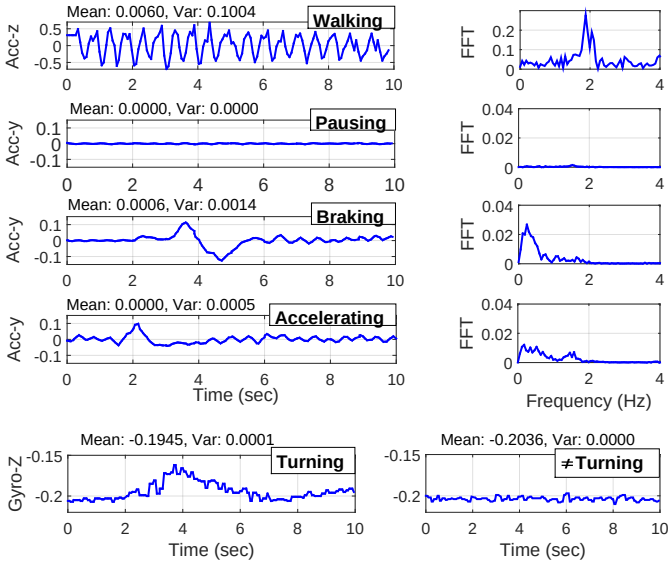

Fig. 6: Visualizing driving states by raw data.

3) Learning and Using the HMM Parameters: Given the observations made online, the HMM shown in Figure 5 allows us to infer the (hidden) driving contexts, if the HMM parameters discussed in Section IV-C1 are fully specified. We briefly explain how we learn the HMM parameters (offline) based on the driving/sensing data collected in advance, as well as how to use them for online inference. For learning the HMM model, we apply the backward-forward recursion similar to Baum-Welch EM algorithm [25], but we opt for the Gibbs Sampling [19] technique for a more robust estimation: it makes better use of our dataset with driving states manually labeled and corresponding observations recorded.

Given the dataset obtained through our intensive tests, we apply common statistics to derive initial values for the model parameters defined in Sec. IV-C1. For example, the stationary probability of $\Pr(C_1 = \text{cruising})$ is obtained by looking at the frequency of the cruising context out of all observed driving contexts. Now given a sequence $\{o_1, o_2, \cdots, o_T\}$ of observed driving states, the forward-backward procedure is used to compute the *forward posterior probability* and the *backward posterior probability*. The model is iteratively updated based on these probabilities and repeated until convergence to stable estimations of the model. To apply this model for inferring the driving contexts online, we make use of a modified Viterbi algorithm [26]; it identifies the most likely sequence of hidden driving contexts from a sequence of observations (driving states). At runtime, whenever a change of context is signaled, the timestamp is also recorded. In particular, the *Parked Time* is recorded if the context cruising changes to parked.

D. Gauging Occupancy using Parking Characteristics

Given the context changing times recorded, we can compute the parking characteristics. While time-to-park is the difference between *Parked Time* and *Parking Arrival Time*, time-in-cruising and time-in-queuing simply accumulate the time spent in the corresponding contexts. In the following, we present how these characteristics can be used to gauge the parking occupancy.

1) Gauging the Occupancy: As explained in Section III-A, there exists an implicit functional relationship between time-to-park and occupancy. Instead of deriving this function analytically, we use regression to fit this function based on our labeled datasets (*c.f.* Section V-F). As a result, whenever ParkGauge obtains an estimated time-to-park, it can infer the corresponding occupancy. Such occupancy estimations can be averaged over a sliding time interval (for example, 20 minutes duration in our reference implementation) and reported to users in real-time. Figure 7 illustrates this relation using real-world data collected from a popular parking garage during a working day (10:00 to 16:00) with at least 1 observation recorded every 10 minutes. It can be observed that lower time-to-park is observed during periods of high parking space availability and vice-versa. When such high-quality data are available, a simple linear or quadratic regression fits a model that can operate with reasonably low estimation errors.

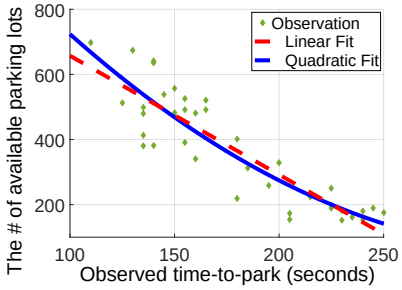

Fig. 7: Training the regression model to fit the relation between occupancy and time-to-park.

TABLE III: Features used to infer parking occupancy.

Parking characteristics: input features for regression		
time-to-park	day of week	turn count
time-in-queuing	arrival time	brake count
time-in-cruising		

However, in a practical mobile crowdsensing system, it may not be feasible to obtain high quality data for every parking garage and this may limit the quality of the parking occupancy inference. To address this issue, we propose to use additional parking characteristics that are obtained from the lower layers of ParkGauge and use them to improve the inference accuracy for garages having a lower number of observations per day (due to a lower penetration of ParkGauge users visiting the garage). Specifically, we propose to exploit the ability of ParkGauge to extract the features (regressor variables) in Table III for each parking session and utilize them offline to train a robust regression model.

In order to choose a robust regression model using these features, we propose to use the well-known Support Vector Regression (SVR) with an appropriate kernel function selected through experimental evaluation (see Section V-F).

2) Support Vector Regression: There has been extensive literature on Support Vector Regression [27] and hence we only briefly describe the relevant concepts here. Given a training data set of n observations $D = (x_i, y_i), i = 1, 2, ..., n$ with $x \in \mathbb{R}^d$ representing the d input features obtained from each parking session and $y \in \mathbb{R}$ representing the corresponding output parking occupancy to be inferred, SVR tries to find the function $f(x)\colon \mathbb{R}^d \rightarrow \mathbb{R}$ mapping the input training data features x to output y, that produces estimation of y which is the closest to the corresponding true values in D.

The problem belongs to Reproducing Kernel Hilbert Spaces (RKHS) $\mathcal{H}$ and can be formulated as

$$f^*(x) = \arg\min_{f \in \mathcal{H}} \left(C \sum_{i=1}^{l} V(f(x_i), y_i) + 0.5||f||^2_{\mathcal{H}} \right) \quad (1)$$

where C is a parameter controlling the relative weight of the loss function and the regularization penalty $||f||^2_{\mathcal{H}}$. The loss function in our case is $V(f(x), y) = |round(f(x)) - y|$.

3) Choosing Kernel Function: Apart from SVR, kernel-based methods have become highly popular in solving regression and classification problems and several kernel functions have been proposed. Kernel functions help to transform the input data space into a Hilbert Space. (i.e. a space which is spanned by distance-based functions or inner-products of the input feature data vectors $\in \mathbb{R}^d$) Some of the popular kernel functions are Linear kernel, Polynomial kernel and Radial basis function (RBF). Recently, [21] proposed as a new alternative, the Pearson VII Universal kernel (PUK) function which is basically the Pearson VII function developed in 1895 by Karl Pearson [28]. PUK can be regarded as a universal kernel function since it can handle linear, polynomial and RBF

based feature mappings. PUK function possesses the flexibility to vary between Gaussian and Lorentzian shapes and more, thereby offering more robust mapping capability and enabling it to deal with a broad spectrum of data and problems. Since PUK can help avoid the tedious process of selecting kernel functions, the model building process can be made simpler and computationally efficient. The Pearson VII function can be understood in the general form

$$\frac{H}{\left[1+\left(\frac{2(x-x_0)\sqrt{2^{(1/\omega)}-1)}}{\sigma}\right)^2\right]^{\omega}}. \quad (2)$$

where x is the regressor variable and H represents the maximum height of the peak of x (i.e., the value of y observed at its center x_0). The half-width and tailing factor of the peak are controlled by the parameters σ and ω. The function is flexible to vary from Gaussian ($\omega \approx \infty$) towards a Lorentzian ($\omega \approx 1$) shape by varying the parameter ω. This property enables PUK to fit peaks of a variety of line widths and shapes, thus helping to serve as a universal kernel function.

V. Empirical Evaluation

In this section, we describe the evaluation procedure, setup for data collection and parking experiments using our ParkGauge prototype, as well as evaluation results.

A. Data Collection and Experiment Setup

For an effective evaluation, we have implemented ParkGauge as an Android-based application along with a back-end server. While the model learning procedure is done offline in the server using the collected dataset, all the online detections are performed by the ParkGauge client application running (at the background) in a smartphone, which periodically reports data to the server.

Our experiments involved five different phones: Samsung Galaxy S4, HTC One M8, LG Google Nexus 5, Samsung Galaxy S3, and HTC One X. All these phones possess both accelerometer and gyroscope, while the first three of them also have a barometer. The last two phones were chosen to test how the ParkGauge system would work without the barometer. To validate the inter-floor height calculated by the barometer, we used a Bosch laser rangefinder DLE40 to obtain the ground truth. We collected data for two months in two different cities (Singapore and London), during which about 120 traces were gathered, including 62 self-parking sessions. Six drivers of different nationalities and diverse driving habits contributed to this dataset and both petrol (Toyota WISH, VW Polo, Mazda 3) and electric (BMW i3) cars were used.

The parking garages in our experiments are specifically selected to be attached to popular shopping malls, so that we do get significant fluctuations in their occupancies during a day. These garages published their occupancies online, which we collect and use as ground truth for our inference. We performed two categories of designed experiments: i) parking among a few close-by garages in a sequence, and ii) parking in a single garage in a cyclical manner, i.e. enter, park, walk, unpark, exit, and repeat. On one hand, the former helps us to evaluate the robustness of ParkGauge in accommodating different parking situations: these garages were quite different in terms of number of floors (ranging from 3 to 12 floors), underground or aboveground, floor area and capacity per floor etc. On the other hand, the latter category enables us to evaluate the consistency of ParkGauge and also to verify the characteristics measured by ParkGauge against the ground truth during the same period of time. Each parking session took roughly 10 to 25 minutes, depending on the occupancy at that time. To capture the fluctuations in the demand for parking, the data collection was performed around the peak hours, i.e., 10:00-16:00 and 18:00-20:00.

B. Data Pre-Processing

We work with 2 kinds of data: (1) sensor data collected from smartphones during parking sessions, (2) parking characteristics determined by ParkGauge users.

1) Pre-processing Raw Sensor Data: The sensor data obtained from the different phones are cleaned and smoothened to remove noise and outliers, as already described in Section IV-C2. The cleaned data is used to extract features and train the driving state classifier, and subsequently, the HMM driving contexts. 10-fold cross-validation is performed to ensure good generalization performance of these methods.

2) Pre-processing the Parking Characteristics Dataset: This dataset included both (a) a feature-rich set of parking characteristics computed by the ParkGauge clients and validated against corresponding ground truth, as well as (b) (availability, time-to-park) data pairs collected at few garages through manual observation. The former dataset is richer in features and very diverse since they are from different days and involving different drivers, albeit the number of observations per driver or vehicle and per garage and per day are small. However, since this is crowdsensed and inferred data, there are some chances of errors. The latter data is collected for a specific garage for a long duration (e.g. 10:00-16:00) and recorded at least once in every 10 minutes, and this resulted in data of high quality and reliability, and nearly identically and independently distributed (i.i.d). This dataset is very important since it is used to construct a regression model and infer the parking occupancy. Therefore, the dataset is first cleaned to remove extreme values or outliers. We first calculate the Inter-Quartile Ranges (IQR) on all records and then filter and eliminate the values in 25% and 75% quartiles that exceed the IQR. Subsequently, we normalize all the numeric features in all the data records using min-max normalization process before applying the regression process.

a) Feature Selection: After pre-processing, we apply a feature selection algorithm to identify a significant feature subset i.e., the best features useful for inferring parking occupancy. Although this could be performed by a filter or wrapper method, we employ Correlation Feature Selection (CFS), a

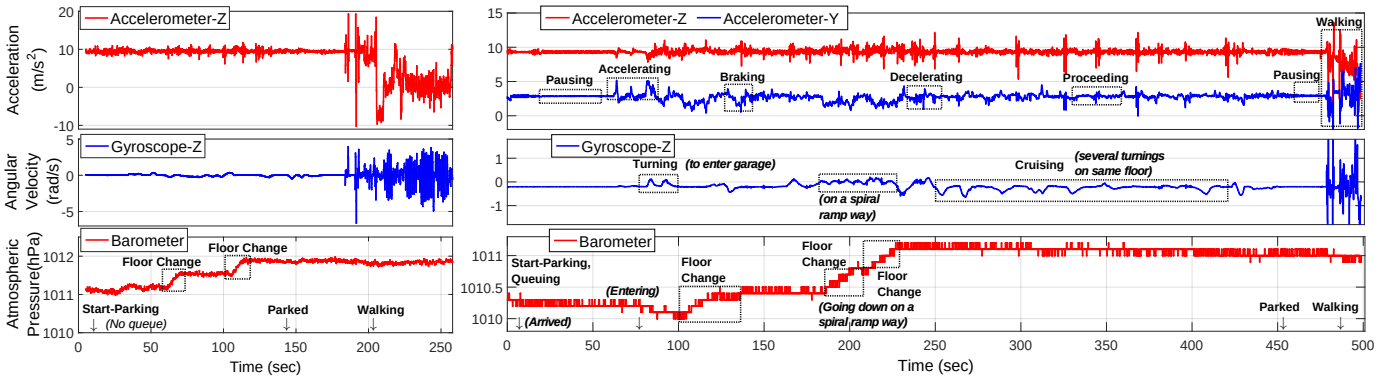

Fig. 8: Raw data traces corresponding to two parking trials with short and long time-to-park. We only provide full labels to the long trace for better readability, but the feature-context correspondence can be easily discerned for the other trace also.

correlation-based feature subset selection method for machine learning [29]. This method evaluates the inference quality of a subset of attributes by evaluating the individual predictive ability of each feature in addition to the degree of redundancy between these features. Running the CFS algorithm on our training dataset with 10-fold cross validation helped to arrive at the 7 features listed in Table III. In our original dataset, additional fine-grained feature attributes were available such as the number of floors traversed, left turns count, and right turns count. However, these attributes were excluded since they did not possess significant predictive ability within our dataset. With larger datasets arriving from a large-scale crowdsensing deployment, such feature selection is useful.

b) Splitting The Dataset for Training and Testing: After pre-processing, we evaluate several regression methods. For each method, we train the model using our training dataset and perform 10-fold cross validation to understand their generalization performance. Finally, we test the trained model with a testing dataset and report the performance.

For this paper, we evaluated the methods with data for different days from the same driver and different drivers. Finally, we intentionally mixed the ParkGauge-reported data from actual parking sessions with the manually observed dataset, since this approximates the data inputs with a real-world crowdsensing system. This dataset consists of 78 records, of which 37 records are manual observations with a high fidelity and the rest are crowdsensed by ParkGauge during parking experiments. The manual observations had empty features which were marked as missing values, but valid features were still considered. This dataset was further divided into training data and testing data. We performed both 10-fold and leave-one-out cross validation approaches to ensure generalization performance. The testing data was obtained by randomly sampling 10 observations without replacement, resulting in a 10-record testing dataset and a 68-record training dataset.

C. Inferring Driving States and Contexts from Sensor Data

To provide a rough idea about how ParkGauge operates in field, we use two sets of raw sensor readings plotted in Figure 8 to demonstrate the inferred driving states and contexts. These two parking trials differ in their values of time-to-park reported by ParkGauge: (1) The left one took only 140 seconds as the vehicle arrived at a parking garage with a rather low occupancy (in the morning), and most of the time was spent mainly on cruising and going two-levels downwards (indicated clearly by barometer readings). (2) The right case was a long parking trial. The vehicle arrived at the parking garage with almost full occupancy, so it was first hit by a long queue at the entrance, then it spent quite a long cruising time in finding a parking lot even after getting three levels down. This trial took place during the peak hour when people drive to the shopping mall to get their lunch.

D. Accuracy of Driving Context Inference

We omit the performance evaluation for driving state inference and directly evaluate the accuracy of estimating driving contexts. This is because driving contexts are directly related to ParkGauge's final performance in deriving parking characteristics and thus gauging the occupancy. To evaluate the accuracy of estimating a certain driving context, we first identify the time intervals spent in that context according to our ground truth observation. Then, we check the output of ParkGauge to see the *percentage* of these time intervals when ParkGauge has made a correct estimation. Assuming an ergodic stochastic process behind the estimation, this percentage exactly indicates the success rate in detecting this driving context. Due to space limit, we choose to show evaluation results for three contexts, namely driving, parked, and cruising.

The other goal of this evaluation is to choose the best window size W used for estimating driving state and context. In fact, we initialize W to 5s, as this is the smallest value to accommodate a certain driving state (e.g., turning). Then, we try different values of W that are all multiples of 5s. As shown in Figure 9, while it is expected that the latency grows almost linearly with W, the highest accuracy is achieved at $W = 10$s. The accuracy first increases with W as ParkGauge gets more information to make a decision, but it starts to decrease later as the "cost" of making a wrong estimation grows with W. As a result, we fix $W = 10$s in our final implementation.

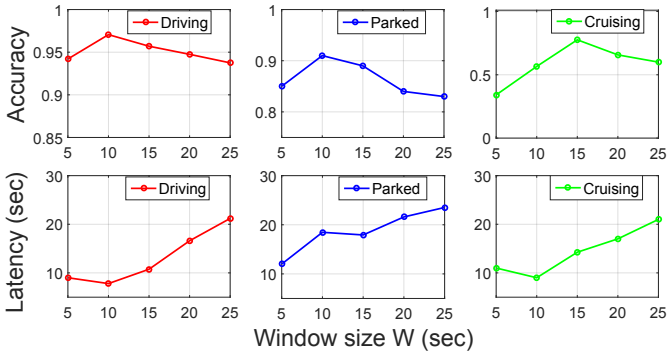

Fig. 9: Detection accuracy and latency for 3 driving contexts (namely driving, parked, and cruising) under various window sizes W with a stepsize of 5s.

E. Accuracy of time-to-park Estimation

Estimating time-to-park is an important functionality of ParkGauge. As shown in Figure 10, the estimation errors range from 5s to around 30s. While extreme values are rare, most

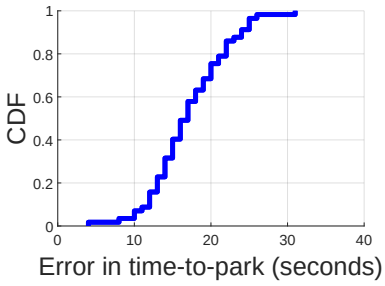

Fig. 10: CDF for the estimation errors of time-to-park

errors are around 18s. Given that the normal values of time-to-park are in the order of minutes (or tens of minutes in the worst case), we deem ParkGauge's performance in estimating time-to-park very accurate. In fact, as the error mean is related to the window size W (it is systematically biased due to the detection latency), we could further reduce the estimation error by subtracting W from each estimated time-to-park. This would bring the error down to the range of $(-5, 20)$s.

F. Performance in Occupancy Gauging

As explained in Section IV-D1, we use regression to recover the functional relationship between occupancy and temporal parking characteristics time-to-park extracted from the data. To perform the evaluation, we use the training and test datasets described in Section V-B2b. We compare four regression approaches, namely: (1) Simple Linear Regression (SLR) with only time-to-park feature, (2) Multiple Linear Regression (MLR) with all 7 features listed in Table III, (3) Artificial Neural Network (ANN) using 10 hidden layers, and (4) SVR with 5 different kernel functions, namely Linear, Polynomial (quadratic, cubic), RBF and PUK.

To compare the different regression methods, we use the Normalized Root-Mean Squared Error,

$$NRMSE = \sqrt{\frac{\sum_{i=0}^{n}(p_i - a_i)^2}{n}}, \quad (3)$$

the square root of the mean of squares of errors between the normalized actual values $\{a_i\}$ (as discussed in Section V-B2) and the corresponding inferred values $\{p_i\}$.

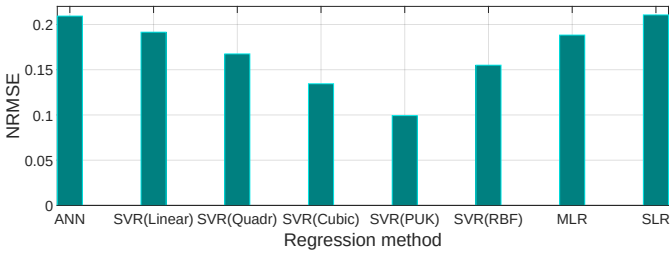

Fig. 11: Comparison of different regression methods

From Figure 11, one observes that linear regression approaches (MLR and SLR) performed reasonably well. The NRMSE is high, but it is still comparable with SVR. ANN does not perform very well with this dataset. Using the hidden layers, it seems to over-fit the model to the training data and hence the test data results in higher errors. However, SVR provides good performance with different kernel functions. Linear kernel offers slightly worser error performance, whereas Polynomial (both Quadratic and Cubic) and RBF kernel functions offer superior performance. But, it is clear that SVR with PUK kernel offers the lowest NRMSE (0.0993) and hence, the best generalization performance.

A key observation from this evaluation is that SLR performs decently (NRMSE 0.2106) even when using time-to-park as the only input feature. However, the accuracy obtained is more than doubled (NRMSE 0.0993) when we perform SVR with PUK kernel on a higher-dimensional training data with additional features such as time-in-queuing as listed under Section IV-D1. This justifies the multi-layered hierarchical methodology adopted by ParkGauge.

G. Tuning The Parameters of SVR(PUK)

Since we have convincing results that help us choose SVR(PUK) as the preferred approach to infer parking occupancy, we now tune the parameters of the PUK kernel function, namely ω and σ, that control its shape and thereby affects the inference output of SVR(PUK). We vary each parameter across a range of 0 to 100 while keeping the other parameter fixed at a value 1. In each iteration, we train a SVR(PUK) model and check its generalization performance using 10-fold cross validation. In Figure 12 (left plot), one observes that the lowest NRMSE is obtained at ω = 2 when σ is fixed at 1. In the right plot, one observes that the best performance (0.0993) is obtained at σ = 1 when ω is fixed at 1. Thus, we choose the common point σ = 1 and ω = 2 as the Pearson parameters for SVR(PUK) in inferring parking occupancy for ParkGauge.

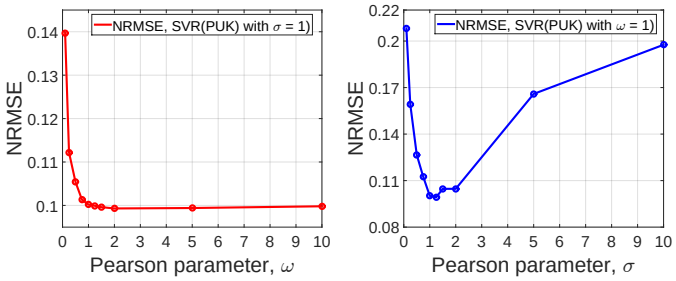

Fig. 12: Determining the values of Pearson parameters ω and σ that yield the best performance for SVR(PUK)

VI. Conclusion and Discussions

Based on the observation that indoor parking garages may lack real-time occupancy information, we set out to develop a new mobile parking data collection, processing, and inference method, ParkGauge; it delivers real-time occupancy information to its users. As indoor parking garages differ in many aspects from outdoor parking places, the method adopted by existing proposals for directly counting occupied parking lots are infeasible. To this end, our design of ParkGauge innovates in using inertial sensing data to infer parking occupancy with reasonably low errors. Inspired by queueing theory, ParkGauge computes parking characteristics such as the time spent to park a vehicle and the time it spends in queuing and cruising, the brake count and number of turns made during the parking process, and uses SVR to infer parking occupancy.

In future, we plan to make a large-scale deployment of ParkGauge and analyze the crowdsensed parking characteristics inferred by ParkGauge. For a real deployment of mobile crowdsensing system to be successful, energy management is also crucial. For this purpose, the application can adopt a strategy utilizing operating modes such as a very-low-power idle mode, a short but energy-hungry prepare mode utilizing high-power sensors such as GPS only when necessary to detect start/give-up-parking, and a low-power active mode utilizing only low-power sensors to compute the parking characteristics.

VII. Acknowledgments

This work is supported by the BMW Group, Germany.

References

[1] D. C. Shoup, "Cruising for Parking," *Elsevier Transport Policy*, vol. 13, no. 6, pp. 479–486, 2006.

[2] R. Arnott and E. Inci, "An Integrated Model of Downtown Parking and Traffic Congestion," *Elsevier Journal of Urban Economics*, vol. 60, no. 3, pp. 418–442, 2006.

[3] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe, "ParkNet: Drive-by Sensing of Road-Side Parking Statistics," in *Proc. of the 8th ACM MobiSys*, 2010, pp. 123–136.

[4] D. Simons, "SFpark: San Francisco Knows How to Park It," *ITDP Sustainable Transport*, no. 23, 2012.

[5] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A Survey of Mobile Phone Sensing," *IEEE Comm.*, vol. 48, no. 9, pp. 140–150, 2010.

[6] R. K. Ganti, F. Ye, and H. Lei, "Mobile Crowdsensing: Current State and Future Challenges," *IEEE Comm.*, vol. 49, no. 11, pp. 32–39, 2011.

[7] Y. Wang, J. Lin, M. Annavaram, Q. A. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh, "A Framework of Energy Efficient Mobile Sensing for Automatic User State Recognition," in *Proc. of the 7th ACM MobiSys*, 2009, pp. 179–192.

[8] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using Mobile Phones to Determine Transportation Modes," *ACM Trans. on Sensor Networks*, vol. 6, no. 2, p. 13, 2010.

[9] M. Keally, G. Zhou, G. Xing, J. Wu, and A. Pyles, "PBN: Towards Practical Activity Recognition Using Smartphone-based Body Sensor Networks," in *Proc. of the 9th ACM SenSys*, 2011, pp. 246–259.

[10] Y. Wang, J. Yang, H. Liu, Y. Chen, M. Gruteser, and R. P. Martin, "Sensing Vehicle Dynamics for Determining Driver Phone Use," in *Proc. of the 11th ACM MobiSys*. ACM, 2013, pp. 41–54.

[11] S. Hemminki, P. Nurmi, and S. Tarkoma, "Accelerometer-based Transportation Mode Detection on Smartphones," in *Proc. of the 11th ACM SenSys*, 2013, p. 13.

[12] H. Aly, A. Basalamah, and M. Youssef, "LaneQuest: An Accurate and Energy-Efficient Lane Detection System," in *Proc. of the 13th IEEE PerCom*, 2015, pp. 163–171.

[13] J. Cherian, J. Luo, H. Guo, S.-S. Ho, and R. Wisbrun, "Poster: Parkgauge: Gauging the congestion level of parking garages with crowdsensed parking characteristics," in *Proc. of the 13th ACM SenSys*. ACM, 2015, pp. 395–396.

[14] B. Xu, O. Wolfson, J. Yang, L. Stenneth, P. S. Yu, and P. C. Nelson, "Real-Time Street Parking Availability Estimation," in *Proc. of the 14th IEEE MDM*, 2013, pp. 16–25.

[15] S. Nawaz, C. Efstratiou, and C. Mascolo, "ParkSense: A Smartphone Based Sensing System for On-Street Parking," in *Proc. of the 19th ACM MobiCom*, 2013, pp. 75–86.

[16] A. Nandugudi, T. Ki, C. Nuessle, and G. Challen, "PocketParker: Pocketsourcing Parking Lot Availability," in *Proc. of the 14th ACM UbiComp*, 2014, pp. 963–973.

[17] J. Yang, S. Sidhom, G. Chandrasekaran, T. Vu, H. Liu, N. Cecan, Y. Chen, M. Gruteser, and R. P. Martin, "Detecting Driver Phone Use Leveraging Car Speakers," in *Proc. of the 17th ACM MobiCom*, 2011, pp. 97–108.

[18] A. Bloch, R. Erdin, S. Meyer, T. Keller, and A. De Spindler, "Battery-Efficient Transportation Mode Detection on Mobile Devices," in *Proc. of the 15th IEEE MDM*, 2015, pp. 185–190.

[19] S. L. Scott, "Bayesian Methods for Hidden Markov Models: Recursive Computing in the 21st Century," *Journal of the American Statistical Association*, vol. 97, no. 457, pp. 337–351, 2002.

[20] J. Zheng, S. Liu, and L. M. Ni, "Effective Mobile Context Pattern Discovery via Adapted Hierarchical Dirichlet Processes," in *Proc. of the 15th IEEE MDM*, 2014, pp. 146–155.

[21] B. Üstün, W. J. Melssen, and L. M. Buydens, "Facilitating The Application of Support Vector Regression by Using A Universal Pearson VII Function Based Kernel," *Chemometrics and Intelligent Laboratory Systems*, vol. 81, no. 1, pp. 29–40, 2006.

[22] K. Muralidharan, A. J. Khan, A. Misra, R. K. Balan, and S. Agarwal, "Barometric Phone Sensors: More Hype Than Hope!" in *Proc. of the 15th ACM HotMobile*, 2014, p. 12.

[23] K. Sankaran, M. Zhu, X. F. Guo, A. L. Ananda, M. C. Chan, and L.-S. Peh, "Using Mobile Phone Barometer for Low-Power Transportation Context Detection," in *Proc. of 12th ACM SenSys*, 2014, pp. 191–205.

[24] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[25] F. Jelinek, L. Bahl, and R. Mercer, "Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech," *IEEE Trans. on Information Theory*, vol. 21, no. 3, pp. 250–256, 1975.

[26] R. Shinghal and G. T. Toussaint, "Experiments in Text Recognition with the Modified Viterbi Algorithm," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 1, no. 2, pp. 184–193, 1979.

[27] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press, 2002.

[28] K. Pearson, "Contributions to The Mathematical Theory of Evolution. II. Skew Variation in Homogeneous Material," *Philosophical Trans. of the Royal Society of London. A*, pp. 343–414, 1895.

[29] M. A. Hall and L. A. Smith, "Practical Feature Subset Selection for Machine Learning," in *Proc. of the 21th ACSC*, 1998, pp. 181–191.