Your Search Path Tells Others Where to Park: Towards Fine-Grained Parking Availability Crowdsourcing Using Parking Decision Models

RUILIN LIU and YU YANG, Rutgers University, USA
DAEHAN KWAK, Kean University, USA
DESHENG ZHANG, LIVIU IFTODE, and BADRI NATH, Rutgers University, USA

A main challenge faced by the state-of-the-art parking sensing systems is to infer the state of the spots not covered by participants' parking/unparking events (called background availability) when the system penetration rate is limited. In this paper, we tackle this problem by exploring an empirical phenomenon that ignoring a spot along a driver's parking search trajectory is likely due to the unavailability. However, complications caused by drivers' preferences, e.g. ignoring the spots too far from the driver's destination, have to be addressed based on human parking decisions. We build a model based on a dataset of more than 55,000 real parking decisions to predict the probability that a driver would take a spot, assuming the spot is available. Then, we present a crowdsourcing system, called ParkScan, which leverages the learned parking decision model in collaboration with the hidden Markov model to estimate background parking spot availability. We evaluated ParkScan with real-world data from both off-street scenarios (i.e., two public parking lots) and an on-street parking scenario (i.e., 35 urban blocks in Seattle). Both of the experiments showed that with a 5% penetration rate, ParkScan reduces over 12.9% of availability estimation errors for all the spots during parking peak hours, compared to the baseline using only the historical data. Also, even with a single participant driver, ParkScan cuts off at least 15% of the estimation errors for the spots along the driver's parking search trajectory.

CCS Concepts: • **Information systems** → **Location based services**; • **Human-centered computing** → **Ubiquitous and mobile computing**;

Additional Key Words and Phrases: Mobile Sensing, Crowdsourcing, Parking, Human Decision Modeling

ACM Reference Format:
Ruilin Liu, Yu Yang, Daehan Kwak, Desheng Zhang, Liviu Iftode, and Badri Nath. 2017. Your Search Path Tells Others Where to Park: Towards Fine-Grained Parking Availability Crowdsourcing Using Parking Decision Models. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3, Article 78 (September 2017), 27 pages.
DOI: http://doi.org/10.1145/3130942

1 INTRODUCTION

Parking availability sensing is an important topic in the context of Smart City. According to a recent survey [38], cars seeking for parking on average spend 3.5 to 14 minutes on cruising before finding an

Authors' addresses: R. Liu, Y. Yang, D. Zhang, L. Iftode, and B. Nath, Computer Science Department, Rutgers University, 110 Frelinghuysen Road, Piscataway, NJ 08854, USA; D. Kwak, School of Computer Science, Kean University, 1000 Morris Avenue, Union, NJ 07083, USA; email: rl475@cs.rutgers.edu, yy388@cs.rutgers.edu, dkwak@kean.edu, desheng.zhang@cs.rutgers.edu, iftode@cs.rutgers.edu, badri@cs.rutgers.edu.

2474-9567/2017/9-ART78 $15.00
DOI: http://doi.org/10.1145/3130942

available spot in downtown areas, which account for 30% of street traffic. Moreover, many cities are experimenting dynamic parking price policies, which are based on real-time parking availability [31, 48]. To date, most parking sensing systems are focused on detecting where and when a parking or unparking event happens using various sensors. Since the early 2000s, researchers have been using dedicated sensors (e.g. infrared sensor at each spot) [12, 32] or human reports [2] to detect the state of monitored spots. More recently, sensors in smartphones were explored to identify the transition between driving and other mobility patterns to infer parking/unparking events of participant drivers [22, 29, 44].

However, most existing sensing systems are not designed to efficiently estimate the availability of background spots, i.e., those not covered by dedicated sensors or by the parking/unparking events of participant drivers. These background spots, due to low penetration rates of existing parking sensing systems, account for the vast majority of urban parking resources. Even worse, in many application scenarios, we need the fine-grained availability estimation at the spot level. For instance, when looking for a parking spot in a busy airport parking lot, a driver needs to know which spots are more likely to be available. Existing crowdsourcing approaches extrapolate statistics generated from sensed spots to the overall parking availability using the random sampling theory [32]. Since the sampling theory is primarily suitable for a large population random process, the current method to estimate background parking availabilities is intrinsically inefficient for fine-grained background parking availability estimations.

To address this issue, we design ParkScan, a crowdsourcing system estimating the availability of the parking spots along drivers' parking search trajectories. The key idea of ParkScan is based on the empirical observation that passing by a parking spot that is of interest to the driver's destination implies that the spot is unavailable. For instance, if a driver cruises around her/his workplace for a long time and parks at a spot far away, we can infer that most spots, if not all, along the cruising path should be unavailable. Otherwise, the driver would take one of them. Given that people's travel destinations for daily commutes can be highly predictable along with the ubiquity of trajectory data, ParkScan leverages the participating drivers as human sensors to scan the spots along their parking search trajectories and then infer the spots that have possibly been taken. Since a driver's search trajectory usually covers many spots [38], ParkScan has a great potential to improve the parking sensing coverage with little cost incurred, compared to solutions based on dedicated sensors. Moreover, as we use the driver's search trajectory to directly estimate the parking availability along the trajectory, we can overcome the drawbacks of extrapolating participant parking events to global parking statistics using only sampling theory. As a result, ParkScan tolerates a much lower penetration rate, e.g. even with a single driver.

The key technical challenge we address in ParkScan is how to model drivers' general parking preferences in heterogeneous parking facilities and to merge the estimates from different parking search trips at different times to build the best estimation. In particular, the contributions of this paper are as follows:

- To study drivers' parking decisions, we collect a large-scale dataset that consists of more than 8,000 vehicle trajectories in the parking lot and 55,000 parking decision events, i.e, either taking or ignoring an available spot during a parking search process. With the features extracted from this valuable dataset, we build a data-driven model that reflects driver's underlying decision strategy. To the best of our knowledge, this is the first dataset at scale that records real driver's parking decisions with rich features. Based on this dataset, our model is the first data-driven parking decision model generated from large-scale real-world parking decision observations. The sample dataset is shared with the research community in [20].
- We design a crowdsourcing system called ParkScan to complement the state-of-the-art parking crowdsourcing systems particularly for fine-grained background parking availability estimations. ParkScan takes parking searching trajectories from participant vehicles as input. By combining

with historical knowledge and static parking facility layouts, ParkScan leverages a learned parking decision model to generate probabilistic parking availability estimations at the parking spot level.
- We evaluate ParkScan in both off-street scenarios (i.e., two public parking lots) and a city-scale on-street parking scenario. In the parking lot experiments, we utilize over 6,000 human parking search trajectories and real-world parking availability to conduct the evaluation. In the city-scale street parking scenario, we conduct data-driven evaluations using over 63,000 parking requests generated from real parking transactions in an urban area covering 35 blocks in Seattle, WA. Both of the experiments showed that with a 5% penetration rate, ParkScan reduces at least 12.9% of availability estimation errors for all the spots in the experimental scenario during the peak parking hours, compared to a state-of-the-art solution based on historical data. More importantly, even with a single participant driver, ParkScan cuts off 15% of estimation errors for the parking spots along drivers' parking search trajectories.

The rest of this paper is organized as follows: Section 2 provides the main idea of the ParkScan system. Section 3 presents the architecture of ParkScan and discusses the technical challenges. Section 4 elaborates on our parking search dataset and how we build our data-driven parking decision model. Section 5 discusses in detail how to use the pre-trained parking decision model to estimate spot availability and how to merge estimations of different qualities into one availability probability estimation. Sections 6 and 7 present the evaluation experiments and results for off-street parking and on-street parking scenarios. Finally, we discuss the limitations and some insights in Section 8 and related work in Section 9, followed by the conclusion of the paper in Section 10.

2 MOTIVATION

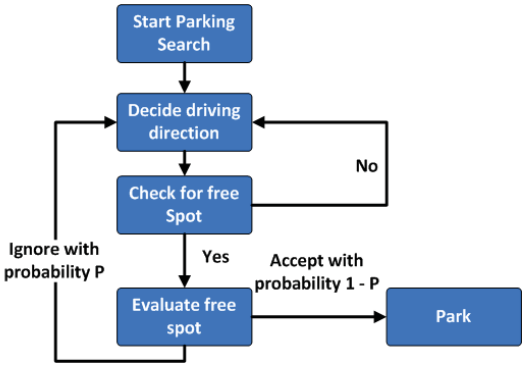

Fig. 1. Parking search process

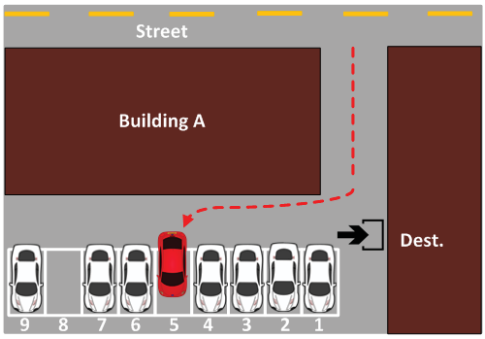

Fig. 2. ParkScan use case

To motivate the design of ParkScan, we analyze the human behavior during the parking search process, i.e., the process from a driver starts looking for parking until the driver parks the car.

As shown in Fig. 1, a typical parking search process can be modeled as a sequence of decisions. During a parking search trip π, a driver u needs to make routing decisions at each intersection. More frequently, u faces a decision moment when u decides whether to take an available spot i. Based on the state-of-the-art work [3, 25, 40, 45], this decision process is probabilistic and determined by the driver's **parking decision model** where with the probability $p_{u,\pi,i}(\neg park \,|\, empty)$, u chooses to ignore the spot i and continues to search for the next available spot; with the probability of $1 - p_{u,\pi,i}(\neg park \,|\, empty)$, u chooses to take the spot i for parking.

An example of this parking process is illustrated in Fig. 2. A driver who is looking for a parking spot in a parking lot visited several spots, i.e., from spot 1 to spot 4. But since these spots were already taken, the driver continues to search until finding an available spot, i.e., spot 5. The driver evaluated this spot 5 by implicitly computing a probability, $p_{u,\pi,i}(\neg park \,|\, empty)$ based on the distance to his/her trip destination, i.e., the building on the right side of the figure. In this case, the probability $p_{u,\pi,i}(\neg park \,|\, empty)$ for ignoring spot 5 is not quite high, so the driver selects spot 5 and parks there.

Given the driver's parking search trajectory in Fig. 2 (the dotted line) and the driver's final destination, intuitively, we can easily find that spot 1 to spot 4 are "better" than spot 5 for this driver. Since the driver has by-passed these spots and parked at a suboptimal spot, we can infer that spot 1 to spot 4 were taken when the driver visited them. During this intuitive process, this inference is actually a Bayesian modeling process defined by the following equation:

$$p_{u,\pi,i}(empty \,|\, \neg park) = \frac{p_{u,\pi,i}(\neg park \,|\, empty) \times p_i(empty)}{p_{u,\pi,i}(\neg park)}, \tag{1}$$

where $p_{u,\pi,i}(\neg park \,|\, empty)$ is the probability for a driver u to ignore spot i when i is available, $p_i(empty)$ is the probability that spot i is available, $p_{u,\pi,i}(\neg park)$ is the probability that u does not park in spot i for trip π. Note that, only $p_{u,\pi,i}(\neg park \,|\, empty)$ is estimated using an observer's parking decision model, so both $p_i(empty)$ and $p_{u,\pi,i}(\neg park)$ are prior knowledge that can be learned using historical data.

The above discussion essentially suggests that if we are given a parking search trajectory, it implies that all spots except for the last one along the trajectory are ignored by the driver. If we can estimate $p_{u,\pi,i}(\neg park \,|\, empty)$ by feeding necessary features into a machine learning model, the observation that the driver u does not choose to take spot i leads to an estimation of the posterior probability that spot i is available. Since this process combines the historical knowledge and real-time observations, it could potentially provide a better availability estimation for the background spots along a driver's parking search trajectory. For instance, when a spot i is extremely attractive for driver u's trip π, $p_{u,\pi,i}(\neg park \,|\, empty)$ will approach zero. In this case, when the driver u ignores this attractive spot, we will accordingly have $p_{u,\pi,i}(empty \,|\, \neg park)$ to be very small, indicating that spot i is not likely to be available.

3 PARKSCAN OVERVIEW

Based on the discussion in Section. 2, we present an overview of the ParkScan crowdsourcing system.

3.1 Key Idea

When a participating driver starts a searching process for a parking spot, the key objective of ParkScan is to provide parking availability estimations of nearby parking spots to facilitate this search process, e.g. by influencing their search routing, with a frontend app and a backend cloud server. The key feature of ParkScan is its transparency to its background participating drivers whose searching processes contribute to parking availability estimations for drivers using ParkScan for parking. While a driver is looking for a parking spot near her/his final destination, our ParkScan frontend app will show locations of potential parking spots with different probabilities indicating parking availability estimations to the driver. Based on these availability estimations, the driver will select a route to find an available parking spot with a high probability. Then, the key question becomes how to obtain parking availability estimations.

In this paper, we utilize a driver's search trajectory along with his/her final destination to infer and update parking availability estimations for the parking spots along the search trajectory. As follows, we introduce these three components, respectively. (i) Search Trajectories: While showing parking availability estimations to the driver, the ParkScan app simultaneously tracks his/her search trajectory. After a parking event is detected for this vehicle, e.g., when a transition from an in-vehicle mode to an on-foot

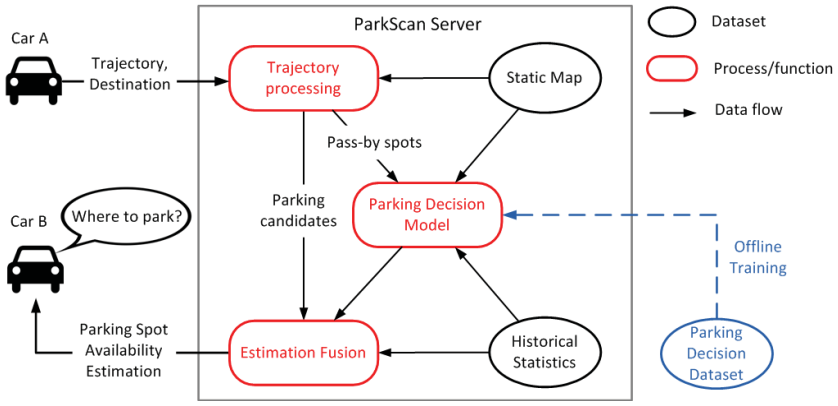

Fig. 3. ParkScan system architecture

mode is detected by Google activity recognition API [9], the search trajectory of this parking event near the final parking spot will be recorded by the ParkScan app. (ii) Final Destinations: Driver's final destinations are important for ParkScan to improve its performance. Some drivers may have their final destinations input in the navigation system, which can be utilized by our ParkScan app. In such case, when the driver occupies a parking spot and starts walking to his/her final destination, the ParkScan app will upload the recorded trajectory and the final destination to a backend ParkScan cloud server. However, other drivers may drive without using a navigation system. In this case, our ParkScan system can estimate the drivers' final destinations by using locations of the first building that drivers entered [28]. (iii) Parking Availability Estimations: Upon receiving a search trajectory (ending with the parked position) together with a final destination, the ParkScan cloud server extracts a few features needed for a pre-built parking decision model. For each of the spots covered by a search trajectory, ParkScan estimates its posterior availability probability (details are in Section 2) to update parking estimations. Finally, these updated parking estimations are then utilized by subsequent drivers during their parking search processes.

In this paper, we focus on the key idea of ParkScan, which is to utilize search trajectories and final destinations to estimate parking availabilities of parking spots because parking/unparking detections and trajectory tracking techniques have been extensively studied, e.g. [22, 28, 39, 47].

3.2 Architecture

Based on the previous discussion, we present the architecture of ParkScan in Fig. 3. The parking decision model is the key component used in the ParkScan server, which is trained offline using a large-scale parking decision dataset. However, as this model describes the parking decision strategy of generic drivers, it does not have to be re-trained for each individual deployment. To justify this design choice, we present a default model and test it using different settings in Section 6 and Section 7.

For online parking availability estimations, ParkScan uses both real-time data (i.e., parking search trajectory and drivers destination estimated from the ParkScan app), and off-line data (i.e., historical parking statistics and stationary parking facility map) as input. When the real-time input data of a driver are received, ParkScan first processes the trajectory and generates two types of parking spots: (i) parking spot candidates (the spots that have been potentially taken by this driver based on the ending position of the search trajectory) and (ii) pass-by spots (the rest spots along the search trajectory).

Since locations reported in the trajectory, especially the ending point, are noisy, we rectify the trajectory and estimate the localization errors by using static map of the parking facility. The parking spot candidates

define probabilistic observations to fill those spots. For the pass-by spots, we generate the features by using both a static parking layout and historical data, and then feed them into the parking decision model to compute the probabilistic posterior availability, i.e., using Equation 1. Since the estimates from a single parking candidate or pass-by spot observation are error-prone, we apply an estimation fusion process to combine the estimates from heterogeneous resources at different moments with the historical data to build the latest parking spot availability estimation. This process generates the final results for participant drivers and other potential applications, e.g. a routing service.

3.3 Challenges

In order to build ParkScan, there are two key challenges to address. This subsection elaborates on these challenges and our main idea to address them.

3.3.1 Modeling Parking Decision with Accessible Scenario-independent Features. The first challenge faced in ParkScan is building a generic model that predicts the likelihood that a driver would take spot i when it is available, i.e., $1 - p_{u,\pi,i}\,(\neg park \,|\, empty)$. As prior parking behavioral studies showed that people apply similar parking search strategies in various environments [6, 10], we applied a data-driven approach to build the model from a dataset of extensive real parking trajectories and driver's parking decisions from surveillance videos of a public parking lot. More importantly, the proposed parking decision model uses only features independent of individual parking facilities and obtainable from search trajectories, static data resources, and historical data. In this way, the model is robust for different scenarios, and does not have to be re-trained for every new deployment. We will explain parking decision modeling in more detail in Section 4.

3.3.2 Generating and Fusing Heterogeneous Estimations. Assuming the parking decision model has been built, we need to extract the spot visits along the search trajectories and infer the parked spots according to the reported noisy parked position. For the spot visited along the search trajectory, we need to generate the features in realtime so that Equation 1 can be leveraged to get the spot availability estimates. Note that, the individual estimates from the parked position or pass-by spot along the search trajectory may be inaccurate due to the GPS noise or only unpredictable causes, e.g. the driver might just neglect an available spot. Therefore, after we get estimates from heterogeneous resources, i.e., parked position, pass-by visits from different vehicles at various times and historical data, we need to fuse them together to build the latest estimates when a driver requests for the estimation map. Moreover, all of these considerations have to work well when the penetration rate of ParkScan is low. To solve this problem, we discuss the realtime process in the ParkScan server and present a framework based on hidden Markov model for estimation fusion (details are shown in Section 5).

4 MODEL PARKING DECISION

In this section, we present our parking decision model based on large-scale surveillance video data we collected in a parking lot. Compared to GPS-based data, surveillance video data provide detailed information about trajectories of all vehicles and empirical parking decisions without installing GPS devices or smartphones in each individual vehicle. Note that surveillance video data are only used to train an offline model beforehand, and ParkScan does not require surveillance video data to function in real time.

This section first starts by describing our method to obtain the parking decision dataset from surveillance videos of a public parking lot, then explains how we extract scenario-independent and accessible features from these videos, and finally builds our generic parking decision model.

4.1 Video Dataset Generation

Our data source is a public parking lot on a university campus, where we placed a surveillance camera to video record the parking lot from an approximately 45-degree birds-eye view. We produced a parking decision dataset by extracting realistic parking events along with driver's final parking decisions from the video. The parking lot has 192 available spots including 12 spots for handicap parking, and drivers usually use this parking lot when they visit two buildings located at two exits shown in Fig. 4.

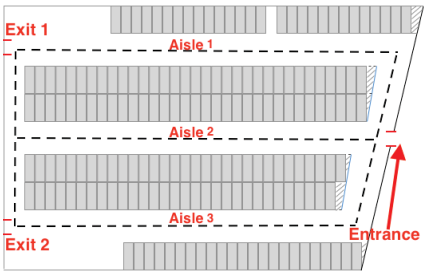

Fig. 4. Layout of the data source lot

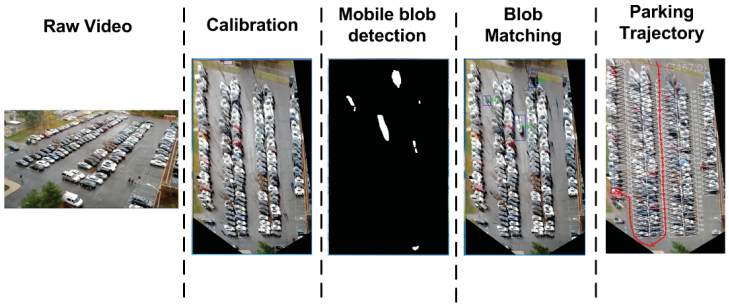

Fig. 5. Video processing

4.1.1 Vehicle Trajectory. We take three steps to process the raw video data to generate the parking trajectories as shown in Fig. 5. (i) Calibration: The calibration step builds a translation matrix from a video frame to the map of the physical world by using affine transformation. (ii) Mobile Blob Detection: Each video frame is fed into a Pixel Based Adaptive Segmenter [11] to detect moving objects and remove shadows using LR Textures based detection [34]. (iii) Blob Matching: Finally, we track the moving blobs and differentiate vehicles from pedestrians using the model presented in [35]. After these steps, we recover the trajectory of a vehicle in the parking lot.

4.1.2 Spot State Ground-truth and Parking Decision. The position where the parking trajectory terminates defines the taken spot. Similarly, the position where the leaving trajectory starts defines the leaving spot. The accumulation of the taken spot and leaving spots generates the ground-truth of the parking lot availability state at any given time. Using the parking trajectory collected in Section 4.1.1, we set up a distance threshold to determine spot visibility of all drivers. If a visible spot is available, it defines a parking decision event for a timestamped location. The driver's choice for each parking decision event is generated such that only the last visit to its taken spot is a "take" decision and all the rest are labeled as "ignore".

4.1.3 Final Destination Extraction. Since the drivers in the experimental parking lot have only two final destination options (i.e., two buildings next to the exits), we build a binary classifier to infer their final destinations using turning directions, when they are searching for parking spots, e.g. the direction of a drivers first turn after entering the parking lot shows a very strong correlation to his/her final destination. Our cross-validation results show that this classifier predicts drivers' destinations with 92% accuracy. Therefore, we use the prediction result as the ground-truth destination for each parking event.

4.1.4 Summary. From the video records in the workdays between Oct. 30, 2015 and Nov. 25, 2015, we extracted 8,181 vehicle trajectories, among which 5,988 trajectories are parking trajectories. In total, over 55,000 parking events are generated, with 5,754 decisions labeled as "take" and the rest labeled as

"ignore". Fig. 6 depicts the cumulative distribution function (CDF) of the parking availability based on the collected spot availability ground truth. Fig. 7 illustrates the CDF of the number of visited spots per parking trajectory and that of visited available spots per trajectory. Fig. 6 highlighted the needs for estimating spot-level parking availability, because during some time, e.g. 8:00 or 10: 00, the CDF grows smoothly, meaning there is much variance among different spots' availability rates. Based on Fig. 7, the mean of the number of all visited spots and that of visited available spots are 47.3 and 14.7, respectively. The large mean of the number of all visited spots implies that the posterior parking availability estimation has a higher potential to obtain high coverage. The high mean of the number of all visited available spots, on the other hand, suggests the requirement of an efficient parking decision model, because we need to filter out the influence of many spots that are ignored by drivers but in fact available.

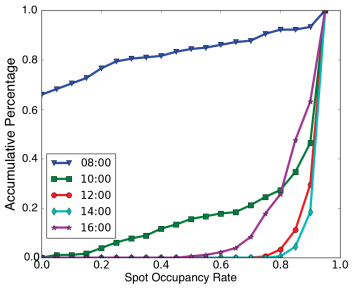

Fig. 6. CDF of parking spot occupancy rate at different times of day

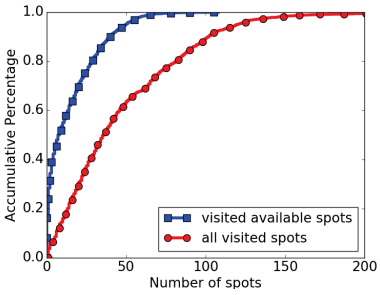

Fig. 7. CDF of the number of visited (available) spots per trajectory

4.2 Feature Engineering

After the training dataset is obtained, including historical parking availability statistics, parking trajectories, final destinations of drivers, and parking decisions, we use this information to define the features that can be used in a parking decision model and accessible in real time for ParkScan.

4.2.1 Spot related features. Since a parking decision model essentially reflects human's evaluation on the properness of a spot for the trip destination, we define a set of features to evaluate a particular spot.

Percent of better spots (PBS): This feature depicts the rank of the current spot s_c among all spots S in the parking lot, according to their total travel cost $cost_{total}$, i.e., the driving time to a spot plus the walking time from that spot to the driver's final destination. We defined a "superior" spot set S_s for the current spot s_c, which includes all the spots whose total travel time is shorter than s_c. As a result, the PBS for a current spot s_c is defined as $|S_s(s_c)|/|S|$, where $|.|$ denotes the size of a set.

Expected cost saving (ECS): ECS is designed to complement PBS to describe how much cost the spots in $S_s(s_c)$ is expected to save for a driver including both the magnitude of the saving for each spot and the available probability for that spot. ECS is defined as:

$$ECS(s_c) = \frac{\sum_{s \in S_s(s_c)} p(s) * (cost_{total}(s_c) - cost_{total}(s))}{|S_s(s_c)|}, \tag{2}$$

where $p(s)$ is the historical probability estimation that a spot s is available at present. ECS is designed for the people who would optimistically seek to minimize their total travel costs.

Expected cost overhead (ECO): In contrast to ECS, we compute the ECO for those who value the risk of missing the current spot as:

$$ECO(s_c) = \frac{\sum_{s \in S - S_s(s_c)} p(s) * (cost_{total}(s) - cost_{total}(s_c))}{|S - S_s(s_c)|}. \quad (3)$$

Conceptually, the larger the ECO value, the more likely a driver would choose to park at the current spot.

Historical availability (HA): It is well believed that the parking availability influences people's parking decision [4, 7, 10, 17, 43]. For instance, when the parking condition is critical, people tend to take a spot even if the spot is not that "good". As we do not have access to people's real visibility, we use HA for a spot i at the same time of day as the decision moment as one feature dimension for the parking decision event for that spot i.

Added walking distance (WD): Intuitively, the further a spot is from the driver's final destination, the less likely a driver would park at that spot. Since we essentially need to choose a spot from a consideration set, e.g. a parking lot, we normalize WD by subtracting the minimum walking distance of any parking spot in the consideration set from the actual walking distance of the current spot.

Driving direction (DD): This is represented as the angle between the current driving direction and the straight line segment connecting driver's current position and the final destination. Ideally, when a driver is heading to the final destination, she/he should have a lower probability to take a spot compared to when she/he is moving further from the destination.

4.2.2 Trip-related features. In addition to the spot related features, there are some trip related features that also influence driver's parking decisions.

Parking search time (PST): This feature is calculated by subtracting the search start time from the current time. Ideally, the larger the value, the more likely a driver should choose to park, once he/she finds an available spot.

Number of visited spot (NVS): This is the total number of visited spots during a driver's parking search trajectory. This feature overlaps with parking search time since a longer search time generally means more search spots. However, we found within a same search duration, exploring more spots reflects a driver's risk-taking trend, therefore it might predict a lower probability if he/she finds an available spot.

4.2.3 Summary. All spot related features defined above can be built, given historical availabilities, the parking facility layout, the driver's destination, the current spot's position and the driving direction (binary along with the street or aisle in a parking lot). Trip-related features can also be easily obtained in a parking lot scenario because a driver's parking search trip starts when the vehicle enters the parking lot. The determination of the starting point of an on-street parking searching trajectory will be discussed in Section 5. Note that, we articulately normalized most features for individual parking facilities instead of using absolute costs, so that the model is generic for different parking facilities without having the model re-trained for every deployment.

We ran the logistic regression and based on the absolute value of features' coefficients, we get the rank list of importance for the features: HA > WD > ECO > PST > NVS > DD > PBS > ECS. More specifically, the first three features in the rank list have dominated weights than the rest. Since HA, WD, and ECO are fully defined by the driver's final destination, the parking facility historical availability, and the parking facility layout, we expect that the driver's parking decision can be highly predictable, which eventually leads to good parking availability estimations using equation 1.

Table 1. Model training with different feature sets

	All Time (42202 decision events)		Busy Hours (3953 decision events)	
Model	Precision	Recall	Precision	Recall
Estimation Model	0.734	0.852	0.937	0.856
Simulation Model	0.738	0.854	0.944	0.869

Note: Busy Hours is defined from 10:30 am to 3:30 pm for cross-validation. Precision is defined as the number of correctly predicted "take" decisions / the number of predicted "take" decisions. Recall is defined as the number of correctly predicted "take" decisions / the number of all ground truth "take" decisions.

4.3 Model Training

To guarantee the quality of the training data, we filtered the events whose features may not be calculated incorrectly, e.g. some trajectories are detected to have started from a non-entrance position. With the features generated in Section 4.2, we trained a boosting tree model over the data after filtering, because it is more robust to heterogeneous features. We tested the performance of our model over all experimental time and the interval only includes busy hours, and then we compare it with a model that uses more features, which are not accessible by ParkScan in real time but are believed to improve the accuracy of parking decision models. For example, they include the availability of adjacent parking spots, the actual spot states along driver's search trajectory, and comparison between the current spot and the best spot the driver has seen so far and some properties associated with the spot, e.g., obstacles or shadows.

The results of the 10-fold cross-validation results are shown in Table 1. The model used in ParkScan to estimate posterior parking availabilities is called the estimation model. The model with additional features is labeled as the simulation model, because this model simulates driver's actual decisions by including the features that only onboard drivers can see but ParkScan cannot access.

From Table 1, we obtain three key findings as follows. (i) The estimation model does not show a largely degraded performance, even though the model only includes the features generated using historical data, map data, and driver's search trajectories and destination. This observation essentially suggests that estimating parking availabilities based on the estimation model is feasible. (ii) The parking decision model shows better performance during busy hours, which implies that ParkScan should have better performance in busy parking time, i.e., exactly when drivers need parking availability estimation the most. (iii) Either in busy hours or free hours, there are about 15% of driver's parking decisions that cannot be predicted (suggested by the recall). While these false negative parking decisions might come from the potential errors contained in our dataset, it motivates us to combine the estimates from different parking search trips so that we can mitigate the errors from a single observation.

5 REAL-TIME PARKING AVAILABILITY ESTIMATION

Based on the parking decision model constructed from human parking decision observations, this section discusses how to use noisy parking search trajectories to generate probabilistic parking availability estimation and how to fuse estimations from heterogeneous sources to mitigate potential errors in the estimations from individual search trajectories.

To simplify the discussion, for a parking lot scenario, we envision the start of a driver's search trajectory (used in Section 5.1) is the entrance of the lot and a driver's consideration set includes (needed in Section 5.2) all spots of the parking lot. Then, we present the estimation fusion framework in Section 5.3. Finally, we discuss how to extend our parking lot scenario to a street parking scenario in Section 5.4, i.e., the choice of the start of the search trajectory and all candidate spots for street parking.

5.1 Trajectory Preprocessing

After receiving the parking search trajectory, ParkScan applies the map matching algorithm [30] to translate the trajectory to a list of points along road segments or aisles in the parking lot and the paths connecting the points. Since parking spots are along the streets or aisles, all spots on the generated path are considered visited by the given search trajectory. Note that, even though each individual point in the search trajectory may not be accurate, the path and visited spots generated using map matching are accurate enough for ParkScan, except for a handful number of spots at the end of the search trajectory. In ParkScan, we need to identify these spots because they are the candidates where the car actually parked (called parking candidates). In the current ParkScan implementation, we set up a round area centered at the trajectory ending position and consider all spots within that area parking candidates.

5.2 Estimating Posterior Availability for Visited Spots

For each of the pass-by spots (i.e., all the visited spots except for the parking candidates), the parking search trajectory defines a decision event when the car passes by the spot and the human's decision is to ignore the spot.

Therefore, ParkScan computes the features in the way used in Section 4.2 and then leverages the learned parking decision model to compute the likelihood of $p_{u,\pi,i}\,(\neg park \,|\, empty)$. However, the difficulty of using Equation 1 is to estimate $p_{u,\pi,i}\,(\neg park)$, the probability that a driver u does not park in the spot i for the trip π. This probability needs to be estimated per driver per timestamped destination, requiring extensive fine-grained historical data. We decompose $p_{u,\pi,i}\,(\neg park)$ using the total probability theorem:

$$p_{u,\pi,i}\,(\neg park) = p_{u,\pi,i}\,(\neg park \,|\, empty) \;\times\; p_i\,(empty) + p_{u,\pi,i}\,(\neg park \,|\, full) \;\times\; p_i\,(full). \tag{4}$$

Since $p_{u,\pi,i}\,(\neg park \,|\, full)$ is 1 for everybody (i.e., nobody can park at a spot that is taken), by substituting Equation 4 into Equation 1, we obtain the following formula:

$$p_{u,\pi,i}\,(empty \,|\, \neg park) \quad = \quad \frac{p_{u,\pi,i}\,(\neg park \,|\, empty) \;\times\; p_i(empty)}{p_{u,\pi,i}\,(\neg park \,|\, empty) \;\times\; p_i(empty) + p_i(full)} \tag{5}$$

where $p_i\,(empty)$ and $p_i\,(full)$ are driver/trip invariant and can be estimated easily from the historical availability data. Once the likelihood $p_{u,\pi,i}\,(park \,|\, empty)$ is generated by the parking decision model, based on the equation 5 we can compute the posterior availability probability $p_{u,\pi,i}\,(empty \,|\, \neg park)$ for a visited spot that is not taken.

The parking candidates, instead of reflecting the state when the car passes by, define possible transitions from empty to full state, since there is a new car parked in that area. We use the prior knowledge that GPS errors are in general Gaussian noise [30]. Therefore, we distribute the transition probability 1 (for one newly parked car) to all parking candidates proportional to $exp(-d^2/\sigma^2)$, where d is the distance from the spot to the reported parked position, σ can be estimated using the method provided in [30].

5.3 Estimation Fusion

As discussed previously, the parking search trajectories provide the transition probability and availability estimates at different times of day, which individually may contain considerable noises. In this section, we fuse estimations for the same spot together to obtain more robust estimations.

We regard the state estimation for a spot i as a hidden Markov process (HMM) shown in Fig. 8. We partition the time into a sequence of intervals, within which the state of the spot will remain unchanged, e.g. 30 seconds. This is because parking a car or leaving from a spot usually takes from seconds to minutes. We define the true state of the spot in the interval t as the hidden variable S_t, which is either full or

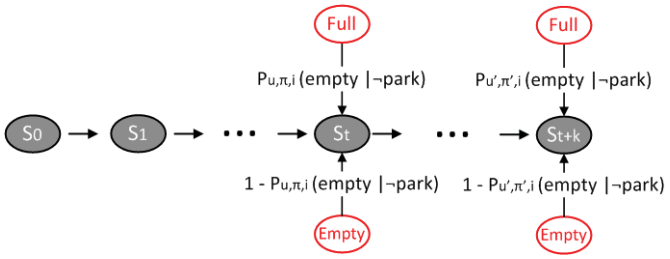

Fig. 8. Hidden Markov process for probabilistic estimation updates

empty. At some intervals, e.g., interval t and interval $t+k$ in Fig. 8, parking search trajectories from participant drivers may cover this spot and generate two state probability estimates, $p_{u,\pi,i}\,(empty\,|\,\neg\,park)$ and $p_{u',\pi',i}\,(empty\,|\,\neg\,park)$. Due to parking or vacating events of both participant and non-participant drivers, the true state may change with some transition probability, e.g. change from empty to full by a parking event. Since the transition probability can be achieved from driver's parking candidates or historical parking/vacating rates, the HMM process targets to find the best combination of states, whose accumulative probability has the highest value among all possible accumulation paths including transition probability and state probability. In our case, we only focus on the latest state of a spot, i.e., full or empty, so we normalize the full and empty probabilities at the last step, according to the sum of all probability accumulation paths ending with the current full or empty states.

Finally, the remaining problem is to estimate the state transition probabilities between two consecutive intervals. From the historical data, we can obtain the historical transition probability, e.g. parking probability can be computed by dividing the total number of parking events by the total length of an interval. Since the interval time is short, the historical transition probability will be very small. Therefore, when there is a transition probability generated by a parking spot in an interval, this transition probability will be used to update the historical value and dominate the transition probability during that interval.

Note that, the estimation fusion process, i.e., HMM process, does not necessarily end at an interval when a pass-by event or transition observation from a participant driver is detected. In most of the time, HMM changes the state estimation of a spot using the historical transition probability, which in fact fades the previously estimated state and gradually makes it converge to the historical availability rate. In this sense, the ParkScan estimation fusion algorithm naturally solves the spot availability dynamism under a limited system penetration rate.

5.4 Street Parking Extension

The discussion in the previous section is based on two assumptions: (i) the start of the searching trajectory is known; (ii) the consideration set, i.e., candidate spots that a driver would like to consider for parking, are also known. These two assumptions are easy to validate for a parking lot scenario: the entrance is the start and all spots in this parking lot are included in the consideration set. However, it is not straightforward to obtain such information for a street parking scenario.

In this work, ParkScan utilizes the intuition that when a driver approaching its destination from distance, the likelihood $p_{u,\pi,i}\,(\neg\,park\,|\,empty)$ gradually decreases, because closer spots to the final destination provide a shorter walking time. Based on this intuition, the start of the search trajectory can be inferred using the first point along the driving path where $p_{u,\pi,i}\,(\neg\,\hat{park}\,|\,empty)$ goes above a pre-defined threshold. Note that the start of the search trajectory does not need to be very accurate, because $p_{u,\pi,i}\,(\neg\,\hat{park}\,|\,empty)$ for those spots are usually approaching 1, providing no more information than the historical data. Moreover, the features closely related to the start of the search trajectory, e.g.

parking searching time, are not the most impactful ones in the parking decision model according to the discussion in Section 4.2. For the consideration set, if we have enough historical data, we find the maximum distance from the spots ever taken for the same destination and then use that as a radius to consider candidate spots; if not, we use the distance from the start of the trajectory estimated above to infer the candidate spots.

6 PARKING LOT EVALUATION

In this section, we test ParkScan with the data collected from two real-world parking lots to investigate its performance in the parking lot scenario.

6.1 Experiment Methodology

We conducted the parking lot evaluation in two different parking lots. The first parking lot (called parking lot 1 hereafter) is the one mentioned in Section 4.1. Since our parking decision model is learned from parking lot 1, we also include a second parking lot (called parking lot 2) to validate the applicability of ParkScan. As shown in Fig. 9, parking lot 2 has 40 parking spots (including two handicapped spots). People come from both exits of the parking lot and go to a grocery store. Since people visit the grocery store at will and they spend on average 15 mins for each visit, the occupancy rate of parking lot 2 is generated by continuous parking and leaving events.

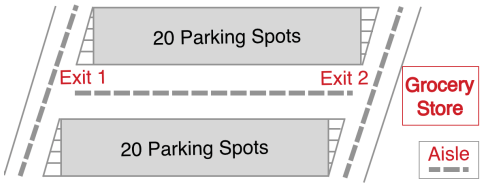

Fig. 9. Layout of parking lot 2

We use the parking search trajectory and ground truth availability collected in Section 4.1 as the request set for the parking lot 1. Similarly, for the parking lot 2, we also video recorded the real parking trajectories and parking lot availability ground truth in 3 weekdays during the busy hours of the grocery store (from 3:15 pm to 5:15 pm).

For experiments in both of the parking lots, we use the parking decision model learned from parking lot 1 in ParkScan. Since parking lot 2 has a different layout and a different visit pattern, the results for parking lot 2 validate the generalization of the parking decision model learned from parking lot 1. For the experiments on parking lot 1, we use a 10-fold cross-validation (based on a chronological partition by dates) to avoid the over-fitting problem for the parking decision model.

6.2 Baseline Solution and Evaluation Metrics

Since no prior study has looked into the spot-level background parking availability estimations (details are given in the related work section), we use the historical availability rate at each spot as the baseline solution for the comparison. We measure the deviation of the probability estimations from the ground truth availability as the estimation error, defined as:

$$error_i = \begin{cases} 1 - p_i(\hat{empty}), & if\ spot\ i\ is\ empty \\ 1 - p_i(\hat{full}), & otherwise. \end{cases} \quad (6)$$

where $error_i$ represents the error of spot i; $p_i(\hat{empty})$ and $p_i(\hat{full})$ are the estimated probabilities for the spot i to be empty or full, respectively.

Estimation errors in equation 6 demonstrate the accuracy for a particular estimation. To evaluate the estimation performance for a group of estimations, we defined the mean absolute deviation (MAD) as follows:

$$MAD = \frac{1}{N}\sum_{k=1}^{N} |\, error_k \,|, \tag{7}$$

where $error_k$ stands for the error of the k-th estimation among all N estimations.

Finally, we defined another metric, i.e., relative error reduction (RER), to measure the accuracy improvements produced by ParkScan and comparing it to the baseline method:

$$RER = \frac{error_{baseline} - error_{parkscan}}{error_{baseline}}, \tag{8}$$

where $error_{baseline}$ and $error_{parkscan}$ denote the errors for the baseline estimations and errors for the estimation using ParkScan. The RER for MAD can also be defined similarly. Note that, a positive RER value illustrates that the ParkScan estimation accuracy outperforms the baseline estimation.

6.3 Evaluation Results

In this section, we present the results for (i) the spots visited by a single parking search trajectory (called one-pass estimation), and (ii) all the spots within the parking lot considering multiple visits from different drivers at different times (called estimation fusion experiment).

6.3.1 One-pass Estimation. In this evaluation, we apply the method presented in Section 5.1 and Section 5.2 to estimate the probabilistic availabilities for the spots that are along any search trajectory. We compute the MAD for all estimations and then calculate the RERs for the MAD using Equation 8.

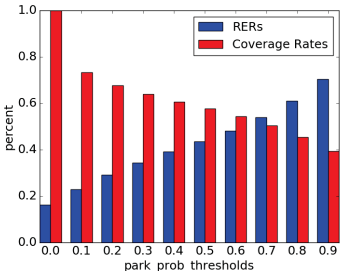

Fig. 10. One-pass estimation results in parking lot 1

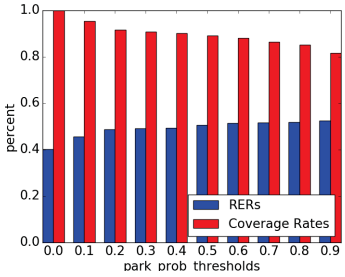

Fig. 11. One-pass estimation results in parking lot 2

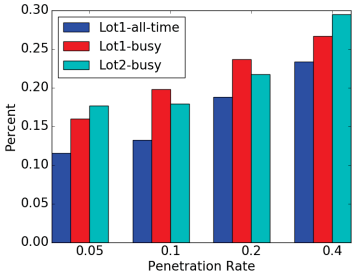

Fig. 12. Mean RERs for various penetration rates under different experiment settings

Fig. 10 and Fig. 11 show how the RERs change in two experimental parking lots while we generate estimations for the only those visited spots whose $p_{u,\pi,i}\,(park \,|\, empty)$ are bigger than the given likelihood threshold. Moreover, we also added the ratio of visited spots which produce estimations out of the total number of spots visited. The results show that even considering the estimation using only a single pass-by visit from one driver, ParkScan provided over 15% estimation accuracy improvements in parking lot 1 and 40% in parking lot 2 compared to the historical estimations. Moreover, as we raise the $p_{u,\pi,i}\,(park \,|\, empty)$ threshold, the estimation accuracy continuously rises. In particular, one-pass estimation could reduce the

MAD by over 70% for the spots whose $p_{u,\pi,i}(park \mid empty)$ are larger than 0.9 in parking lot 1 and over 50% in parking lot 2 for the same likelihood threshold.

By comparing Fig. 10 and Fig. 11, we found that different layouts and visit patterns in the two parking lots change the coverage rates and RERs defined by the same likelihood threshold. For instance, since the walking distance feature for the spots in parking lot 2 is smaller than that in parking lot 1, the results for parking lot 2 show larger $p_{u,\pi,i}(park \mid empty)$ likelihood value in general. However, if we consider all spots visited by the parking search trajectory (left-most bar in either figure), parking lot 2 actually shows improved RER, which suggested that ParkScan system has a good generalization property even when using the default parking decision model.

6.3.2 Estimation Fusion. In this evaluation, we apply an estimation fusion algorithm presented in Section 5.3 to provide the parking availability estimations for any spot each minute. Since the number of participant drivers influences the spatiotemporal coverage of spot visits along detected search trajectories, we also test different penetration rates to show the scalability of the ParkScan system. For the performance measurement, we calculate the MAD for all spots in each minute if there is at least one on-going parking search process. The MADs from ParkScan are also compared to the historical estimation MADs to present the RERs.

RERs of MAD for different penetration rates. Fig. 12 demonstrated the mean RERs under different system penetration rates for different parking lots or time intervals, i.e., all time or only busy hours. One can find from the figure that ParkScan reduces over 11% of MAD at a 5% penetration rate, and the performance gains grow when the penetration rate increases. By comparing the all time results (i.e., the first bar in each cluster) and the busy time results (the second bar in each cluster) in parking lot 1, we find that the parking availability estimation performance of ParkScan is obviously better during busy parking hours. This is caused by two reasons: (i) the searching trajectory is usually long during busy hours, which increases the coverage of the visited spots along the search trajectories; (ii) a drivers parking decision during the busy parking hours is more predictable, shown by the high prediction model accuracy presented in Section 4.3. These two reasons bring about the better performance for ParkScan during busy hours, exactly when people need the system the most. Finally, by comparing the RERs in parking lot 1 and those in parking lot 2, we confirmed that the default parking decision model works well in the new parking lot setting (i.e., parking lot 2), leading to even better RERs during the busy hours.

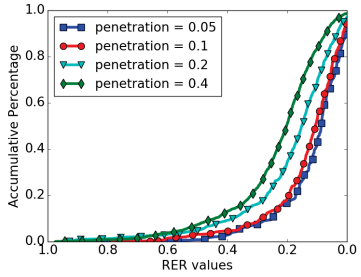

Fig. 13. CDF of RER for various penetration rates in lot 1 (all time)

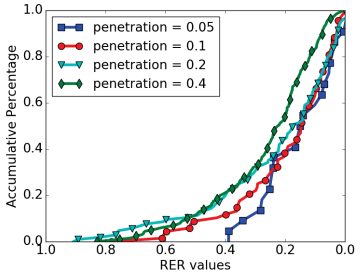

Fig. 14. CDF of RER for various penetration rates in lot 1 (busy time)

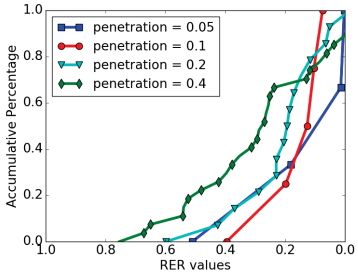

Fig. 15. CDF of RER for various penetration rates in lot 2 (busy time)

To understand the distribution of performance improvement among the measurements at different times of day (e.g. whether it is influenced by only heavier striker), we present the CDF of RERs in different

settings in Fig. 13 to Fig. 15. All three figures demonstrate that the RERs are normally distributed. Moreover, the CDF of RER curves are generally pushed to the up-left side when the penetration rate increases, implying that higher penetration rates raise both estimation accuracy and coverage.

Comparison of MAD by using only one-pass estimation and estimation fusion. The RER values in the estimation fusion experiment are numerically less impressive than those from the one-pass estimation, e.g. by comparing Fig. 12 to Fig. 10. However, this is because in the estimation fusion experiment, we consider all parking spots in each one-minute interval instead of evaluating only the covered spots at the visited time. To show the benefits of estimation fusion algorithms presented in Section 5.3, we computed the MAD of the estimation fusion results and that using the one-pass estimation from the latest spot visit to calculated the RER between these two MADs. Fig. 16 demonstrated positive RERs of estimation fusion results with respect to the one-pass estimation at all penetration rates, meaning that our estimation fusion algorithm generates better spot availability estimations. In particular, it could reduce the MAD of one-pass estimation by up to 55% in the parking lot 1 experiments. Since we only have 3 days historical data in parking lot 2, the statistics, especially the fill/vacating rate statistics, are not reliable. As a result, parking lot 2 shows less improvement in Fig. 16. However, the parking lot 2 experiments to some extent demonstrated the robustness of our estimation fusion method over the noise of historical statistics, because the estimation fusion results still show improvements compared to the one-pass estimation (positive RERs) for parking lot 2.

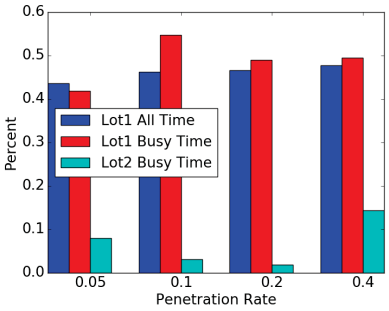

Fig. 16. RERs of estimation fusion with respect to the one-pass estimation for various penetration rates in different settings

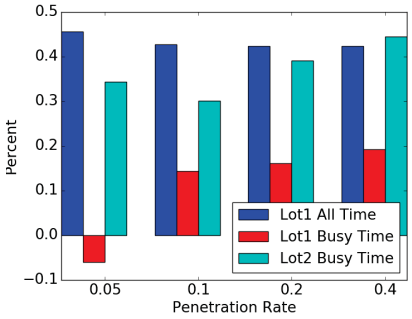

Fig. 17. RERs of ParkScan estimation with respect to the coarse-grained solution for various penetration rates in different settings

Comparison of MAD by using ParkScan and the coarse-grained solution. Though the spot-level historical availability rate is used as the baseline to conduct most comparisons, we finally compute the MAD of ParkScan and that generated using PhonePark [44] to compare ParkScan with prior coarse-grained solutions. Note that, PhonePark also combines the detected realtime parking events and historical statistics as ParkScan does but uses Kalman filter to estimate the parking availability rate of a street parking section or a parking lot. Moreover, PhonePark additionally assumes that the penetration rate of the participant drivers is known in the targeted parking section. Since PhonePark cannot differentiate the spot-level parking availability within the targeted section, for the parking lot scenarios, we use the parking availability of the whole lot as that of each spot to compute the MAD. Fig. 17 demonstrates that in most cases ParkScan provides better spot-level availability estimation than PhonePark (by positive RER values). The only exception happens in parking lot 1 when the penetration rate is 0.05 during the busy time and it is caused by two reasons: (i) We provide PhonePark with the ground-truth penetration rate (not used in ParkScan), which enables PhonePark to very accurately estimate the overall parking

Table 2. Seattle dataset description

Pay Stations	121 stations (road segments)
Parking Transactions	63960 parking requests
Parking spots	1276 spots
Time period	weekdays from 2015-03-02 to 2015-03-27

availability rate of the parking lot. (ii) The vast majority spots in parking lot 1 are usually taken during the busy time, meaning that the spot-level parking availabilities are all close to the overall availability of the parking lot. However, ParkScan outperforms PhonePark by up to over 40% when the penetration rate increases, or when looking at parking lots with different availability distribution pattern (e.g., parking lot 2 has fluctuant availability since people frequently come to and leave the store).

6.3.3 Evaluation Summary. During the parking lot evaluation, we collected driver's real-world parking search trajectories and tested ParkScan in two different parking facilities under different penetration and time interval settings. The key findings from the parking lot evaluation are as follows: (i) ParkScan reduces the estimation errors, compared to historical estimations for visited spots by a single parking search trajectory (shown in Fig. 10 and Fig. 11). This implies that ParkScan has the ability to work at an extremely low penetration rate, e.g. as low as only a single participant driver. (ii) ParkScan effectively reduces the estimation errors for all spots at the moment when drivers need the estimation map (shown by Fig. 12 to Fig. 17). (iii) During the experiment, we learned the parking decision model from parking lot 1 and applied it to a different parking lot (i.e., parking lot 2) with a different layout and different visit patterns. The good performance of ParkScan in parking lot 2 suggested that the parking decision model has good generalizability and does not need to be retrained for every new deployment.

7 STREET PARKING EVALUATION

As mentioned in Section 5.4, we treat on-street parking as a special case, where each individual driver has her/his own start of a search trajectory and a consideration set of all possible spots. In this section, we investigate the performance of ParkScan in the street parking scenario.

7.1 Methodology

We leverage the on-street parking facility map and the parking transaction dataset [14] from Seattle, WA in the street parking evaluation. The parking facility map has the information of all the parking payment stations, including the station ID, the type description, the GPS coordinates, the effective time and so on. Based on the station coordinates, we associate the parking station with a road segment using OpenStreetMap and the nearby parking spots based on data collected from Google Street View.

For the evaluation setup, we first select the experiment area. After visualizing the parking transactions (shown in Fig. 18), we find that Belltown area is the hottest parking region in the city. Since a parking crowd-sensing system is most needed in a busy parking area, we choose the region that is bordered by 5th Ave, Vine St., Elliott Ave., and Stewart St. as our experiment area. We filter the city-wide parking transaction data to only preserve the data from the experiment area (its statistics is shown in Table 2 and its temporal distribution of transactions is shown in Fig. 19). Note that, only paid on-street parking spaces are available for public use in the experimental area, so the parking transactions cover the vast majority of all parking requests in this area.

Fig. 18. Hottest parking area of Seattle

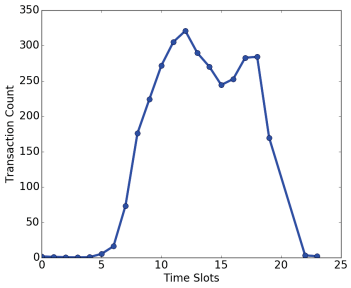

Fig. 19. Average parking transaction count in a day

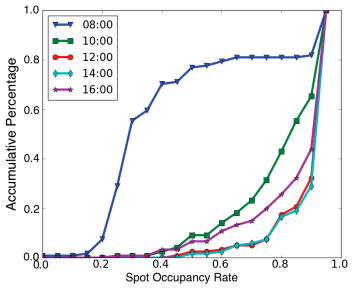

Fig. 20. Occupancy rate distribution

7.1.1 Total Parking Requests and Parking Availability. The experiment needs the parking requests, i.e. the time-destination pairs, to initiate parking search trips. Since parking transaction is associated with a payment station, in this experiment, we assume the driver's destination is a random point along the street where the payment station is located. Moreover, considering potential parking violations, we use the model between the paid parking time and the total parking time presented in [36], to estimate the ratio of paid parking requests among all parking requests on each road segment. We use the estimated ratio to super-sample the parking request set generated by the parking transactions, and obtain the extrapolated parking requests representing all parking demands. Once the total parking request set are identified, we also get the ground-truth of the parking availability at any time of a day (statistics are shown in Fig. 20), because it is essentially a parking/leaving event that causes the parking availability to change.

7.1.2 Parking Search Simulator. Since there is no parking search trajectory in the transaction dataset, we rely on a simulator to generate the trajectory and the final parked spot. Given a final destination and a starting point far enough from the destination, the parking search simulator assumes that the driver is rational and always turns to the direction that minimizes the expected total travel time (driving time plus walking time) based on the historical availability. When an available spot is found, the simulator utilizes our parking decision simulation model, which takes extensive features than the estimation model (refer to Section 4.3 for the simulation/estimation model), to decide whether that spot should be taken. This process iterates until a spot is taken eventually. Note that a distance threshold is defined as a parameter for the parking simulator to determine the consideration candidate spots. This parameter also defines the start of the search trajectory, which is the first point located within the distance threshold when the vehicle approaches its destination. We tuned this parameter in Section 7.2.

7.1.3 Experiment Process. After the experiment starts, the parking requests are picked one by one from the total parking request set to simulate the parking search process, which includes the parking search trajectory and the final parked spot using the parking search simulator (Section 7.1.2). The state of a taken spot will be updated in the ground truth availability map so that subsequent drivers will not park there until the parked car's paid parking time expires.

During the experiment, the ParkScan system obtains only the whole movement trajectory and the final destination. ParkScan infers the start point of the search trajectory and the driver's consideration set and then estimates the spot-level parking availability. Since the ground truth availability map is maintained by the simulator, the estimation results from ParkScan can be evaluated.

We extracted the parking request set for all workdays in March 2015 to generate historical knowledge information, i.e., historical parking availability rates and historical park/vacate probabilities (discussed in

Section 5.3) at the road segment level. However, due to the extensive computation cost to simulate the step-by-step vehicle movements, we chose the parking requests for one day (March 6, 2015) from 8:00 am to 5:00 pm to evaluate the parking estimation performances of the proposed ParkScan system.

7.2 Validation of Experiment Settings

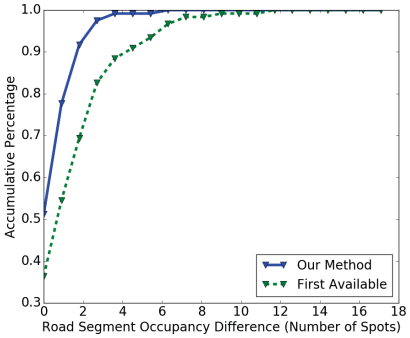

Fig. 21. The CDF of occupancy differences generated by different simulation algorithms

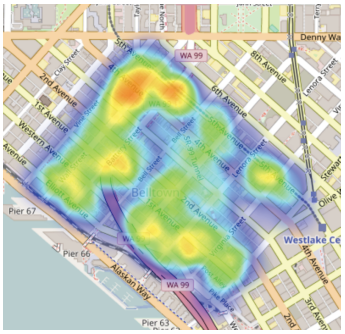

Fig. 22. Parking occupancy at 9:00 am generated by our simulation algorithm

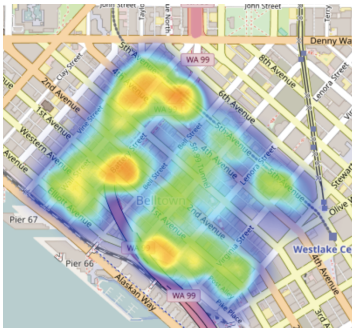

Fig. 23. Parking occupancy at 9:00 am generated by real parking transactions

To verify our simulation design and tune the parameter of distance thresholds in the parking search simulator, we run the parking search simulator presented in Section 7.1.2 for March 6, 2015, starting from 8:00 am to 5:00 pm to simulate the cars' final parking positions. As the baseline, we also test another simulator which assumes drivers will take the first available spot while searching for parking. We compare the resulting occupancy map from the two simulators to the ground-truth occupancy generated using the parking transaction. An ideal simulator with a good parameter setting should generate realistic parking search trajectory and thus incur smaller occupancy difference.

The comparison result demonstrates that our simulation method relying on parking decision simulation model substantially outperforms the baseline simulator when we set the distance threshold as 150 meters. Fig. 21 shows the road segment level occupancy differences after 2 hours of the simulation. In particular, 95% of the road segments have an occupancy difference of fewer than 3 cars. Moreover, since quite a few car's final parking positions are just one road segment shifted from the actual parking position, the actual occupancy map shows only subtle differences, e.g. Fig. 22 and Fig. 23 visualize the heap map of the two resulting occupancy maps at 9:00 am. These results demonstrate that our way to simulate the driver's parking search trajectory is realistic.

7.3 Evaluation Results

In this section, we use the same metrics and evaluation process as in Section 6.3 to test the performance of ParkScan in the street parking scenario.

7.3.1 One-pass Estimation. Fig. 24 illustrates the RER and coverage rate results from the one-pass experiment in the Seattle street parking scenario. One-pass estimation computes the availability prediction for all spots along the reported search trajectory. Similar to the parking lot results, the RER values in the street parking one-pass experiment also increase as $p_{u,\pi,i}\,(park \mid empty)$ threshold values increase. However, the coverage rate drops sharply from the threshold value $p_{u,\pi,i}\,(park \mid empty) = 0.2$. This phenomenon is

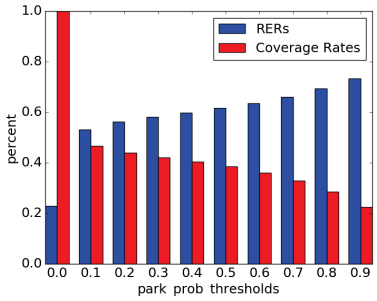

Fig. 24. One-pass estimation results in the Seattle street parking experiment

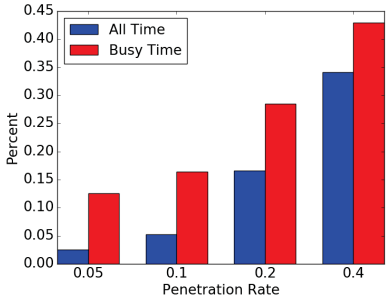

Fig. 25. Mean RERs for various penetration rates in the Seattle street parking experiment

reasonable because the drivers may visit a lot of spots along the reported driving trajectory, however, they only consider the "good" parking candidates limited to a smaller number of spots that are close to the drivers' final destinations. Our parking decision model naturally differentiates the "good" candidates and the rest by generating different $p_{u,\pi,i}\,(park \,|\, empty)$ likelihood values. Motivated by this finding, we set the likelihood threshold of $p_{u,\pi,i}\,(park \,|\, empty)$ as 0.2 in the estimation fusion process to determine the start position of each search trajectory (refer to Section 5.4 for details).

7.3.2 Estimation Fusion. We also applied the estimation fusion algorithm presented in Section 5.3 in the street parking scenario. For the performance measurement, for each minute we calculate the MAD for all spots in the simulation urban area, if there is at least one ongoing parking search process. The MADs from ParkScan are compared with the historical estimation MADs to compute the RERs.

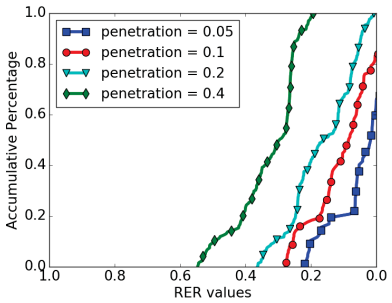

Fig. 26. CDF of RERs for various penetration rates in the Seattle street parking experiment (all time)

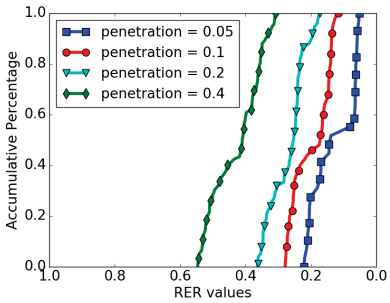

Fig. 27. CDF of RERs for various penetration rates in the Seattle street parking experiment (busy time)

RERs of MAD for different penetration rates. Fig. 25 demonstrates the mean RERs under different system penetration rates over different time interval settings. In the street parking experiment, when the penetration rate is 5%, we achieve 2.5% RER. However, when the penetration rate rises to 40%, the RER of MAD grows to 34.1%. During the busy parking hours, the RERs increase from 12.9% at a 5% penetration to 42.8% at a 40% penetration. The trends are in accordance with the findings in the parking lot experiment. Fig. 26 and Fig. 27 further illustrate the CDF of RERs for the street parking experiment for all intervals throughout the experiment and for busy parking hours, respectively. Similar

to the parking lot experiment, the benefits in the street parking experiment are also normally distributed among most time. In addition, a higher penetration rate pushes the CDF of RER curve to the up-left side, implying increased benefits in both estimation accuracy and coverage. In the street parking all time results (Fig. 26), low penetration rates sometimes lead to worse estimation accuracies comparing to the historical data. This is because we use a stochastic process to make parking decisions in the parking search simulator: even if the predicted $p_{u,\pi,i}\,(park)$ is higher than 0.5 for an available spot, there are still chances that a driver will ignore that spot. When the penetration rate is low, this random decision process generates considerable noise, which causes wrong predictions at some visited spots. Fortunately, this type of errors is mitigated as the penetration increases or during the busy parking hours.

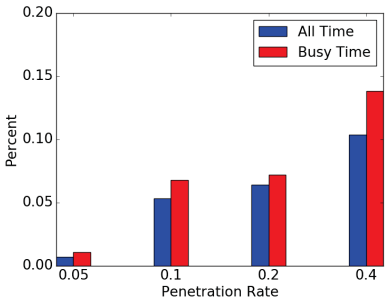

Fig. 28. RERs of estimation fusion with respect to the one-pass estimation for various penetration rates

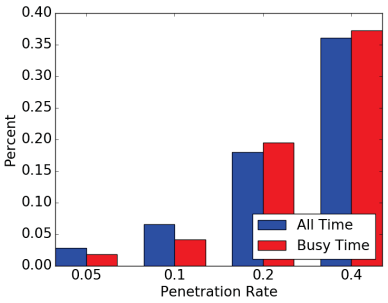

Fig. 29. RERs of ParkScan estimation with respect to the coarse-grained solution for various penetration rates

Comparison of MAD by using only one-pass estimation and estimation fusion. Fig. 28 demonstrates positive RERs of estimation fusion results in street parking scenario with respect to the one-pass estimation at all penetration rates, meaning that estimation fusion produces better spot availability estimations in the street parking scenario. The estimation fusion shows less improvement in the street parking scenario than the parking lot 1 experiment. This is because the park/vacate probabilities estimated from the historical data in the Seattle parking transaction are different from the real rate generated from our vehicle parking simulation. However, the positive RER values also demonstrate the robustness of our estimation fusion algorithm over noise park/vacate rate estimations.

Comparison of MAD by using ParkScan and the coarse-grained solution. Similar to the parking lot evaluation, we also compare the spot-level availability estimation using ParkScan and the coarse-grained availability estimation using PhonePark [44] in the street parking scenario. We provide the PhonePark with ground-truth penetration rates and use Kalman filter to combine the detected real-time parking events and historical statistics. The section unit used in PhonePark is each street parking section, i.e., road segment. Fig. 29 demonstrates that ParkScan provides more accurate spot-level availability estimation than PhonePark (by positive RER values). Moreover, by comparing the RER values at different penetration rates, we find that the performance improvement of ParkScan increases with rising penetration rate in the street parking scenario.

RERs of MAD for different likelihood thresholds. Section 7.3.1 determined the likelihood threshold to detect the start of a search trajectory and the consideration candidates set. In this evaluation, we tested the threshold values from 0.0 to 0.8 under different penetration rates to evaluate the likelihood threshold configuration. Fig. 30 and Fig. 31 show the CDF of RERs for different likelihood settings when the penetration rate is 0.1. In both figures, a threshold of 0.2 provides the best performance,

which validates our setting of the likelihood thresholds in Section 7.3.1. This result is consistent with all penetration rates in both experimental interval settings, though, due to the limit of space, we do not present the figures for other penetration rates. Interestingly, it is also shown in Fig. 30 and Fig. 31 that different thresholds achieve similar results. This implies that ParkScan is not very sensitive for likelihood thresholds, so this value does not need to be precisely tested for each real-world deployment.

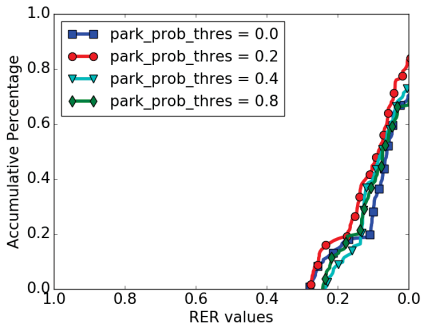

Fig. 30. CDF of RER for various park probabilities (all time, penetration rate = 0.1)

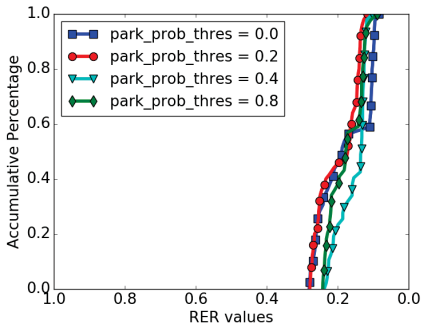

Fig. 31. CDF of RER for various park probabilities (busy time, penetration rate = 0.1)

7.4 Evaluation Summary and Limitations

Our findings from street parking experiments are as follows: (i) ParkScan reduces the estimation errors compared to historical estimations, when considering only visited spot at the visit moments (shown in Fig. 24). (ii) ParkScan reduces the estimation errors for all spots at the moment when drivers need the estimation map (shown in Fig. 25 to Fig. 29). (iii) The method to estimate the parking search start and consideration set improves the performance on ParkScan for the street parking scenario (as shown in Fig. 30 and Fig. 31). Moreover, ParkScan is not very sensitive to threshold values, which determine the park search start. So we can start from a default value (e.g. 0.2) for the real world deployment.

In the street parking experiment, since we do not have as detailed data as the parking lot experiment. We made a few assumptions (e.g., a driver's final destination is a random point along the parked road segment; a simulator starts the parking simulation from a random point 150 meters from the final destination) to build the experiment setting. Moreover, we made an approximation to estimate the historical statistics in the unit of road segments instead of spots. The assumptions on the experiment setting mainly influence the factuality of the ground truth in our simulation towards the real situation in Seattle. When assumptions change, the inputs of ParkScan (i.e., search trajectory and final parked position) also change. Therefore, we believe the influence of these assumptions on the ParkScan performance measurement is low, especially after we have validated the results of these settings in Section 7.2. The approximation of the historical statistics is due to the fact that we only have road segment level information (i.e., parking transactions are associated with payment stations at each road segment). We acknowledge that this approximation may influence the evaluation results, because the historical availability rate is the baseline algorithm. However, since our ParkScan is also fed with the historical statistic data with the same unit, the improvement on this approximation will also improve the performance of ParkScan.

8 DISCUSSION

Acquisition of driver's destination and search trajectory ParkScan parking availability estimation leverages the driver's destination and the parking search trajectory, which is not uncommon in other

parking crowdsourcing systems, e.g., [28]. The final destination can be obtained from vehicle navigation system or estimated using place visit detections and indoor localization technologies [15, 18, 28]. Note that, ParkScan may work with any parking/unparking detection technique to detect the parked positions and can tolerate location noises. Given that vehicle tracking becomes more accurate in both outdoor [1, 16] and indoor environments [39, 47], estimating the parking trajectory is increasingly feasible.

Generalization of the parking decision model Section 4 builds the parking search model for a group of people that use a particular parking lot. Since the estimates from multiple sources are fused to eliminate the individual errors, ParkScan works well even if the model is not tailored for each participant driver. However, we do acknowledge that the parking decision model of individuals may be different, for instance, some driver may prefer shadow spots during the summer. Once the ParkScan system has been installed in the driver's smartphone, we could use opportunistic sensing to generate samples to refine driver's decision model, e.g., parking at a spot generates a positive parking decision sample, and ignoring a spot, which according to a dedicated parking status sensor or our estimation is almost 100% sure available, generates a negative parking decision sample. We believe this will further improve the performance of ParkScan.

Historical data and parking layout map ParkScan leverages the historical data when estimating the background parking availability. Without the initial historical data, ParkScan may need a long time to converge. Fortunately, given the development of the internet of things, we can retrieve historical statistics from many government resources (e.g., we compute historical statistics using the parking transaction dataset in Seattle) and online parking information providers (e.g. ParkMe [13] provides the historical parking availability of many cities). Parking layout maps are also available from map providers or government open datasets. Inspired by the recent advances to build road and building maps from satellite images [27, 46], parking layout maps can also be potentially extracted using similar technologies.

Enhancement with sensor data, coarse-grained estimation and more powerful models ParkScan currently relies on a Bayesian inference model to estimate spot-level parking availability. The Bayesian model has minimum deployment requirements, i.e., it only needs the parking search trajectory, trip destination, historical statistics, and facility layout to estimate parking availability. However, the Bayesian model also has limitations. For instance, the current ParkScan system may provide biased estimations, because based on the Bayesian model, passing by a spot without taking it never makes the availability probability higher than the historical availability rate. While some applications, e.g. routing, just need the relative availability rate instead of the absolute values, we argue that ParkScan can be enhanced by using dedicated parking sensor data, coarse-grained crowdsourcing approaches, and more complex models. For example, the ParkScan estimation result of a spot can always be over-written by the sensor monitoring the same spot. Even if only the coarse-grained parking availability can be measured using sensors, e.g. gate counter, or coarse-grained crowdsourcing approaches, we can use the coarse-grained availability statistics to renormalize the ParkScan results. Finally, if adequate data is accessible for a parking facility, more powerful models with additional features can be explored. For instance, traffic demand (e.g. taxi pickup/drop-offs), POIs data, and social network check-ins can be used in addition to the $P(empty|\neg park)$ as new features. We expect these features can be combined together (e.g. by tensor decomposition) to improve the accuracy of the parking availability estimation.

9 RELATED WORK

ParkScan is a crowdsourcing system relying on a parking decision model to compute probabilistic posterior estimations of spot-level parking availability. So both smart parking systems and parking search process modeling are related to our work.

Table 3. Smart parking system survey

Scenarios		Dedicated Sensors	Dedicated Sensor Free
Park/Unpark Event Detection		Spark[21], RFID or computer vision solutions	ParkSense[29], Updetector[22], PhonePark[44], PocketParker[28], BluePark[39]
Background Availability Detection	Coarse-grained	ParkNet[26], Parking Gate Counter†, etc.	PhonePark[44], PocketParker[28]†, ParkGauge[8]†, SPIRE[33]†
	Fine-grained	ParkNet[26], SFPark[37], LA Express Park[19], QuickSpot[24]	ParkScan (This work)

Note: † solutions only for off-street parking usage.

9.1 Smart parking systems

Table. 3 surveyed the smart parking systems, which can be divided into solutions based on dedicated sensors and dedicated-sensor free solution. Smart parking systems using dedicated sensors have been excessively studied back to the last decade [12]. These solutions can be used to detect the parking/unparking event for a particular vehicle [21] or detect the availability of spots [19, 37]. Recently, the sensors are deployed in a mobile fashion to increase the coverage [26]. However, suffering from the cost issues in installation and maintenance, these solutions are deployed in very limited regions. The dedicated-sensor-free solutions rely on the sensors already built in smartphones or other wearable devices. They use various sensors to detect the transition between driving and walking to estimate the parking/unparking events [22, 28, 29, 39, 44]. Systems use either sampling theory [44], or use some special heuristics for off-street parking, e.g. a parking lot is very likely to be full if a car goes in and comes out without parking inside [28, 33] or parking search time (and lower-level vehicle maneuvers, e.g., brakes and turns) is related to parking availability statistics [8], to infer background parking availabilities under low penetration rates.

Our work looks into the human's probabilistic parking decision model, which is dedicated sensor free and can be used both for on-street and off-street parking scenarios. Moreover, the system provides fine-grained and reliable estimations under a low penetration rate rather than only coarse-grained statistics.

9.2 Parking Behavior Modeling

The modeling of drivers' parking behavior in the transportation domain can be divided into parking choice modeling and parking decision modeling. The parking choice model is a decision maker that selects a parking space from given options. Many data-driven parking choice models have been proposed, e.g. [4, 7, 10, 43]. Prior studies on the parking decision model (the decision maker to decide whether to take the current spot or not) mostly use ad-hoc models without using real-world data, e.g. [3, 25, 40, 45], or assume that drivers are fully rational when facing parking decisions or routing choices [5, 23, 41, 42]. The only data-driven study is based on lab questionnaires [6]. Nevertheless, the behavioral studies in both categories show homogeneity of the models among various users and parking facilities [6, 10].

Our paper applied a data-driven approach to learning the parking decision model from real-world parking decision events with human decisions. We automate the data collection process through computer vision techniques and thus make the calibration of parking decision models practically feasible. To the best of our knowledge, this is the first work that extracts fine-grained trajectory data and individual parking decisions to build a parking decision model.

10 CONCLUSION

This paper presented a parking availability crowdsourcing system, ParkScan, based on the key idea that the behavior of a parking seeker's ignoring a visited spot probabilistically implies the unavailability of that spot. We collected an extensive dataset containing 8,000 vehicle trajectories and over 55,000 human parking decisions, and we utilized a data-driven approach to building a parking decision model. Using the learned model, we provided an efficient way to estimate the priors used in the Bayesian rule and presented an approach to combine estimations from multiple spot visits covered by different parking search trajectories. Finally, we evaluated the ParkScan system using real-world data both from the parking lot and the street parking scenarios. Both of the experiments show that ParkScan improves the fine-grained parking availability estimations, even under extremely low penetration rates.

ACKNOWLEDGMENTS

We thank Dr. Vinod Ganapathy for his help and support on this study and the anonymous reviewers for their valuable comments. This work is supported in part by the US National Science Foundation under grant number CNS-1111811 and grant number CNS-1420815, and by a Google Research Award.

REFERENCES

[1] Heba Aly and Moustafa Youssef. 2013. Dejavu: an accurate energy-efficient outdoor localization system. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 154–163.

[2] Alessio Bechini, Francesco Marcelloni, and Armando Segatori. 2013. A mobile application leveraging QR-codes to support efficient urban parking. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2013*. IEEE, 1–3.

[3] Itzhak Benenson, Karel Martens, and Slava Birfir. 2008. PARKAGENT: An agent-based model of parking in the city. *Computers, Environment and Urban Systems* 32, 6 (2008), 431–439.

[4] Angela Stefania Bergantino, Angela De Carlo, Andrea Morone, and others. 2015. *Individuals' behaviour with respect to parking alternatives: a laboratory experiment*. Technical Report. University Library of Munich, Germany.

[5] Stephen D Boyles, Shoupeng Tang, and Avinash Unnikrishnan. 2015. Parking search equilibrium on a network. *Transportation Research Part B: Methodological* 81 (2015), 390–409.

[6] Michal Chalamish, David Sarne, and Sarit Kraus. 2007. Mass programmed agents for simulating human strategies in large scale systems. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. ACM, 135.

[7] Emmanouil Chaniotakis and Adam J Pel. 2015. Drivers' parking location choice under uncertain parking availability and search times: A stated preference experiment. *Transportation Research Part A: Policy and Practice* 82 (2015), 228–239.

[8] Jim Cherian, Jun Luo, Hongliang Guo, Shen-Shyang Ho, and Richard Wisbrun. 2016. ParkGauge: Gauging the Occupancy of Parking Garages with Crowdsensed Parking Characteristics. In *Mobile Data Management (MDM), 2016 17th IEEE International Conference on*, Vol. 1. IEEE, 92–101.

[9] Google Developers. 2017. ActivityRecognitionApi. (2017). https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionApi/.

[10] Liya Guo, Shan Huang, Jun Zhuang, and Adel W Sadek. 2013. Modeling parking behavior under uncertainty: a static game theoretic versus a sequential neo-additive capacity modeling approach. *Networks and Spatial Economics* 13, 3 (2013), 327–350.

[11] Martin Hofmann, Philipp Tiefenbacher, and Gerhard Rigoll. 2012. Background segmentation with feedback: The pixel-based adaptive segmenter. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 38–43.

[12] MYI Idris, YY Leng, EM Tamil, NM Noor, and Z Razak. 2009. Car Park System: A Review of Smart Parking System and its Technology. *Information Technology Journal* 8, 2 (2009).

[13] ParkMe Inc. 2016. ParkMe. (2016). https://www.parkme.com/.

[14] Seattle IT. 2015. Seattle Parking Transactions. (2015). https://data.seattle.gov/Transportation/Seattle-Parking-Transactions/updn-y53g/data.

[15] Yifei Jiang, Xin Pan, Kun Li, Qin Lv, Robert P Dick, Michael Hannigan, and Li Shang. 2012. Ariel: Automatic wi-fi based room fingerprinting for indoor localization. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 441–450.
[16] Yurong Jiang, Hang Qiu, Matthew McCartney, Gaurav Sukhatme, Marco Gruteser, Fan Bai, Donald Grimm, and Ramesh Govindan. 2015. CARLOC: Precisely Tracking Automobile Position. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 253–265.
[17] Merkourios Karaliopoulos, Konstantinos Katsikopoulos, and Lambros Lambrinos. 2014. Bounded rationality can increase parking search efficiency. In *Proceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 195–204.
[18] Donnie H Kim, Younghun Kim, Deborah Estrin, and Mani B Srivastava. 2010. Sensloc: sensing everyday places and paths using less energy. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. ACM, 43–56.
[19] LADoT. 2017. LA Express Park. (2017). http://www.laexpresspark.org/.
[20] Ruilin Liu. 2017. Parking Trajectory Dataset. (2017). http://www.cs.rutgers.edu/~rl475/parkscan.html#dateset.
[21] Rongxing Lu, Xiaodong Lin, Haojin Zhu, and Xuemin Shen. 2009. SPARK: a new VANET-based smart parking scheme for large parking lots. In *INFOCOM 2009, IEEE*. IEEE, 1413–1421.
[22] Shuo Ma, Ouri Wolfson, and Bo Xu. 2014. Updetector: Sensing parking/unparking activities using smartphones. In *Proceedings of the 7th ACM SIGSPATIAL International Workshop on Computational Transportation Science*. ACM, 76–85.
[23] Ayub Mamandi, Saleh Yousefi, and Reza Ebrahimi Atani. 2015. Game theory-based and heuristic algorithms for parking-lot search. In *Computer Science and Software Engineering (CSSE), 2015 International Symposium on*. IEEE, 1–8.
[24] Elena Màrmol and Xavier Sevillano. 2016. QuickSpot: a video analytics solution for on-street vacant parking spot detection. *Multimedia Tools and Applications* 75, 24 (2016), 17711–17743.
[25] Karel Martens and Itzhak Benenson. 2008. Evaluating urban parking policies with agent-based model of driver parking behavior. *Transportation Research Record: Journal of the Transportation Research Board* 2046 (2008), 37–44.
[26] Suhas Mathur, Tong Jin, Nikhil Kasturirangan, Janani Chandrasekaran, Wenzhi Xue, Marco Gruteser, and Wade Trappe. 2010. Parknet: drive-by sensing of road-side parking statistics. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 123–136.
[27] Volodymyr Mnih and Geoffrey Hinton. 2010. Learning to detect roads in high-resolution aerial images. *Computer Vision–ECCV 2010* (2010), 210–223.
[28] Anandatirtha Nandugudi, Taeyeon Ki, Carl Nuessle, and Geoffrey Challen. 2014. Pocketparker: Pocketsourcing parking lot availability. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (Ubicomp '14)*. ACM, 963–973.
[29] Sarfraz Nawaz, Christos Efstratiou, and Cecilia Mascolo. 2013. Parksense: A smartphone based sensing system for on-street parking. In *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 75–86.
[30] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. ACM, 336–343.
[31] Gregory Pierce and Donald Shoup. 2013. Getting the prices right: an evaluation of pricing parking by demand in San Francisco. *Journal of the American Planning Association* 79, 1 (2013), 67–81.
[32] G. Revathi and V.R.S. Dhulipala. 2012. Smart parking systems and sensors: A survey. In *Computing, Communication and Applications (ICCCA), 2012 International Conference on*. 1–5. DOI:http://dx.doi.org/10.1109/ICCCA.2012.6179195
[33] Mikko Rinne, Seppo Törmä, and D Kratinov. 2014. Mobile crowdsensing of parking space using geofencing and activity recognition. In *10th ITS European Congress, Helsinki, Finland*. 16–19.
[34] Andres Sanin, Conrad Sanderson, and Brian C Lovell. 2012. Shadow detection: A survey and comparative evaluation of recent methods. *Pattern recognition* 45, 4 (2012), 1684–1695.
[35] Andrew Senior, Arun Hampapur, Ying-Li Tian, Lisa Brown, Sharath Pankanti, and Ruud Bolle. 2006. Appearance models for occlusion handling. *Image and Vision Computing* 24, 11 (2006), 1233–1243.
[36] SFMTA. 2014. Sensor Independent Rate Adjustments Methodology and Implementation Plan. (2014). http://sfpark.org/wp-content/uploads/2014/05/SIRA-methodology-and-implementation-plan_2014_05-14.pdf.
[37] SFMTA. 2017. SFpark. (2017). http://www.sfpark.org/.
[38] Donald C Shoup. 2006. Cruising for parking. *Transport Policy* 13, 6 (2006), 479–486.
[39] Sonia Soubam, Dipyaman Banerjee, Vinayak Naik, and Dipanjan Chakraborty. 2016. BluePark: tracking parking and un-parking events in indoor garages. In *Proceedings of the 17th International Conference on Distributed Computing*

and Networking. ACM, 33.
[40] Thérèse Steenberghen, Karel Dieussaert, Sven Maerivoet, and Karel Spitaels. 2012. SUSTAPARK: an agentbased model for simulating parking search. *URISA Journal* 24, 1 (2012), 63–77.
[41] Shoupeng Tang, Tarun Rambha, Reese Hatridge, Stephen Boyles, and Avinash Unnikrishnan. 2014. Modeling Parking Search on a Network by Using Stochastic Shortest Paths with History Dependence. *Transportation Research Record: Journal of the Transportation Research Board* 2467 (2014), 73–79.
[42] Oliver Ullrich, Benjamin Brandt, Daniel Lückerath, and Naphtali Rishe. A Simple Model of Cruising for Garage Parking. In *Proceedings of ASIM-Workshop STS/GMMS-ARGESIM Report*, Vol. 51. 10–11.
[43] Peter van der Waerden, Aloys Borgers, and Harry Timmermans. 2003. Travelers micro-behavior at parking lots: a model of parking choice behavior. In *Processing of the 82nd Annual Meeting of the Transportation Research Board*, Vol. 1212.
[44] Bo Xu, Ouri Wolfson, Jie Yang, Leon Stenneth, S Yu Philip, and Peter C Nelson. 2013. Real-time street parking availability estimation. In *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, Vol. 1. IEEE, 16–25.
[45] William Young. 1986. A model of vehicles movements in parking facilities. *Mathematics and computers in simulation* 28, 4 (1986), 305–309.
[46] Jiangye Yuan. 2016. Automatic Building Extraction in Aerial Scenes Using Convolutional Networks. *arXiv preprint arXiv:1602.06564* (2016).
[47] Mingmin Zhao, Tao Ye, Ruipeng Gao, Fan Ye, Yizhou Wang, and Guojie Luo. 2015. Vetrack: Real time vehicle tracking in uninstrumented indoor environments. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 99–112.
[48] Onno Zoeter, Christopher Dance, Stéphane Clinchant, and Jean-Marc Andreoli. 2014. New Algorithms for Parking Demand Management and a City-scale Deployment. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. ACM, New York, NY, USA, 1819–1828. DOI:http://dx.doi.org/10.1145/2623330.2623359

Received May 2017; revised July 2017; accepted July 2017