Hard to Park? Estimating Parking Difficulty at Scale

Neha Arora James Cook Ravi Kumar Ivan Kuznetsov Yechen Li Huai-Jen Liang
Andrew Miller Andrew Tomkins Iveel Tsogsuren Yi Wang
Google Research
Mountain View, CA, US
{nehaarora,falsifian,tintin,ivanku,yechenl,hjliang,atmiller,tomkins,iveel,wayi}@google.com

ABSTRACT

In this paper we consider the problem of estimating the difficulty of parking at a particular time and place; this problem is a critical sub-component for any system providing parking assistance to users. We describe an approach to this problem that is currently in production in Google Maps, providing inferences in cities across the world. We present a wide range of features intended to capture different aspects of parking difficulty and study their effectiveness both alone and in combination. We also evaluate various model architectures for the prediction problem. Finally, we present challenges faced in estimating parking difficulty in different regions of the world, and the approaches we have taken to address them.

ACM Reference Format:
Neha Arora James Cook Ravi Kumar Ivan Kuznetsov Yechen Li Huai-Jen Liang Andrew Miller Andrew Tomkins Iveel Tsogsuren Yi Wang. 2019. Hard to Park? Estimating Parking Difficulty at Scale. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330767

1 INTRODUCTION

Americans spend an average of 17 hours per year searching for parking, resulting in an annual cost of $345 per driver in wasted time, fuel, and emissions [18]. This wasted time tends to be particularly stressful, so its impact on well-being may be disproportionate. Parking woes are not limited to travel or leisure activities: we conducted a survey of the US population using a user survey application and found that 29% of respondents who commute by car have parking problems even at their home or work.

Google has a long-standing goal to help people navigate the roads easily and efficiently, and this includes providing support for the difficult problem of parking. Any such solution must take into account the nature of parking at the destination of a journey. If cheap street parking is abundant at the destination, then directing the user to more expensive and remote parking in a garage is a poor experience. On the other hand, it is equally derelict to abandon the user at the front door of a restaurant in a neighborhood with no available parking. To distinguish between these cases, the estimation of difficulty of parking in a particular context (both location and time) is a critical capability. This paper therefore studies the problem of large-scale automated parking difficulty estimation, and describes a system currently in use at Google to solve this problem. Figure 1 shows an example from Google Maps of the inferences produced by this system.

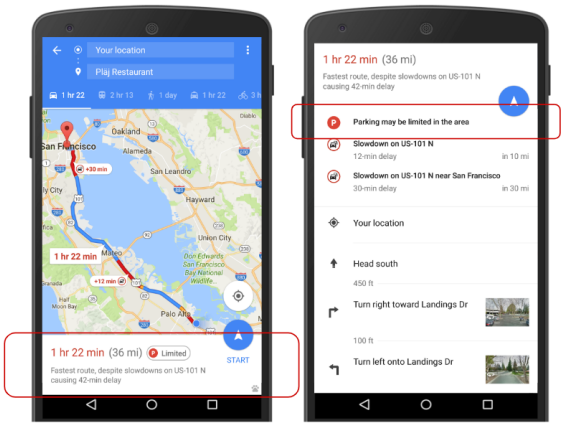

Figure 1: Parking difficulty shown in Google Maps.

Providing this feature required addressing significant challenges:

(i) Parking availability is highly variable from one time and place to another.

(ii) Precise data about parking availability by time and place is limited. Even in areas with internet-connected parking meters providing information on availability, the data does not account for those who park illegally, park with a permit, or depart early from still-paid meters.

(iii) Parking behavior varies from one city to another. It is *a priori* not clear whether a system built to estimate parking difficulty in one place will work well in another.

To address these issues, we developed a system to estimate parking difficulty. This system is trained using crowd-sourced data gathered by surveying Google Maps users about the difficulty of parking at various times and places. Based on this ground truth, we develop an ML-based system to predict this difficulty. Much as Google Maps uses aggregate statistics to estimate the speed of traffic, we use similar aggregate statistics to provide signals on parking difficulty that are trained against our survey-based ground truth. The aggregate statistics we use cover a wide range of different indicators of parking difficulty, in order to train a robust model that is effective across different parking environments. We consider features such as the average distance people park from their destination, the dispersion of parking locations for a particular destination, and the average amount of time spent circling the destination. Overall we consider

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08.
https://doi.org/10.1145/3292500.3330767

approximately 300 such features in our models. As the underlying location trajectory data used to build our features is noisy, the formulation of a feature space that is predictive, and generalizes well across different parking situations around the world, is the primary contribution of our work. We also describe models based on logistic regression and deep feed-forward networks, and study the generalization of our models across different cities spanning multiple continents.

We show that it is possible to develop an effective solution to parking difficulty estimation. The deployment of such a system provides value to users in planning their parking, and perhaps not surprisingly, it also has impact beyond the mechanics of parking itself. In a live experiment comparing the system with and without parking difficulty information, we saw a significant increase in clicks on the transit travel mode button, indicating that users with additional knowledge of parking difficulty were more likely to consider public transit options as an alternative to driving.

The primary contributions of this paper are:

(i) Design and evaluation of a wide range of features that can be produced from location trajectories, and that are effective in measuring parking difficulty.

(ii) Design, implementation, and different modeling approaches to this problem by using aggregates of these features, trained against survey-based ground truth.

(iii) Large-scale quantitative study using a dataset that covers a wide variety of places, generalization to new cities, and importance of different features.

The paper is organized as follows. We survey related work in Section 2, and describe the problem and our approach in Section 3. We describe our data in Section 4, features in Section 5, and modeling in Section 6. In Section 7, we evaluate the relative power of different features and modeling approaches, and the ability of our system to generalize across cities. Section 8 gives concluding remarks.

2 RELATED WORK

2.1 Parking sensors

A significant body of work exists around the detection and analysis of data on whether individual parking spots are occupied at a point in time. One line of work covers purpose-built sensors deployed in parking lots or garages [7, 30, 36, 46]. Another line of work makes use of ultrasonic sensors already available in some passenger vehicles to detect the state of occupancy of parking spots adjacent to the vehicle [4, 8, 25, 44]. Yet another line of work employs techniques from image processing to determine the occupancy of parking spaces based on image data collected from various sources [5, 9, 12, 16, 17, 31, 34, 35, 39]. Yet other solutions have been proposed based on crowd-sourcing data from smartphones [40], GPS [26], and payments [32, 41, 42].

In comparison to these approaches, we instead make use of historical geolocation data and anonymized surveys, which are both scalable and affordable.

2.2 Parking occupancy modeling

From a methodological perspective, various ML/statistical models for predicting parking occupancy have been proposed recently. These techniques cover a broad range of approaches including clustering [28, 32, 33], SVR [44], time-series analysis [22], Markov chains [21], vector autoregressive models [27], neural networks [1, 19, 29, 36, 41], and representation learning [45]. Particularly, Alajali et al. [1] propose a Bayesian regularized neural network that takes into account historical data, traffic flow, and weather conditions. DuParking [29] employs DNN and LSTM models to predict real-time parking availability, based on geolocation data from Baidu map and sensor data from parking lots. Yang et al. [41] incorporate both CNN and RNN/LSTM for modeling spatiotemporal correlations, based on both real-time and historical data from multiple data sources including occupancy, conditions of traffic, road, and weather, as well as network topology. These learning approaches rely in part on parking lot sensor or parking meter data.

2.3 Route planning

In addition to parking occupancy prediction, parking search has also been studied in the context of routing algorithms. ParkAssistant [11] aims to minimize an overall measure of parking cost that incorporates parking price and time, traffic rules, driver preference, and other factors. PSR [15] adapts the A* algorithm in a road network simulation environment. A number of agent-based models [2, 10, 24, 37, 38] perform simulations of driver behavior and parking supply. Other recent parking simulation works [6, 14, 43] have studied key aspects of driver behavior under the uncertainty that arises due to missing information while attempting to park. These simulations are often compute-intensive and require empirical calibrations of driver behavior, often specific to an area. Finally, some recent work proposes other low-cost data sources such as so-called *floating car data* based on mobile network accesses from devices in vehicles [13, 23].

3 FORMULATION AND OVERVIEW

Parking difficulty varies a lot by time and place. Our primary goal in this work is to model this variation in order to give predictions to users about whether they will have difficulty parking.

Goal. Obtain a function f that, given a destination location ℓ and time t, produces a prediction $f(\ell, t) \in \{\texttt{easy}, \texttt{medium}, \texttt{limited}\}$ of whether there will be difficulty parking. For example,

$$f(\ell = 37 \text{ Main Street}, t = 3{:}12\text{pm on January 2}) = \texttt{limited},$$

would be a prediction that parking is likely to be difficult at that time and place.

To make the problem concrete and amenable to standard machine learning approaches, we first partition the possible inputs ℓ, t into "spatiotemporal buckets" (Section 4.2). We compute a feature vector x for each bucket (Section 5). What remains is to learn a model M to predict parking difficulty $M(x)$ based on those features. Putting it all together, these are the steps we use to compute $f(\ell, t)$:

Computing the parking difficulty prediction $f(\ell, t)$.

(1) Find the spatiotemporal bucket B that contains (ℓ, t).
(2) Compute the feature vector x_B for that bucket.
(3) Apply the model M, producing $f(\ell, t) = M(x_B)$.

We extract the features for each spatiotemporal bucket from user location data for users who have opted into Google's Location History service. We segment users' data to produce trajectories where the user arrived at a place by driving and parking a car. We

then compute features for each such trajectory that reflect difficulty in parking, like longer time to park, long walks to the destination or circling around to find parking (Sections 5.1 and 5.2). We then aggregate those features across all trajectories for a place and time, and add other aggregate statistics such as dispersion of parking locations for a particular destination (Section 5.3). We use this combined set of features to train a model (Section 6) to predict survey answers we collect (Section 4.3) to provide ground truth for subjective parking difficulty.

4 DATA

In this section we describe the data we use to build and evaluate the parking difficulty model. We aggregated user location data from Location History-enabled users to find parking difficulty for specific regions on Earth by dividing Earth's surface into discrete regions. We also collected ground truth data through anonymized surveys. We now describe each of these data sources in detail.

4.1 Travel trajectory data

In order to understand the difficulty of parking, it is important to analyze users' spatiotemporal trajectories to record total time, distance from parking location to destination, and various other features. From user location data, we used inferred travel modes (driving, walking, train, etc.) and destinations [20], as shown in Google Maps Timeline. Parking location was estimated to be near the end of the last driving location before destination arrival.

These detected parking locations may be individually noisy, so the models we describe operate instead on aggregates of features, generated across multiple trajectories.

4.2 Defining spatiotemporal buckets

As we discussed in Section 3, we compute parking difficulty for a spatiotemporal bucket B. The spatial boundaries of a bucket are determined based on Google's open-source S2 Geometry Library (s2geometry.io), which decomposes the entire surface of the Earth into hierarchical S2 cells. The top level of the hierarchy is obtained by projecting the points of the sphere (i.e., Earth) into a cube, giving rise to six S2 cells on Earth's surface. The lower levels are obtained by subdividing each cell into four child cells. This continues recursively a total of 30 times. For example, Figure 2 shows two of the six-face cells, one of which has been subdivided several times:

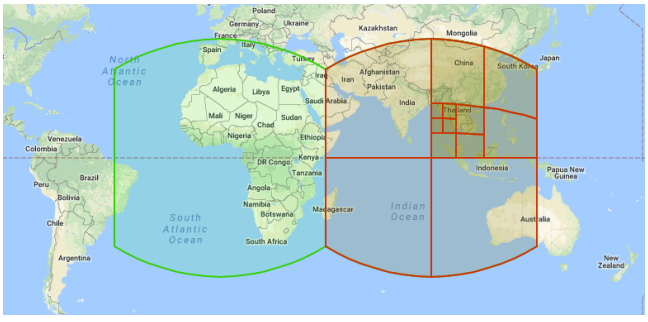

Figure 2: S2 cell hierarchy. From http://s2geometry.io/.

Each S2 cell is a quadrilateral bounded by four geodesics and has a hierarchy level, ranging from 0 to 30. We used S2 cells at level 15, which represents an edge length of about 250 meters and an average area of about 80,000 square meters. This roughly corresponds to a large city block, which is the level of granularity that is desirable for parking difficulty estimation.

For the temporal boundaries of a spatiotemporal bucket B, we use the 168 distinct hours of the week as the temporal regions. Thus, a bucket B is defined by an hour of the week and a level 15 S2 cell.

4.3 Ground truth

To train and evaluate our models, we need labeled data indicating if parking is difficult in a variety of spatiotemporal situations. We gathered these labels using user surveys: we asked individuals at a diverse set of locations and times to answer a yes-or-no question about the current difficulty of finding parking. To produce our dataset, we joined these answers with model features covering the same times and places. The features are described in Section 5.

Our first survey asked "Is it hard to find parking near here right now?" (The surveys described here include translations to other languages.) We learned that answers to subjective questions like this produce inconsistent results—for the same parking scenario, different people give different answers.

We then asked "Does it currently take more than M minutes to find parking and walk to this place?", for different choices of M. We found that for the right choice of M, this gave more consistent answers, enabling us to crowd-source a high-quality set of ground truth data with over 100K responses.

4.3.1 Inter-rater agreement. To assess the quality of our ground truth, we first grouped the survey responses by time and location into spatiotemporal buckets, as described in Section 4.2, since our feature aggregation prevents our model from making any distinctions within the same aggregation bucket. We then computed the inter-rater agreement using joint probability of agreement within each spatiotemporal bucket. Specifically, we measured how often two responses agree when they fall in the same spatiotemporal bucket. For each bucket with a set S of answers, $|S| > 1$, we computed agreement, and averaged the agreement over all the buckets within a geographic region (typically the metropolitan area around a city) to get the overall inter-rater agreement.

Our analysis shows that the inter-rater agreement in the ground truth can vary from 50%–80%. It is affected by a variety of reasons:

(i) Naturally users will have different parking experiences. For example, one user may get lucky and find a spot right away, another might need to hunt for 15 minutes, and a third might simply decide to pay to use a garage.

(ii) We bucket by hour of week, so if labels show that difficulty differs this Tuesday at 2pm from last Tuesday at 2pm, we put those in the same bucket and it counts against inter-rater agreement. Similarly, our spatial bucketing might group together places with different parking situations, which can happen at S2 cell level 15.

(iii) Some survey respondents may provide unreliable data. For example, people who did not drive and do not understand the parking situation may still try to provide a response.

As the agreement figures indicate, even a perfect model will still disagree in some cases with the experiences of an individual parker.

We evaluate model accuracy against average difficulty reported by a set of raters, understanding that this average difficulty represents a distribution of individual experiences with nontrivial variance.

Figure 3 shows the relationship between inter-rater agreement and overall parking difficulty. When it is harder to park, the variance of answers is higher because of a higher difference in individual parking experiences, which in turn results in lower inter-rater agreement as shown in the figure.

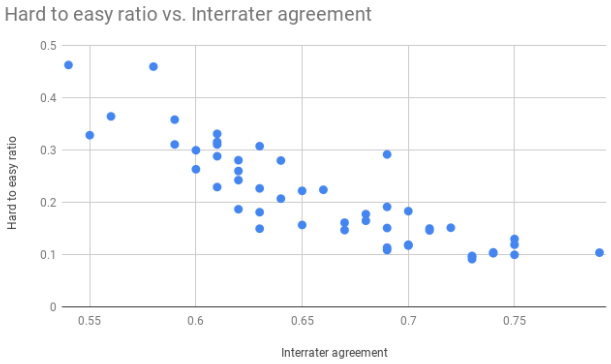

Figure 3: Comparison of parking difficulty and inter-rater agreement. Each point represents one geographic region.

5 FEATURES

In this section we describe the various classes of features we derive from the travel trajectory data described in Section 4.1. We place each trajectory into a spatiotemporal bucket based on the level 15 S2 cell containing the destination, and the hour of week containing the start time of the visit. For each trajectory-level feature, and for each bucket, we then aggregate the value of that feature across all trajectories in the bucket. We use five different aggregation functions: count, 10th percentile, median, mean, and 90th percentile. This allows us to convert any number of trajectory-level features for a bucket into five final aggregate bucket-level features.

Sections 5.1 and 5.2 describe the per-trajectory features that are aggregated in this manner. Section 5.3 describes additional features that use custom forms of aggregation across trajectories.

5.1 Trajectory-based time and distance features

Based on user trajectory data, we compute features that represent the properties of a single trajectory. These features are aggregated as described above before being passed to our model. We now describe the features.

5.1.1 Parking distance. This is defined as the distance from parking location to the destination; as we will see later, this turns out to be one of the most important features. The intuition behind this feature is immediate: if it is easy to find parking for the destination (e.g., street parking before the destination is typically empty or the destination has a customer-only parking lot that is typically empty), then this feature will have a low value. Conversely, if the distance between the destination and the parking location is large, then it is some indication of parking difficulty.

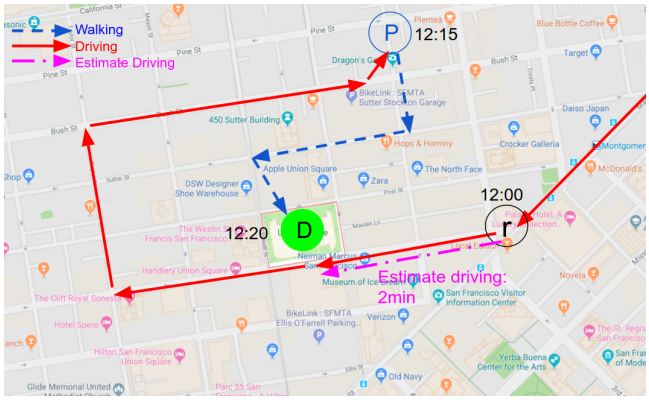

Figure 4: Mock trajectory that demonstrates arrival deviation. See Section 5.1.4. (Not derived from user data.)

5.1.2 Vicinity and destination times. These features compare the time when the user first approaches the destination to the time when they park or arrive at the destination. If the user must circle or drive around for a long time, we expect their values to be higher.

We define multiple variants of these features based on different notions of "first approach to the destination." We require a few definitions for this and subsequent features. First, let *park_time* be the time when the user parks the vehicle and let *arrival_time* be the time when the user reaches the destination. Let R be a set of reference points along the trajectory, such as the first time the vehicle approaches within 500m, or 1000m, etc. Let $time_r$ be the time when the vehicle reaches reference point r.

Then for each $r \in R$:

$$vicinity\_to\_park\_time_r = park\_time - time_r,$$
$$vicinity\_to\_destination\_time_r = arrival\_time - time_r,$$

where the former captures the time spent between the reference point and parking, and the latter captures the time spent between the reference point and the arrival at the destination.

5.1.3 Deviation times. To estimate time spent in searching for parking, we aggregate the difference between when a user should have arrived at a destination if they simply drove to its access point (say, the front door), versus when they actually arrived, taking into account circling, parking, and walking. If many users show a large gap between these two times, we expect this to be a useful signal that parking is difficult. So we compute two sets of features.

Driving and arrival deviation. This feature captures the additional driving time caused by parking. We use Google Maps to perform a thought experiment: what if the user had driven from reference point r straight to the parking location? We define $drive\_estimate_r$ as the estimated time from r to the parking location, and then define driving deviation as the additional time needed to find the parking space, beyond what is required simply to drive there:

$$driving\_deviation_r = vicinity\_to\_park\_time_r - drive\_estimate_r.$$

Arrival deviation. Similarly, we consider the time when the user actually arrived at the destination (after circling, parking, and then walking) compared to the time it would have taken simply to drive

to the front door; this is intended to capture the total marginal time introduced by the need to park:

$$arrival\_deviation_r = vicinity\_to\_destination\_time_r - drive\_estimate_r.$$

Figure 5 shows weekly variation for deviation features for an example S2 cell in San Francisco.

5.1.4 An example. Figure 4 demonstrates a mock trajectory that approaches Union Square, San Francisco. With r as a reference point, based on traffic, Google Maps estimates that driving from r to the destination will require 2 minutes. However the driver circles for 15 minutes and finally parks in Chinatown followed by a 5 minute walk. In the figure, **r** is the reference point r, **D** is the destination Union Square, and **P** is the parking location. We have:

- $time_r$ = 12:00
- $park\_time$ = 12:15
- $destination\_arrival\_time$ = 12:20
- $vicinity\_to\_park\_time_r$ = 12:15 − 12:00 = 15min
- $vicinity\_to\_destination\_time_r$ = 12:20 − 12:00 = 20min
- $driving\_deviation_r$ = 15min − 2min = 13min
- $arrival\_deviation_r$ = 20min − 2min = 18min

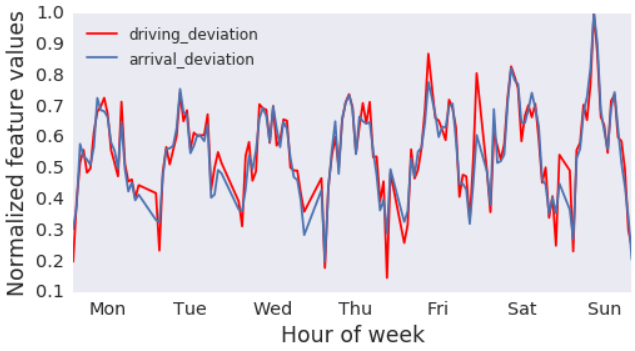

Figure 5: Feature variation throughout the week for an S2 cell in San Francisco.

5.2 Circling features

The features described in this section combat specific issues we saw in early versions of our system, particularly due to taxis and buses.

Taxi and rideshare dropoffs are very common in urban areas. In those cases, deviation times and distance features may be misleading, especially for destinations with a large fraction of arrivals via rideshare. In such cases, we see a large number of users drive to the front door of the destination and exit their vehicle. If we mistakenly believe these users are parking immediately in front of the destination, we may conclude that parking is easy. This bias is particularly damaging as difficult destinations will often draw larger fractions of rideshare visitors.

Buses may result in both false positives and false negatives. For cities with effective public transit systems, we may see signs of "easy" parking for destinations near bus stops, and signs of "hard" parking for destinations that are a longer walk from the closest bus stop. Figure 6 shows an example of a mock bus trajectory.

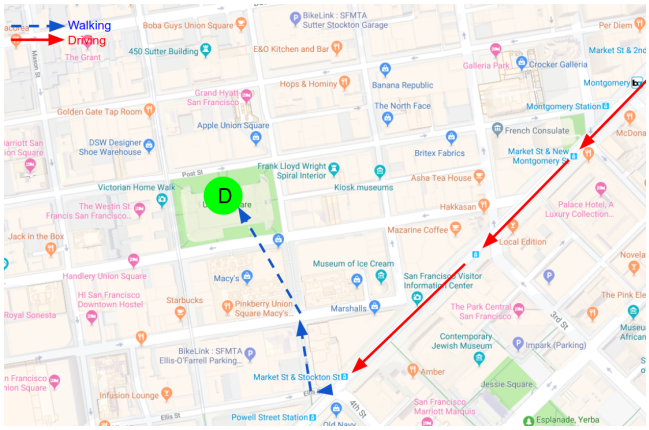

Figure 6: Mock trajectory that demonstrates a bus. (Not derived from user data.)

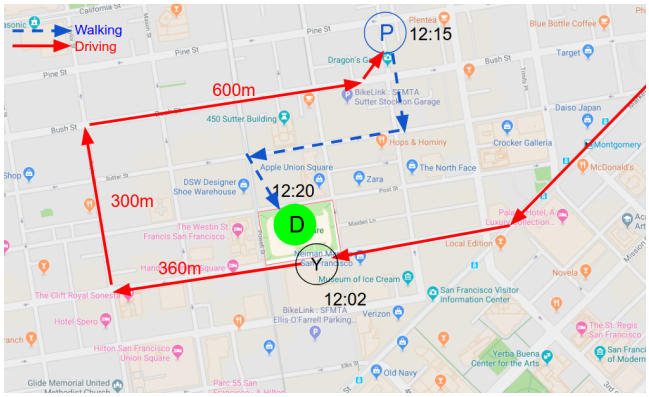

Figure 7: Mock trajectory that demonstrates a circling pattern (Not derived from user data.)

A strong pattern that reflects some difficulty in finding parking is *circling*: if we see a user driving closer and further, looping around the destination, then this trajectory might indicate some parking difficulty. Based on this, we define a set of features for circling behavior, capturing a higher-confidence set of difficult-to-park examples without attempting to distinguish easy parking from other ways to arrive at the front door. Consider a simple model in which the driver always drives directly towards the destination, and starts looking around for parking spots once they arrive. Let Y be the closest location to the destination before the driver starts looking for a parking spot; a proxy for Y is the first location report that shows an increase in distance to destination. We define $circling\_time$ as the time spent between Y and the parking location P, and $circling\_distance$ as the driving distance between Y and P.

Figure 7 shows an example. At 12:02pm, Y is the closest point to D when we start seeing the distance increasing. Then, $circling\_time$ = 10min, and $circling\_distance$ = 360m + 300m + 600m = 1260m.

5.2.1 Monotonicity. This feature captures a different aspect of circling. Given a sequence of location reports before arriving at a destination, in an idealized and easy to park situation, we expect the

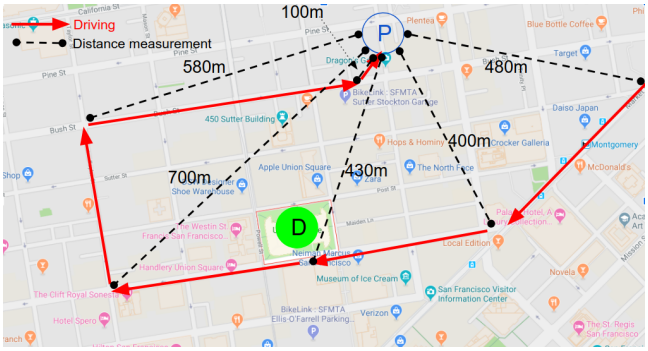

Figure 8: Mock trajectory that demonstrates monotonicity feature calculation. (Not derived from user data.)

sequence to be monotone non-increasing in terms of its distance to the destination. We capture this by defining *monotonicity* for a given sequence to be the ℓ_2 distance to the closest monotone sequence; we attach increasing weights to each point in the sequence closer to the destination. We can compute the monotonicity feature by using well-known algorithms for isotonic regression (e.g., see [3]).

Figure 8 shows an example in which there are six location reports before parking. The sequence of distances to the destination is (480, 400, 430, 700, 580, 100, 0) say with weights (1/7, 1/6, 1/5, 1/4, 1/3, 1/2, 1/1). The best monotone fit to the given sequence is (539,539,539,539,539,100,0), with an ℓ_2 error of 114.

5.3 Ambient features

Unlike the per-trajectory features, ambient features are computed using a custom aggregation of information from the trajectories in a spatiotemporal bucket. We describe two such classes of features.

5.3.1 Geo dispersion. Consider the trajectories with a destination in a particular spatiotemporal bucket B. If this bucket has easy parking, we expect that most vehicles would park nearby, ideally even in the same S2 cell. On the other hand, if the parking situation is hard, vehicles would conceivably park far away from the destination and in more diverse locations. Let $C_1, \ldots, C_n$ be all the S2 cells that contain at least one parking location of a trajectory in B. Let c_i be the number of trajectories in B parking in C_i. Finally, let $p_i = c_i / \sum_j c_j$ be the probability that a trajectory in B parks in C_i. We may now define two features capturing the geographic dispersion of parking locations for B.

Collision. This is the probability of two trajectories visiting the same destination parked in the same C_i.

$$\text{collision}(B) = \sum_{i=1}^{n} p_i^2.$$

Entropy. This is the lack of predictability of parking locations for the destination.

$$\text{entropy}(B) = \sum_{i=1}^{n} p_i \log(1/p_i).$$

5.3.2 Relative busyness. To capture times when an area is unusually busy, we include *relative busyness* features. We start by measuring the number $n_{\ell,t}$ of users for a spatiotemporal bucket $B = (\ell, t)$ defined by an S2 cell at location ℓ and an hour of week t. We then marginalize over time and compute the maximum, 90th, and 75th percentiles $n_\ell^{\max}$, $n_\ell^{90\%}$, and $n_\ell^{75\%}$ for every ℓ, and use these statistics to normalize the features across various t. This produces three features at each location ℓ and time t: $rel\_busyness\_max = n_{\ell,t}/n_\ell^{\max}$, $rel\_busyness\_90th = n_{\ell,t}/n_\ell^{90\%}$, and $rel\_busyness\_75th = n_{\ell,t}/n_\ell^{75\%}$.

6 LEARNING MODELS

For a fixed spatiotemporal bucket, we generate all the aggregated features (Section 5) and likewise generate the associated ground truth (Section 4.3); we treat this as a single training example.

In Google Maps, parking difficulty is displayed using 3 levels: `easy`, `medium`, and `limited`. Our model must therefore output one of these labels for each spatiotemporal bucket. Our ground truth data, however, is binary: either `easy` or `limited`. To optimize the model, we must determine how much the system is penalized for each type of mistake: how damaging is it, for instance, to report that parking is `medium` when in fact it is `limited`, compared to reporting `limited` when it is `easy`, and so on.

Based on discussion with domain experts, we define a 2x3 reward matrix to encode these values (Section 6.1). The loss function used in training is then derived from this matrix. Sections 6.3 and 6.4 describe specific models trained within this framework.

6.1 Reward matrix

The reward matrix R we use is the following.

		Predicted		
	R	`easy`	`medium`	`limited`
Actual	`easy`	1	-0.7	-3
	`limited`	-3	-0.7	1

Table 1: Reward matrix to convert from two-class ground truth to three-class predictions.

This reward matrix gives high penalty for incorrect predictions and allows the model to predict `medium` when it is unsure. Note that for balanced ground truth, the expected reward of always predicting `medium` (-0.7) is slightly higher than the expected reward of always predicting either `easy` or `limited` (-1.0). Hence, the model should employ `medium` until its level of certainty exceeds this difference.

6.2 Loss function

To optimize for overall reward, we define a loss function based on this matrix. The loss assumes that the model predicts a distribution over the three output labels, and is defined as the expected reward under the predicted distribution, given the actual label. For a training instance x with ground truth label y:

$$\text{loss}(x, y \mid \theta) = - \sum_{\hat{y} \in \{\texttt{easy}, \texttt{medium}, \texttt{limited}\}} p_\theta(\hat{y} \mid x) \cdot R(\hat{y}, y),$$

where $p_\theta(\hat{y}|x)$ is the probability the model assigns to output label $\hat{y}$ given features x and learned parameters θ, and $R(\hat{y}, y)$ is the reward for predicting output label $\hat{y}$ for ground truth label y.

6.3 Single-layer regression

We trained a standard single-layer multiclass regression ML model over the ground truth data and reward matrix. The objective is to minimize the loss function defined in Section 6.2. The simplicity of this model gives it a few advantages. First, its behavior is well understood, and it tends to be resilient to noise in the training data; this is a useful property when the data comes from crowd-sourcing a complicated response variable like difficulty of parking. Second, it makes the model interpretable and allows us to measure the contribution of each feature, as detailed in Sections 7.3 and 7.4.

6.4 Feedforward deep neural network model

We have also explored an alternate feedforward neural network model. With the same set of features used by the model of Section 6.3, we constructed a DNN model that has two hidden layers (20 and 10 hidden units, respectively), and applied the ReLU activation function to each layer. We used the loss defined in Section 6.2 and optimized using AdaGrad.

7 EVALUATION AND RESULTS

In this section we evaluate our models. Our goal is to understand the performance of our features and model architectures, and to study geographical generalization. Hence, we will report results for a model trained for the SF Bay area in Northern California, and we will evaluate this model's performance in other geographies, and also relative to ablated models trained on the same data.

7.1 Examples of model output

With our model in hand, we are able to generate an estimate of the difficulty of parking at any place and time. Figure 9 gives examples for the level 15 S2 cells of downtown Oakland, CA for a few different times of the week. On Mondays, parking is more difficult during the morning commute than at night. On Saturdays, parking is relatively easy in the morning but becomes hard at night.

7.2 Balanced normalized rewards (BNR)

Parking difficulty varies by city. Some cities are harder to park in than the others, and this results in a different distribution of answers corresponding to `easy` and `limited` in our survey responses. To compare model performance over different cities, which might have different label distributions, we report the rewards attained by our system (as per Table 1) after re-balancing the ground truth data distribution to an equal mass of `easy` and `limited` instances. For ease of interpretation, we also report average reward per evaluation instance. We refer to the resulting quantity as *Balanced Normalized Reward*, or *BNR*. It is defined as follows:

$$BNR = \frac{1}{2} \cdot \frac{\text{reward(easy)}}{\#\,\texttt{easy}\ \text{samples}} + \frac{1}{2} \cdot \frac{\text{reward(limited)}}{\#\,\texttt{limited}\ \text{samples}}.$$

7.2.1 Relative BNR. We treat the single-layer model of Section 6.3 cross-validated on the SF Bay area as our baseline model B, and define *relative BNR* of any model M as BNR(M) − BNR(B): the difference between BNR of the model being evaluated and the BNR of our baseline. All the evaluation results in following sections use relative BNR as the metric for model comparison. This measure reports how much additional reward per (balanced) instance model M attains compared to the baseline.

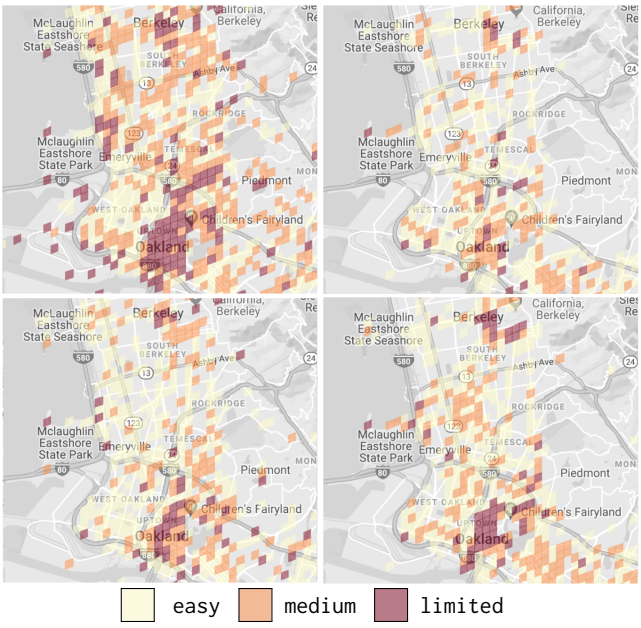

Figure 9: Output of our parking difficulty model for Downtown Oakland, CA. Top row: a typical Monday at 8am (left) and 9pm (right). Bottom row: the same times but on a typical Saturday.

7.3 Power of individual feature families

In this section we assess the power of each of the families of features described in Section 5. For this discussion, we use training data from SF Bay area. We first train our baseline model B on all the features. For each class of features F, we train a separate model M_F on the same data using just that class of features. Table 2 shows the relative BNR change of each model M_F, compared to model B using all the features. It should be read as follows: the first row shows that the parking distance features alone can attain just 0.069 less BNR than the full model M. Per this table, the parking distance, vicinity, and deviation time features appear to be the most powerful standalone features while monotonicity is the weakest feature.

7.4 Ablation analysis of feature families

In this section we study how much lift each feature family provides in attaining the performance of the full model. We generate this data as follows. In addition to model B, for each feature family F we train on the same data a new model $M_{(-F)}$ using all features except for those in F. We then report the relative BNR change of each model $M_{(-F)}$ compared as usual to baseline model B. Table 3 shows the results. The table should be read as follows: the first row

Feature class	Relative BNR
Parking distance	-0.069
Vicinity times	-0.011
Deviation times	-0.079
Monotonicity	-0.369
Circling	-0.104
Ambient	-0.115

Table 2: Relative BNR (Section 7.2.1) for models trained using only feature classes compared to training with all features, for SF Bay area.

of the table shows that, without the parking distance features, the model can only attain 0.018 less BNR than the full model.

Distance, vicinity, and deviation times are the most powerful standalone features classes, but they capture redundant information about parking search time, so there is no dramatic impact from ablating any one class of features.

Ablated feature class	Relative BNR
Parking distance	-0.018
Vicinity times	-0.024
Deviation times	-0.014
Monotonicity	-0.004
Circling	-0.004
Ambient	-0.003

Table 3: Relative BNR (Section 7.2.1) for models trained using all but that feature class compared to training with all features, for SF Bay area.

7.5 DNN performance and power to generalize

We now consider the two-layer feedforward DNN model architecture described in Section 6.4. We again train this model using data from SF Bay area, and evaluate it against the same data. In addition, we report the results of applying both the baseline single-layer model B and the two-layer model M against evaluation data from four other cities worldwide. This gives us some insights regarding how the patterns of parking difficulty in one city generalize when applied to a different city. The results are shown in Table 4. The table shows the relative BNR of model B and the DNN model, and in column Δ, shows the improvement in BNR from using the DNN model compared to the single-layer regression model.

From this table, we can observe that the single-layer and two-layer models perform similarly in the SF Bay area. However, the DNN generalizes better than the single-layer model to new cities.

7.6 Cross-city generalization vs. local training

In this section we again consider the performance of the two-layer DNN model trained in the SF Bay area transferred to apply to other cities. However we now compare the results against a locally-trained model using data from the target city. Table 5 shows the relative BNR of the DNN model trained on the SF Bay area compared to the relative BNR of the same architecture trained on data from the target city.

Evaluation city	Relative BNR		
	Model B	DNN	Δ
SF Bay area	0.000	0.002	0.002
Los Angeles, CA	-0.202	-0.105	0.097
Manchester, UK	-0.242	-0.124	0.118
Pune, India	-0.395	-0.298	0.097
Sao Paulo, Brazil	-0.432	-0.329	0.103

Table 4: Relative BNR (Section 7.2.1) of the model B and DNN model M trained on SF Bay area evaluated in four other cities.

Evaluation city	Relative BNR		
	Trained on SF Bay area	Trained locally	Δ
Los Angeles, CA	-0.175	-0.105	0.070
Manchester, UK	-0.197	-0.124	0.073
Pune, India	-0.358	-0.298	0.060
Sao Paulo, Brazil	-0.425	-0.329	0.096

Table 5: Per-city relative BNR (Section 7.2.1) of the DNN model trained on the SF Bay area compared to the DNN model trained on the target city's local ground truth. Column Δ shows the improvement that results from locally training the model.

The results are consistent with other cities we have evaluated: a model trained on data from a North American metro region tends to perform reasonably well across North America, South America, and Europe. In general, locally training a model tends to yield between 0.06 and 0.1 improvement in BNR compared to transferring the model from SF Bay area.

8 CONCLUSIONS AND FUTURE WORK

In sum, we explored a set of feature classes that reflect parking difficulty of a given region and produced subjective parking difficulty estimates. Experiments on Google Maps on a few regions found that exposure to a parking difficulty badge, regardless of `easy`, `medium`, or `limited` parking led to a statistically significant 4% increase in queries for other travel modes like transit and taxi. This and overall model performance metrics led us to launch parking difficulty predictions in 55 cities worldwide.

While we use features from aggregated historical examples to predict aggregated survey results, they may come from a diverse set of users with different individual experiences. To account for that it would be interesting to train and predict for each survey response and then aggregate them to produce the final outputs.

The parking situation changes based on seasons, structural and functional changes in parking spots, events etc. It would be interesting to take these into account while aggregating features and ground truth and to produce real-time estimates reflecting the current situation.

REFERENCES

[1] Walaa Alajali, Sheng Wen, and Wanlei Zhou. 2017. On-Street Car Parking Prediction in Smart City: A Multi-source Data Analysis in Sensor-Cloud Environment. In *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*. 641–652.

[2] 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.

[3] Michael J Best and Nilotpal Chakravarti. 1990. Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming* 47, 1-3 (1990), 425–439.

[4] Fabian Bock and Monika Sester. 2016. Improving Parking Availability Maps using Information from Nearby Roads. *Transportation Research Procedia* 19 (2016), 207–214.

[5] Jordan Cazamias and Martina Marek. 2016. *Parking Space Classification using Convoluional Neural Networks.* Technical Report. Stanford University.

[6] Michal Chalamish, David Sarne, and Raz Lin. 2013. Enhancing parking simulations using peer-designed agents. *IEEE Transactions on Intelligent Transportation Systems* 14, 1 (2013), 492–498.

[7] Jatuporn Chinrungrueng, Udomporn Sunantachaikul, and Satien Triamlumlerd. 2007. Smart parking: An application of optical wireless sensor network. In *SAINT Workshops*. 66–66.

[8] Vladimir Coric and Marco Gruteser. 2013. Crowdsensing maps of on-street parking spaces. In *DCOSS*. 115–122.

[9] Paulo RL De Almeida, Luiz S Oliveira, Alceu S Britto Jr, Eunelson J Silva Jr, and Alessandro L Koerich. 2015. PKLot–A robust dataset for parking lot classification. *Expert Systems with Applications* 42, 11 (2015), 4937–4949.

[10] Karel Dieussaert, Koen Aerts, Thérèse Steenberghen, Sven Maerivoet, and Karel Spitaels. 2009. SUSTAPARK: an agent-based model for simulating parking search. In *AGILE International Conference on Geographic Information Science, Hannover*. 1–11.

[11] Nemanja Djuric, Mihajlo Grbovic, and Slobodan Vucetic. 2016. Parkassistant: An algorithm for guiding a car to a parking spot. In *Transportation Research Board 95th Annual Meeting*. 16–24.

[12] Radovan Fusek, Karel Mozdřeň, Milan Šurkala, and Eduard Sojka. 2013. AdaBoost for parking lot occupation detection. In *CORES*. 681–690.

[13] Andrea Gemma and Giuseppe Vaccaro. 2019. Improving the Assessment of Transport External Costs Using FCD Data. In *CSUM*. 131.

[14] 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.

[15] Mareike Hedderich, Ulrich Fastenrath, Gordon Isaac, and Klaus Bogenberger. 2017. Adapting the A* algorithm for park spot routing. *Transportation Research Procedia* 27 (2017), 1066–1073.

[16] Ching-Chun Huang and Sheng-Jyh Wang. 2010. A hierarchical bayesian generation framework for vacant parking space detection. *IEEE Transactions on Circuits and Systems for Video Technology* 20, 12 (2010), 1770–1785.

[17] Hidetomo Ichihashi, Tatsuya Katada, Makoto Fujiyoshi, Akira Notsu, and Katsuhiro Honda. 2010. Improvement in the performance of camera based vehicle detector for parking lot. In *FUZZ*. 1–7.

[18] INRIX. 2017. Searching for Parking Costs Americans $73 Billion a Year. http://inrix.com/press-releases/parking-pain-us/

[19] Yanjie Ji, Dounan Tang, Phil Blythe, Weihong Guo, and Wei Wang. 2014. Short-term forecasting of available parking space using wavelet neural network model. *IET Intelligent Transport Systems* 9, 2 (2014), 202–209.

[20] Andrew Kirmse, Tushar Udeshi, Pablo Bellver, and Jim Shuma. 2011. Extracting patterns from location history. In *GIS*. 397–400.

[21] Andreas Klappenecker, Hyunyoung Lee, and Jennifer L Welch. 2014. Finding available parking spaces made easy. *Ad Hoc Networks* 12 (2014), 243–249.

[22] Shixu Liu, Hongzhi Guan, Hai Yan, and Huanhuan Yin. 2010. Unoccupied parking space prediction of chaotic time series. In *ICCTP 2010: Integrated Transportation Systems: Green, Intelligent, Reliable*. 2122–2131.

[23] L Mannini, E Cipriani, U Crisalli, A Gemma, and G Vaccaro. 2017. On-Street Parking Search Time Estimation Using FCD Data. *Transportation Research Procedia* 27 (2017), 929–936.

[24] 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.

[25] 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 *MobiSys*. 123–136.

[26] Lara Montini, Andreas Horni, Nadine Rieser-Schüssler, and Kay W Axhausen. 2012. Searching for parking in GPS data. *Working paper/Transport and Spatial Planning* 780 (2012), 1–25.

[27] Tooraj Rajabioun and Petros A Ioannou. 2015. On-street and off-street parking availability prediction using multivariate spatiotemporal models. *IEEE Transactions on Intelligent Transportation Systems* 16, 5 (2015), 2913–2924.

[28] Felix Richter, Sergio Di Martino, and Dirk C Mattfeld. 2014. Temporal and spatial clustering for a parking prediction service. In *ICTAI*. 278–282.

[29] Yuecheng Rong, Zhimian Xu, Ruibo Yan, and Xu Ma. 2018. Du-Parking: Spatio-Temporal Big Data Tells You Realtime Parking Availability. In *KDD*. 646–654.

[30] Matthias R Schmid, Savas Ates, Jürgen Dickmann, Felix von Hundelshausen, and H-J Wuensche. 2011. Parking space detection with hierarchical dynamic occupancy grids. In *IV*. 254–259.

[31] Jae Kyu Suhr and Ho Gi Jung. 2012. Fully-automatic recognition of various parking slot markings in Around View Monitor (AVM) image sequences. In *ITSC*. 1294–1299.

[32] Arbi Tamrazian, Zhen Qian, and Ram Rajagopal. 2015. Where Is My Parking Spot? Online and Offline Prediction of Time-Varying Parking Occupancy. *Transportation Research Record: Journal of the Transportation Research Board* 2489 (2015), 77–85.

[33] Tim Tiedemann, Thomas Vögele, Mario Michael Krell, Jan Hendrik Metzen, and Frank Kirchner. 2015. Concept of a Data Thread Based Parking Space Occupancy Prediction in a Berlin Pilot Region.

[34] Nicholas True. 2007. Vacant parking space detection in static images. http://cseweb.ucsd.edu/classes/wi07/cse190-a/reports/ntrue.pdf

[35] Marc Tschentscher and Marcel Neuhausen. 2012. Video-based parking space detection. In *Proceedings of the Forum Bauinformatik*. Citeseer, 159–166.

[36] Eleni I Vlahogianni, Konstantinos Kepaptsoglou, Vassileios Tsetsos, and Matthew G Karlaftis. 2016. A real-time parking prediction system for smart cities. *Journal of Intelligent Transportation Systems* 20, 2 (2016), 192–204.

[37] Rashid Waraich and Kay Axhausen. 2012. Agent-based parking choice model. *Transportation Research Record: Journal of the Transportation Research Board* 2319 (2012), 39–46.

[38] Rashid A Waraich, Christoph Dobler, and Kay W Axhausen. 2012. Modelling parking search behaviour with an agent-based approach. In *IABTR*.

[39] Qi Wu and Yi Zhang. 2006. Parking lots space detection. https://www.cs.cmu.edu/~epxing/Class/10701-06f/project-reports/wu_zhang.pdf

[40] Bo Xu, Ouri Wolfson, Jie Yang, Leon Stenneth, S Yu Philip, and Peter C Nelson. 2013. Real-time street parking availability estimation. In *MDM*, Vol. 1. 16–25.

[41] Shuguan Yang, Wei Ma, Xidong Pi, and Sean Qian. 2019. *A deep learning approach to real-time parking occupancy prediction in spatio-termporal networks incorporating multiple spatio-temporal data sources.* Technical Report. arXiv:1901.06758.

[42] Shuguan Yang and Zhen Sean Qian. 2017. Turning meter transactions data into occupancy and payment behavioral information for on-street parking. *Transportation Research Part C: Emerging Technologies* 78 (2017), 165–182.

[43] William Young and Tan Yan Weng. 2005. Data and parking simulation models. In *Simulation approaches in transportation analysis*. Springer, 235–267.

[44] Yanxu Zheng, Sutharshan Rajasegarar, and Christopher Leckie. 2015. Parking availability prediction for sensor-enabled car parks in smart cities. In *ISSNIP*. 1–6.

[45] Ali Ziat, Bertrand Leroy, Nicolas Baskiotis, and Ludovic Denoyer. 2016. Joint prediction of road-traffic and parking occupancy over a city with representation learning. In *ITSC*. 725–730.

[46] Onno Zoeter, Christopher R Dance, Mihajlo Grbovic, Shengbo Guo, and Guillaume Bouchard. 2012. A general noise resolution model for parking occupancy sensors. In *19th ITS World CongressERTICO-ITS EuropeEuropean CommissionITS AmericaITS Asia-Pacific*.