ParkUs 2.0: Automated Cruise Detection for Parking Availability Inference

Michael Jones[1,2], Aftab Khan*[1], Parag Kulkarni[1], Pietro Carnelli[1,2], Mahesh Sooriyabandara[1]
[1]Toshiba Research Europe Limited, 32 Queen Square, Bristol, BS1 4ND, UK
[2]University of Bristol, Bristol, UK
mj14472@my.bristol.ac.uk,{aftab.khan,parag.kulkarni,pietro.carnelli,mahesh}@toshiba-trel.com

ABSTRACT

Recent studies show that a key contributor to congestion and increased CO_2 emissions within cities are drivers searching (or cruising) to find a vacant on-street parking space. It has been shown that approximately (depending on the city) 20-30% of vehicles in congested urban areas were cruising to find a parking space with a parking search time varying in the order of several minutes. In the city of Bristol alone, we have shown, using our collected trip and publicly available census data that over 790 metric tons of CO_2 is generated every year due to cruising. At a total cost of £368,000 (US$467,000) in terms of fuel wasted. The solution, described in this paper, aims to reduce parking search times using our automated real-time parking system called ParkUs 2.0. Our proposed method leverages sensor and location data collected from smartphones (carried by drivers), uses machine learning (classification) to detect cruising behaviour, automatically annotates parking availability on road segments based on the classified data and displays this information as a heatmap of parking availability information on the user's smartphone. This is the first such attempt to automatically detect cruising to the best of our knowledge. Evaluation through controlled trials with volunteer participants highlights the potential of our novel approach as we are able to detect cruising with an accuracy of 81%.

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**;

KEYWORDS

Activity recognition, smart parking, cruise detection, mobile sensing, automated annotation

MobiQuitous 2017, November 7–10, 2017, Melbourne, VIC, Australia

ACM ISBN 978-1-4503-5368-7/17/11...$15.00
https://doi.org/10.1145/3144457.3144495

1 INTRODUCTION & RELATED WORK

Since 1950 the proportion of people living in cities has been increasing. A 2014 UN urbanisation report estimates that 54% of the world's population live in urban areas. This proportion is expected to rise to 66% by 2050 [19]. As cities continue to grow in population and overall size, further demand is placed on transportation infrastructure such as road networks. Vehicular congestion is a major problem in most cities in the world and is likely to get worse in light of the aforementioned trend.

Donald Shoup pioneered the research into the behaviour and economics of drivers cruising for vacant on-street parking spaces. His research summarises the results of a collection of studies into cruising conducted throughout the 20^{th} century. These studies indicate that, on average, 20%-30% of vehicles in congested urban areas are cruising for a parking space. Even a relatively short average cruising time of 3.3 minutes (as experienced in Westwood Village, LA) per driver per parking activity can, over the course of a year, generate 730 metric tons of CO_2 emissions [17].

Providing users with real-time parking availability is a widely-researched problem [9]. However, most of this previous work presents solutions from one of two main categories. First, those that rely solely on parking availability information manually inputted by the user, such as Alphabet's now-discontinued OpenSpot project, and ParkJam [8]. Secondly, there are those that use physical infrastructure, such as sensors in parking spots [20] and CCTV footage of parking spots. These require huge upfront investments for development, installation and ongoing costs for maintenance. For example, the SFPark project in San Francisco cost $23M to equip 6,000 (less than 25% of) on-street parking spaces with sensors connected to the Internet. A thorough overview of research into existing parking availability solutions is provided by Lin *et al.* [9].

As a consequence, recent research has focused on finding ways to monitor in real-time parking occupancy without incurring huge physical infrastructure set up and maintenance costs. Our previous effort, ParkUs 1.0 [1], attempted to achieve this by automatically detecting parking and unparking events using a driver's smartphone accelerometer and magnetometer sensor data (no user input or separate infrastructure installation was required). Although more accurate and energy efficient than it's competitors such as PhonePark, Park Here and Park Sense [10, 14, 18] respectively, ParkUs 1.0 was not able to detect driver cruising behaviour. Detecting

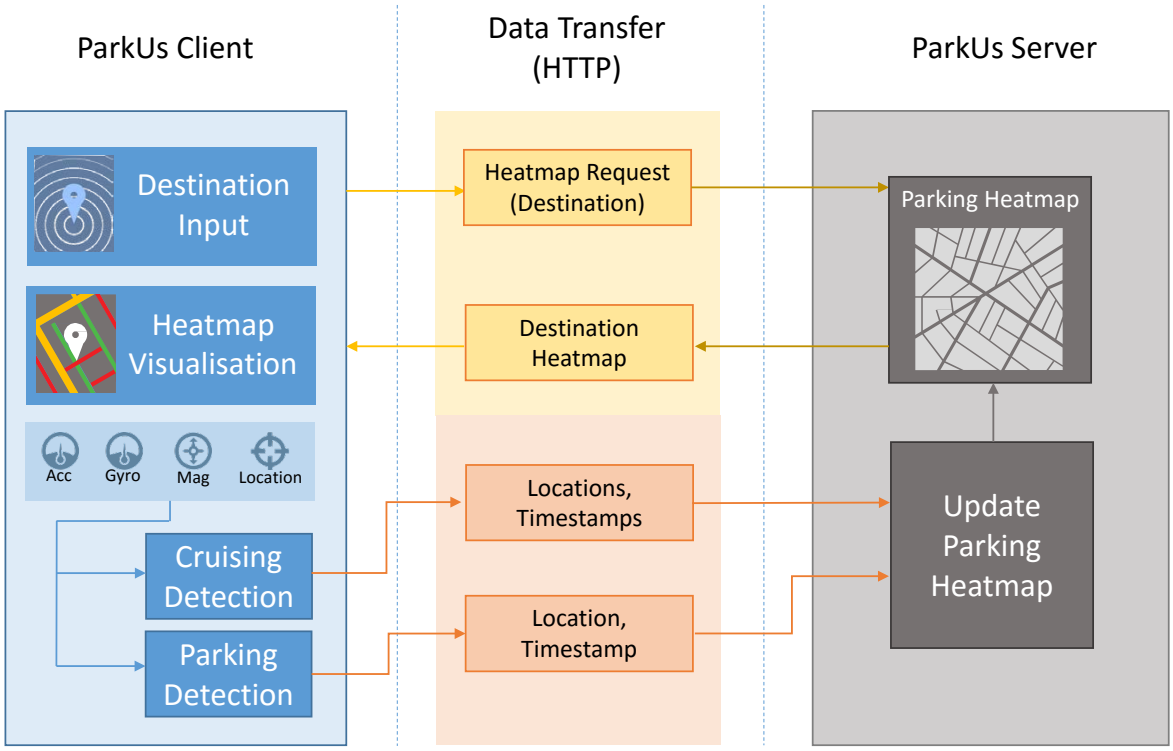

Figure 1: ParkUs 2.0 system overview

cruising combined with parking activity has a major advantage when it comes to providing real-time parking availability information. By detecting solely parking and un-parking activities the system only collects one data-point i.e. where and when the driver parked and un-parked, thus yielding little information about the parking availability on nearby streets. On the other hand, detecting and monitoring cruising behaviour allows us to infer the availability of parking spaces on multiple streets. As it is assumed that a driver would park as soon as they find a vacant spot near to their desired destination. Therefore we can infer with reasonable certainty that there were no vacant parking spaces along those roads or parking bays that the driver searched shortly before they parked. Consequently we can now advise other users looking to park in the same area not to waste time by (re-)searching those streets or car park, as they are most likely full. Instead we can potentially suggest a better search strategy or in extreme cases to use public transport instead.

An important aspect of ascertaining parking availability is to reliably identify and eliminate areas where parking is not available. It is well-accepted that detecting a cruising event is a proxy for lack of parking. However, detecting that a vehicle is cruising is not a trivial problem. Prior work on vehicle cruise detection [3] mainly relies on two approaches for identifying a cruise event. The first approach harnesses GPS data for detecting deviation from the planned route to the destination. Whilst this is a good first step, it falls short along a number of dimensions. Deviation from the planned route can happen at any time and any number of times during the journey which may yield multiple false positives. The second approach in [3] uses trajectory-based parking search behaviours. There are practical limitations as this requires manually defining and labeling these behaviours in order to detect parking search. For instance, the data has to be annotated to identify parking search behaviours only after the journey has been completed, as the driver is not legally allowed to use a phone whilst driving.

Solutions such as the novel ParkUs 2.0 system presented in this paper have the advantage of costing very little to implement, whilst being able to be used anywhere in the world. Our cruise detection system relies on the principle of detecting a significant local minimum in the GPS trace with respect to distance from the destination. In addition to GPS data, other sensing data from the driver's smartphone such as accelerometer, gyroscope and magnetometer were also collected. Features extracted from such sensor data can help in training models and effectively detecting when the vehicle does and doesn't cruise.

However, making parking availability information in the desired area available in real-time based on information gathered from other drivers in the area is a challenging problem. In short, the success of such solutions depends on the uptake;

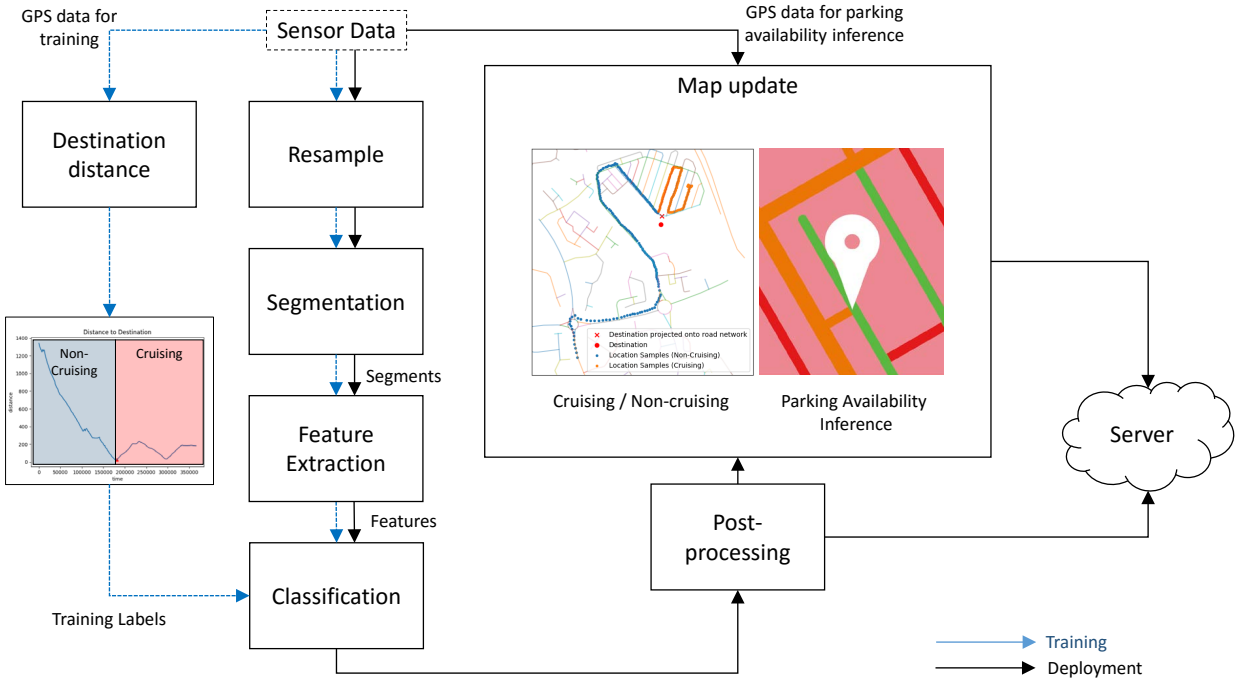

Figure 2: Data flow pipeline for training and deployment

the more people that use ParkUs 2.0 within a given area or city, the better the accuracy.

Overall, the ParkUs 2.0 system is practical from an implementation perspective, has been realised as a smartphone app augmented with a backend server system and has been trialled through controlled experiments in the city of Bristol, UK. Findings from this evaluation are elaborated upon in the second half of the paper. In summary, this paper makes the following key contributions:

- An automated cruising detection method, the first such attempt to the best of our knowledge.
- A feature representation framework tailored for cruising detection.
- A novel method for automated cruise annotation using GPS data. Alternative methods are either not legally possible, e.g. cannot annotate when driving or are cumbersome and error-prone such as annotating retrospectively once parking is done.
- A parking availability inference system (end-to-end encompassing the app and the backend system), and
- Findings from a real world trial with users in the city of Bristol using a dataset with 41 journeys.

2 SYSTEM ARCHITECTURE

The broader goal of ParkUs 2.0 is to reduce congestion resulting from a search for a vacant parking space. This should help to not only reduce unnecessary CO_2 and NO_2 emissions but also save time and money for the driver.

2.1 ParkUs 2.0 Design

ParkUs 2.0 is a crowd-sensing based solution which has been realised as an Android smartphone app augmented with an intelligent backend system (see Fig. 1). The app collects location and sensor data from the drivers, uses a machine learning model to pick up patterns relating to "vehicle cruising behaviour" and uses this in addition to input solicited from the vehicle driver (app user) once they have parked to update probabilistic parking availability information on the backend server. When a user enters their desired destination address or postcode before beginning the journey, the app displays a heatmap of parking availability in the destination area within a user specified radius. The backend server keeps updating the road-by-road parking information as new data is received from other users in the destination area and sends an updated heatmap periodically to the smartphone app thereby resulting in a refresh of parking availability information.

Whilst making vehicle parking information available on a smartphone is not new, the novel aspect of the ParkUs 2.0 system is automatic cruise detection using sensor and location data and a parking availability inference system which requires minimal user interaction. To the authors' knowledge this is the very first approach using machine learning to detect cruising behaviour. The benefit of detecting cruising in the destination area is that this could be considered a proxy for lack of availability of parking which can be useful to update the probabilistic parking availability on the roads where the driver is found to be cruising.

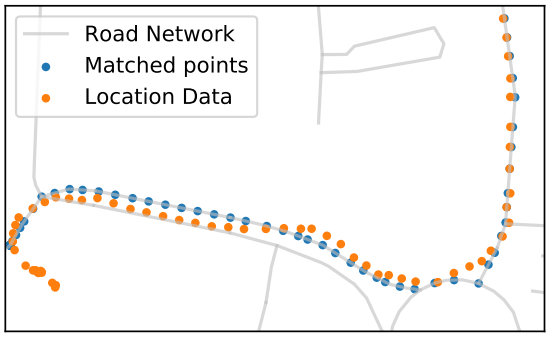

Figure 3: Example of the raw location points and the map-matched points from the algorithm.

2.2 Data Annotation and Cruising Detection

Automatic cruising detection is a key component of the ParkUs 2.0 system, as it can potentially allow parking availability information to be inferred without user input, based on the principle that the roads on which cruising is detected are likely to contain no vacant parking spaces. For our data collection trial, volunteers were asked to record their trips using our data collection application in order to provide us with a ground truth.

To achieve cruising detection, a method for automatically detecting cruising based on location data is presented. This method is then used to automatically label sensor data – accelerometer, magnetometer, gyroscope, GPS speed, and GPS bearing – as cruising or non-cruising.

Figure 2 shows the flow of data from the sensors on the smartphone to the set of labeled feature vectors to train a binary classifier. It also shows the flow of data once the classifier has been deployed, and the distance-based labeling is no longer needed.

The advantage of the proposed method is that data can be automatically labeled. To start with sensing data is by default labeled as "not cruising". As soon as the significant minimum point is crossed, data is automatically labeled as "cruising", an example of this is shown in Figure 5. Such automatic annotation (or labeling) of data has tremendous advantages since it reduces the time required to annotate the data (compared to manual annotation) and the time spent training the model. Furthermore, the driver is not required to interact with the phone during driving, thus ensuring compliance with the law.

2.3 Map-Matching

To accurately determine which streets and or which rows of a car park a driver searched before finding a vacant parking space a map-matching algorithm was developed and used to pre-process our GPS data. Figure 3 highlights the problem; as the data points collected during our trial do not always follow the road layout. Although on average, the GPS points were not too far off from the ground truth route used by the driver, a map-matching algorithm was developed and applied to all GPS data points collected.

We formulate the map-matching problem as a Hidden Markov Model (HMM) where the hidden states are the points precisely on the road network (data obtained from Open Street Map), and the observations are the actual values reported by the GPS sensor (embedded in the driver's smartphone). Thus, our aim is to identify the most likely actual position of the vehicle given the 'noisy' GPS data and the precise OSM road network way points. We use an adaptation of Newson's algorithm [11] with the threshold at which the observation points are deemed too unlikely to correspond to a hidden state to be of use reduced to 100m. This was done given that after some initial testing, the reported locations were mostly within $5 - 10$m of the true location (thus reducing the number of candidate points and the subsequent computation required). Another adaptation was to use points as hidden states instead of entire road segments. Since the candidate points were restricted to a threshold radius of each observation, it follows that each observation will usually have a different set of candidate points. The candidate points were chosen for each road segment within the radius as the closest point on that road segment to the observation.

Finally the Viterbi algorithm was used to find the most likely path taken by the driver given our collected set of GPS data points per trip. The Viterbi algorithm calculates the Viterbi path, i.e. the most likely sequence of a HMM given the set of observations. This can be expressed as a dynamic programming problem, starting from the base case and building up, storing the calculated likelihood values as a table thus avoiding recalculation. Finally, to find the actual path taken by the driver, during each step of the Viterbi algorithm, the previous candidate point that resulted in the maximum probability is recorded.

3 METHODOLOGY

3.1 Training Data Collection

Sensor data from a total of 41 journeys was collected from 5 volunteers over a period of several weeks between March and June 2017. Accelerometer and gyroscope data was sampled at a rate of 25 Hz, while magnetometer data was sampled at a rate of 5 Hz. Location data from Alphabet's Fused Location Provider [1] was sampled at 1 Hz.

For each journey, volunteers were asked to record their desired destination before setting off in their vehicle. The smartphone sensor data (such as accelerometer, gyroscope and magnetometer) were then collected along with GPS data for the entire journey until the user parked at their destination and pressed a "parking done" button in the application. Sensor and location data were then uploaded to a server for further processing (detailed in the following section). The app was designed such that a user does not have to use their smart-phone whilst driving, a common motor regulation in most countries. Furthermore, volunteers were not restricted

[1] https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderApi

Table 1: Summary of the dataset used to develop the cruising detection method.

User ID	Device(s)	Cruising Journeys	Non-Cruising Journeys	Mean Journey Time (mins)	Mean Cruise Time (mins)
1	Samsung Galaxy Note 3	11	0	9.84	2.35
2	Samsung Galaxy S3, LG Nexus 5	13	1	9.82	2.51
3	Samsung Galaxy S4 Mini	3	0	5.60	2.31
4	OnePlus 3	10	1	5.58	1.92
5	Samsung Galaxy S5	0	2	24.9	N/A

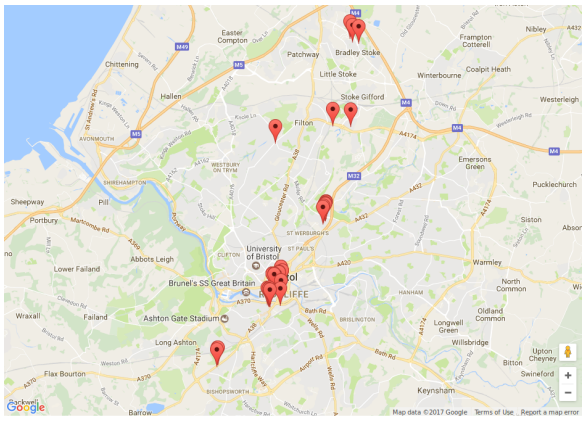

Figure 4: The destinations of the journeys made in Bristol during our trial data collection.

in placement or orientation of their smartphones during the data collection phase.

A summary of the dataset is provided in Table 1. With the aim of maximising the generality of the cruising classifier, the data was collected at several locations across the city of Bristol, UK. Figure 4 shows the destinations of the journeys in the dataset.

3.2 Journey Annotation using Distance to Destination

Manually identifying and labeling the starting time of cruising for each journey would be a time-consuming and error-prone task. The method described in this section automates the process of determining this time stamp by using the observation that, in a typical journey in which cruising occurs, the distance to the destination will decrease until it hits a local minimum. At which point the distance increases again as the driver starts moving further away from their destination in the search for parking. This minimum might be caused by the driver passing by their destination and observing that no parking is available, or it might be caused by a circular route or S-shaped curve as might be observed in a car park.

The distance along the road network is used to calculate the distance to the destination rather than simply the straight-line distance. The algorithm to find the time at which the driver begins cruising is as follows:

(1) For each location point within a certain radius of the destination, calculate the distance along the road network to the destination using Dijkstra's algorithm.
(2) Apply a Savitzky-Golay filter [15] to smooth this curve and remove small spikes/minima caused by imperfections in the road network representation.
(3) Find all the valleys in the function by taking the first discrete difference and finding sign changes. In order to only find valleys that are significant, the difference in y-value between the top and bottom of the valley was calculated. This difference is referred to the *prominence*, and the first valley with the prominence above a certain value is considered the start of cruising, provided that the cruising lasts for a minimum time.

Figure 5 shows an example of a location trace that involves cruising in a car park, along with the distance-to-destination curve after filtering, with the minimum detected by finding the first valley with significant prominence.

3.3 Feature Engineering: Sub-window Representation

Typically, overlapping time windows are used to extract features from real-time sensor data [7]. We propose a modified technique, still based on time window segmentation. This is motivated by the observation that, in comparison to differentiating activities such as walking and driving wherein the movement patterns in a short time window are noticeably different (e.g. the inherent oscillation in walking [1]), cruising and non-cruising are more subtle behaviours to differentiate using single, short, independent time windows.

The method attempts to introduce a notion of dependence between adjacent time windows, by considering each window in the baseline approach as a *subwindow*, and calculating a feature vector for a window (consisting of several adjacent subwindows) by concatenating the features extracted from each subwindow. A visual explanation of this method is shown in Figure 6 for overlapping subwindows.

More formally, a standard set of features and labels can be represented as:

$$\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{n} \in \mathbb{R}^{d=n\times(m+1)} \tag{1}$$

where x_i represents features for a given window i and y_i represents the associated label. In total there are n windows and m features with 1 label column. For the proposed sub-window approach, as seen in Figure 6, number of total windows is reduced but the feature dimensionality increases:

$$\hat{\mathcal{S}} = \{\{x_i^j\}_{i=1}^{k}, M_o(\{y_i^j\}_{i=1}^{k})\}_{j=1}^{(n-1)/k} \in \mathbb{R}^{d=((n-1)/k)\times((k\times m)+1)} \tag{2}$$

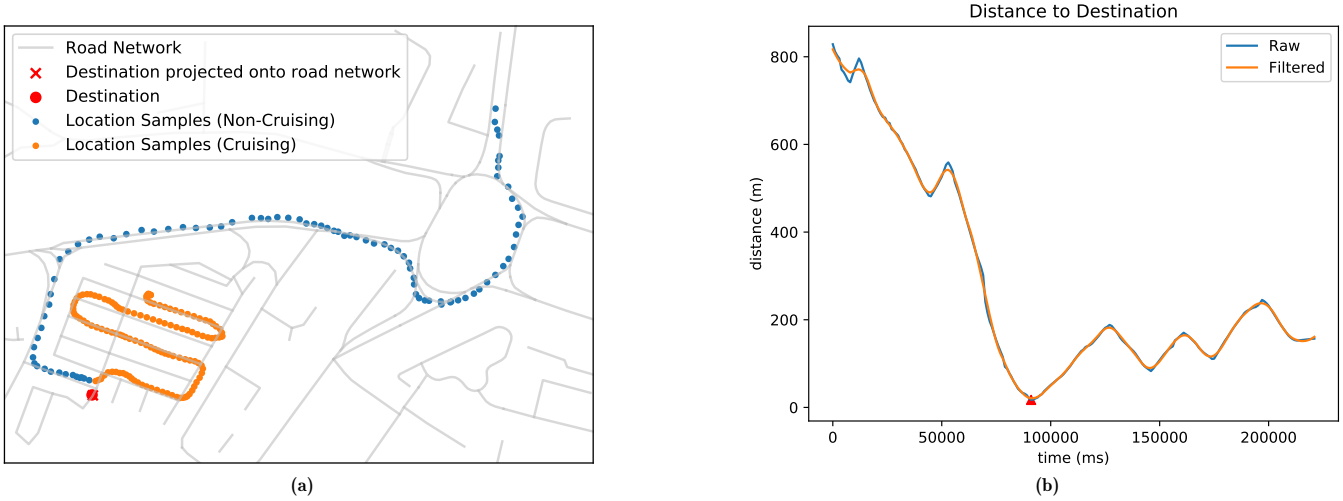

Figure 5: Location samples plotted on map (a), and corresponding distance-to-destination curve for a journey that involves cruising (b). The red triangle in the figure on the right indicates the start of cruising identified by our method.

where k represents the number of concatenated sub-windows, and $M_o(\cdots)$ is the mode function. Labels are associated with a given window using the mode of all labels in the sub-windows.

The features extracted from the time windows of each sensor were based upon those used in previous research into transportation mode detection using accelerometer data [7]. In particular, for each time series, the following categories of features were extracted:

- Statistical (μ, σ, range, interquartile range, ECDF[5])
- Frequency (Discrete Fourier Transform (DFT))
- Discrete integral

3.4 Feature Selection

We extract a total of $m = 188$ features. Due to the high dimensionality of the feature vectors, and the fact that these features were mostly standard statistical features (or features used in previous work), some features were likely to be irrelevant for cruising detection and some had a low variance throughout the dataset. To reduce this problem, a k-best feature selection method was applied [4], which ranked the features according to their analysis of variance (ANOVA) F-value between the label and feature. The k-best number of features (optimised along with the classifier parameters) were then used for each of the classification mechanisms described in the following section. The total number of selected features over a window reduce to f_s^w (f_s^k over a sub-window) optimised separately for classification.

3.5 Classification Mechanism

In order to evaluate the proposed system (using both traditional feature representation and our proposed method), we utilised three classification strategies. In particular, we used two kinds of supervised classifiers: *a)* discriminative models: Decision Trees and Support Vector Machines, and an *b)* Instance-based learning model i.e., k-Nearest Neighbors. For SVMs, we used a Radial Basis Function (RBF) Kernel; for which cost (C) and gamma (γ) parameters were optimized using a grid-search procedure [16]. The value of k was also optimized for the k-NN model. For all of these methods, ground truth labels (as discussed in Section 3.2) were used for training.

3.6 Post-Processing

The nature of cruising is such that a vehicle is either cruising or not cruising for fairly significant periods of time – i.e., longer than the window sizes we are going to consider. As a result, in order to reduce the effect of the classifier producing short "gaps" of non-cruising during true cruising, or cruising during non-cruising, the output of the classifier was filtered. The method used was a mean filter applied to the series of 0s and 1s representing the classifier output, followed by a round to the nearest integer (0) or (1).

3.7 Evaluation Metrics

In order to alleviate the potential bias introduced by cross validation schemes such as in a standard k-fold cross validation scheme, a leave-one-user-out cross validation technique was applied during all steps of training and evaluation of the classifiers [6]. For each held-out user's dataset, the classifier was trained on the whole journey (labeled automatically). It was then evaluated on the portion of the user's journey within 400m of the destination. For objective measure of the performance of our classifiers, we used class-weighted F-1 scores. An F-1 score is the harmonic mean of *precision* – which can be interpreted as a measure of the classifier's ability to not label as positive a sample that is negative – and *recall* – which can be interpreted as a measure of the

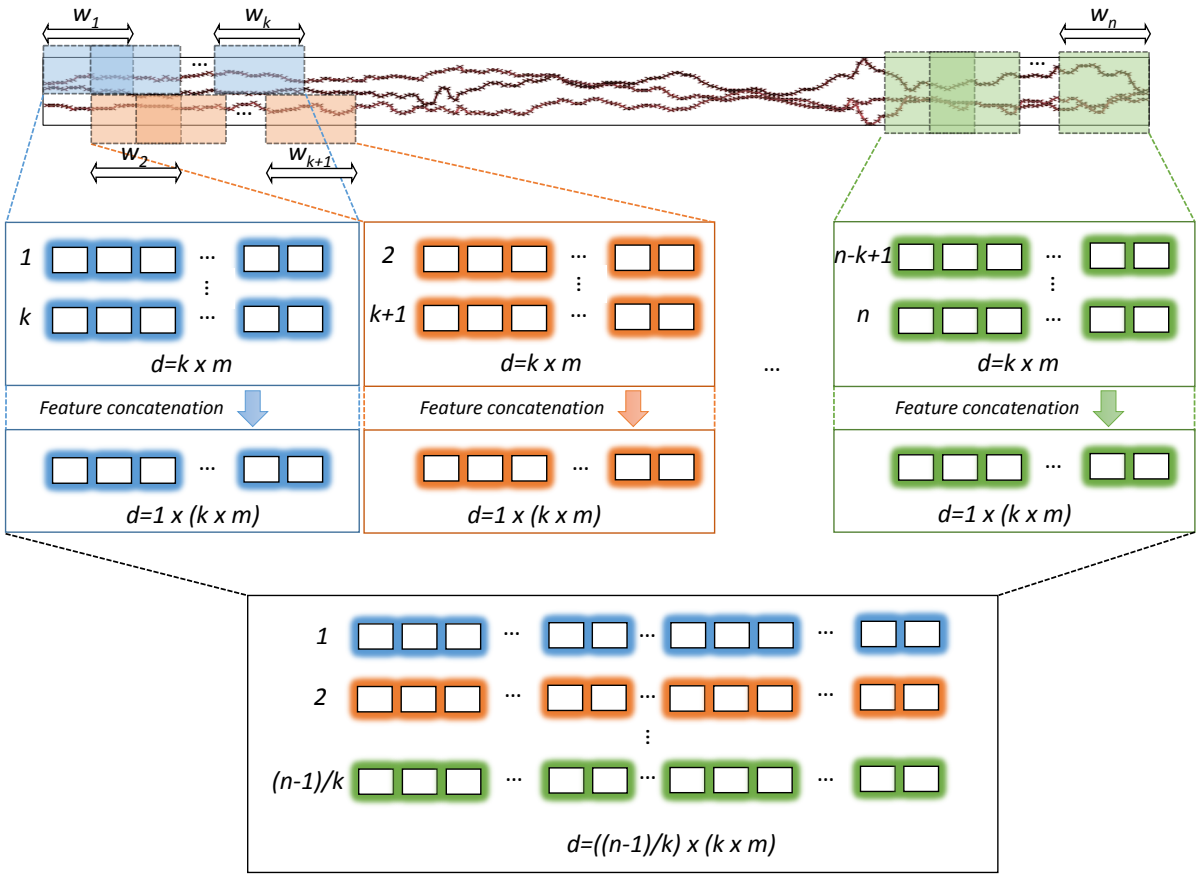

Figure 6: Diagram demonstrating the proposed feature concatenation method for overlapping subwindows.

classifier's ability to correctly label all positive instances. Using the harmonic mean to evaluate the classifiers ensures that its ability to perform both of these tasks successfully is considered.

4 RESULTS AND DISCUSSION

4.1 Summary of Findings

Table 2 shows the results of the three classifiers (Decision Tree, k-Nearest-Neighbour, and Support Vector Machine) for both approaches (traditional segmentation method (which serves as our baseline) using overlapping sliding windows and the proposed feature representation method). The results were produced after an exhaustive search over the following parameter lists:

(1) Window size (w): 30, 20, 15, 10, 5, 2 seconds (for the baseline method)
(2) Sub-window count (k): 8, 5, 3 seconds (for the proposed method only)
(3) Number of selected features (f_s^w, f_s^k): 3, 5, 8, 10, 20, 50, 100 (f_s^w for the baseline and f_s^k for the proposed)

For each classifier using the baseline method, the optimal window length was 30 seconds. For this window length, the 5 best features as selected by the ANOVA F-value algorithm were (1) maximum GPS speed, (2) integral of acceleration magnitude, (3) mean GPS speed, (4) first FFT component of acceleration magnitude, and (5) standard deviation of GPS speed. Within the top 10 features, 5 are derived from non-GPS sensors. The fact that the GPS-based features that were selected are derived from the speed, and the fact that the integral of the acceleration (which approximates change in velocity over the window) was one of the best features, could be explained by the fact that cruising is likely to result in slower driving.

For the proposed method, with the shorter 10s window length, the GPS features again dominated, with maximum speed being the best feature. However, among the top 10 features were components 2, 3 and 4 of the gyroscope's FFT, and accelerometer signal entropy.

These results show that SVMs outperform the Decision Tree and k-NN classifiers in both the baseline ($\gamma = 1$, $C = 1$) and the proposed ($\gamma = 0.59$, $C = 6.21$) approach. The optimal number of neighbors for k-NN approach were found to be 9 and 7 for the baseline and proposed approaches respectively. This is an expected result, because SVMs generally perform very well in the context of binary classification problems.

Table 2: Class-weighted F-1 scores of each classifier on each user, and the average across each train-test split, for the sliding-window approach and our proposed approach.

		Results						Parameters
Baseline		U1	U2	U3	U4	U5	μ_h	(w, f_s^w)
	DT	0.58	0.74	0.71	0.71	0.80	0.70	$(30s, 10)$
	k-NN	0.76	0.67	0.66	0.81	0.89	0.72	$(30s, 10)$
	SVM	0.80	0.72	0.85	0.81	0.75	0.77	$(30s, 3)$
Proposed		U1	U2	U3	U4	U5	μ_h	(w, k, f_s^k)
	DT	0.66	0.68	0.74	0.73	0.80	0.72	$(7s, 5, 5)$
	k-NN	0.77	0.59	0.79	0.81	0.88	0.75	$(8s, 6, 10)$
	SVM	0.81	0.77	0.85	0.87	0.75	0.81	$(10s, 3, 10)$

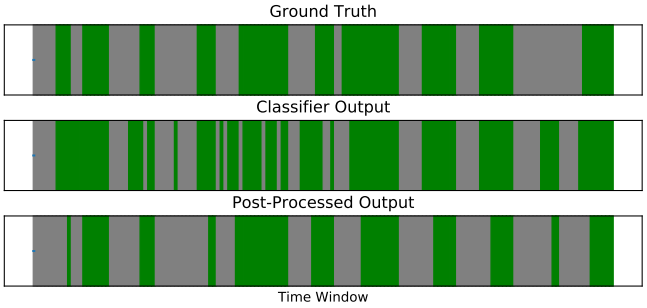

Figure 7: Ground Truth (annotation), classifier output, and post-processed output of the SVM for the baseline approach for one user. Green and grey coloured areas depict cruising and not-cruising events respectively.

The results also show that for the best performing classifier with each approach (SVM), there is a 4% improvement in the mean of the F-1 scores across all users using the proposed approach.

Figure 7 shows a plot of the class of each time window (Green depicting cruising and Grey depicting not-cruising on the Y-axis) vs. time on the X-axis as determined by the distance-based annotation method (for the user's 11 journeys concatenated together), along with the raw classifier output, and the output of the classifier following a simple post-processing technique. As evident from this figure, the proposed cruising detection approach is able to accurately predict cruising in most cases.

Figure 8 shows the confusion matrices for the baseline and proposed (sub-window based) classifiers. The benefit of the SVMs over the other two classifiers is very clear, as the false positive rate is greatly reduced (note the total number of labels differ in all cases as a result of optimisation over the window sizes and subwindow counts).

4.2 Environmental Impact

Use of the ParkUs app can provide a number of benefits; the most important being reduction in congestion which in turn could potentially reduce vehicle emissions. Further benefits could arise from the time and money saved not only for the user (as they are able to find a parking space much faster) but also improving journey times for other users on the road. Whilst tracking reduced congestion is difficult to quantify, it may be possible to get information from the user on their perceived amount of time saved (using ParkUs) trying to find parking on average through qualitative surveys or questionnaires. This will be the focus of our future work through an expanded trial being planned with the University of the West of England (UWE) in the near future.

To estimate the potential savings in terms of air quality a CO_2 emission model was developed. Using publicly available UK census data on daily work commutes through a web tool called Datashine [12] we were able to estimate the total number of daily trips (around $28,500$) made by vehicles into the city centre of Bristol.

From our collected training data we estimated the average distance vehicles traveled whilst cruising to be around 0.57km. Since vehicles travel at much slower speeds when cruising compared to when being driven on highways, we corrected our fuel consumption estimates accordingly and used 250 grams of CO_2 generated per kilometre [2] driven at an average velocity of 15 km/h (as typically experienced whilst cruising).

Consequently, if we were to assume that at least 75% of those daily trips resulted in cruising for a parking space (this is a conservative estimate since our data puts the figure closer to 90%), we were able to estimate that over a year (of 260 working days), cruising in the city centre of Bristol produces just over 790 metric tons of CO_2. This value is a conservative estimate given that we don't include non-work and weekend trips made by drivers as we do not have reliable enough data to make this estimate.

4.3 Economic Benefits

Using the same figures for fuel efficiency at low velocities (from Fontaras et al.) as used in our CO_2 emission model we estimated the cost of cruising to drivers over a year in Bristol. Using the current price for petrol in the UK (around £1.17 or roughly US$1.50 per litre at the time of writing) we can estimate the cost (in terms of fuel consumed) of a single cruising activity to be approximately £0.06 [13]. Assuming again, similar to our CO_2 estimates that 75% of the $28,500$ daily commute journeys into the centre of Bristol experience an average cruise time of 2.27 minutes (from our data set) we estimate the total yearly cost (i.e. assuming 260 working days a year) of fuel consumed cruising in the City of Bristol to be around £$368,000$ (US$$467,000$).

4.4 Behaviour Change

One of the other aims of ParkUs is also to nudge the drivers not to use their vehicles as much as possible. One can envisage this to be a positive side-effect of ParkUs wherein a user queries ParkUs before beginning a journey, sees that there is little or no parking available in the destination area and therefore decides to use an alternative means of transport such as taxis, public transport or decides to walk if the

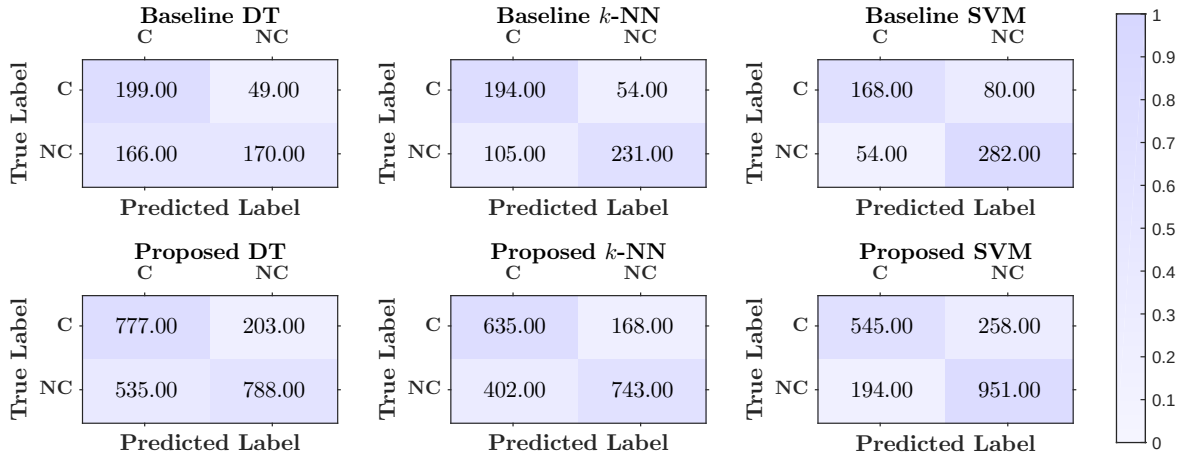

Figure 8: Sum of confusion matrices for all the classifiers' results using both the baseline and proposed approaches.

destination is not too far away. Whilst this is difficult to quantify, we believe, such information could also be gleaned from qualitative surveys or questionnaires. It is worth noting that whilst ParkUs is designed as a standalone app, it could also be integrated as a part of a larger travel planning application. For example, the tfl.gov.uk transport planner for London is quite popular for planning a journey from one part of the city to the other and displays the different modes of transport; bus, tube, cycling or walking. It is not hard to imagine such an application also displaying parking availability in the destination area of interest in addition to the other modes of transport. Devising mechanisms to identify what proportion of users actually avoid using personal vehicles and instead opt for public transport would be an interesting study but outside of the scope of this paper.

4.5 Challenges in Realising ParkUs

Realising a crowd-sensing based parking availability information app is a hard problem. Without the app reaching a critical mass of adoption it will not be possible to provide accurate parking availability information. It is essential for lots of users to be using the application on an ongoing basis such that up to date parking information can be recorded and displayed to the users. Whilst public outreach and engagement is vital to accomplishing this, enlisting a large number of users to start with would be quite challenging as our initial experience suggests. Getting early adopters to regularly use the application would be hard, as there may be little benefit for them initially. One way to work around this is to conduct trials in smaller areas of the city by employing set of participants that live or work within these target areas. Only once the credibility of the application is established, are we likely to see more drivers using it.

Although we aim to reduce user input as much as possible in order to provide a seamless service, potentially a couple of well thought and executed prompts could help the ParkUs 2.0 system greatly. For instance, when a driver parks, a notification could prompt the driver to input a simple "yes/no" to suggest if parking space is available in the vicinity based on a simple visual inspection (a quick glance around). In the instance where no user reports are available in a particular area, parking availability could be marked as "unknown" and displayed with "grey legend" in the heatmap. Thus, the more users that contribute information to the system, the more accurate and up to date the information maintained by the backend system and available to the users. Another user input could be a very simple estimation of how long they were cruising to find a parking space. Our current model relies on the driver getting near to the destination before moving further a field to find parking. Some drivers might preempt this by parking somewhere along the route (before their desired destination) and walking the rest of the way. By recording how long they searched for and comparing it with historic data for the surrounding roads could prove useful in providing an estimate of parking availability, especially in areas less frequently visited by the users of ParkUs 2.0. Additionally, knowledge of cruising detection flagged by the model could be used to update parking availability of the road segments where this was detected. The consolidated heatmap maintained by the backend system can then be made available to any user of the system who searches for parking availability in the given destination area.

In terms of evaluating the efficacy of ParkUs itself, one could use the percentage of the user journeys where the user parked on a road segment indicated as "available" by the app, percentage of user journeys where the user parked on a road segment indicated as "un-available" by the app and percentage of journeys where the user parked on a road segment indicated as "unknown" by the app. This should throw light on the usefulness of the app in terms of the accuracy of the information presented to the user. Since the focus of the study so far has been on cruise detection, we do not have the data to carry out the above mentioned

analysis. However, we intend to instrument this in the app in preparation for the next trials.

In general, it is worth noting that in the absence of the app, the user is merely at the mercy of chance for finding a parking space. Using the app, however may potentially increase this chance by helping the user to guide themselves to a road with higher probability of finding a vacant parking space. This in turn would help to also accrue the aforementioned benefits. The ParkUs app performs well when there are engaged users competing for parking spaces, this could be anywhere in a city. However, more competition for parking typically occurs in city centres due to the physical constraints on space and the high demand from daily commuters. This is in contrast to typical suburbia, where residents can park on their own driveways. However, competition for parking in suburbs may move from on-street parking to large car parks typically found at shopping centres, universities or enterprise zones outside of the city centre.

5 CONCLUSION AND FUTURE WORK

In this paper we presented a novel system for detecting cruising behaviour in vehicle journeys and developed a real-time parking information system, ParkUs 2.0 around this feature. We extensively evaluated our method using data collected with 5 users (41 journeys). Using our approach, we are able to detect cruising on average 81% of the time.

Although there are some obstacles surrounding adoption of our ParkUs system that need to be addressed, these, however, are outside the scope of this paper and could be the focus of future work. Firstly, remembering to use the app is an issue which we have found to be an important one based on our early experience. Secondly, even if a user were to sign-up for using the app, sustaining an engagement with the user ensuring that they continue using it on an ongoing basis throws open opportunities for investigating different incentive mechanisms. Thirdly, whilst validation of the app is vital prior to roll out, such validation has to be conducted through controlled tests. On the other hand, critical mass is essential to validate the app in an uncontrolled setup which is a challenge. Furthermore, given the app relies on crowdsensing which does not have knowledge of fine-grained parking information (each individual parking space) but instead coarse grained probabilistic information (likelihood of availability on a road segment), ensuring accuracy/usefulness of this information will require sustained user engagement and clever techniques that could potentially fuse data from multiple different sources. Overall, we believe that this is a fertile ground for research throwing open several interesting problems along a number of different dimensions.

6 ACKNOWLEDGEMENTS

The authors would like to thank Pawel Laskowski for his help in developing the Android smartphone application. This research was supported by the EPSRC funded Industrial Doctorate Centre in Systems (Grant [EP/G037353/1]) and the EU's Horizon 2020 programme under grant agreement No. [691735].

REFERENCES

[1] Pietro Carnelli, Joy Yeh, Mahesh Sooriyabandara, and Aftab Khan. 2017. ParkUs : A Novel Vehicle Parking Detection System. In *AAAI Conference on Innovative Applications*. San Francisco, US, 4650–4656.

[2] Georgios Fontaras, Nikiforos-Georgios Zacharof, and Biagio Ciuffo. 2017. Fuel consumption and CO2 emissions from passenger cars in Europe – Laboratory versus real-world emissions. *Progress in Energy and Combustion Science* 60 (2017), 97–131.

[3] F. Gaebler and O. Dousse. 2017. Method and Apparatus for providing parking availability detection based on vehicle trajectory sampling. US Patent Application Publication Number US20170103654A1. (2017).

[4] Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of machine learning research* 3, Mar (2003), 1157–1182.

[5] Nils Y. Hammerla, Reuben Kirkham, Peter Andras, and Thomas Ploetz. 2013. On preserving statistical characteristics of accelerometry data using their empirical cumulative distribution. In *Proceedings of the 17th annual international symposium on International symposium on wearable computers - ISWC '13*. ACM Press.

[6] Nils Y. Hammerla and Thomas Plötz. 2015. Let's (not) stick together. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '15*. ACM Press.

[7] Samuli Hemminki, Petteri Nurmi, and Sasu Tarkoma. 2013. Accelerometer-based transportation mode detection on smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems - SenSys '13*. ACM Press.

[8] Jacek Kopecký and John Domingue. 2012. PARKJAM: Crowdsourcing Parking Availability Information with Linked Data (Demo). *Extended Semantic Web Conference (ESWC)* (2012).

[9] Trista Lin, Herve Rivano, and Frederic Le Mouel. 2017. A Survey of Smart Parking Solutions. *IEEE Transactions on Intelligent Transportation Systems* (2017).

[10] Sarfraz Nawaz, Christos Efstratiou, and Cecilia Mascolo. 2013. ParkSense: A Smartphone Based Sensing System For On-Street Parking. In *Proc. MobiCom*. ACM, Cambridge.

[11] 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 - GIS '09*. 336.

[12] Oliver O'Brien and James Cheshire. 2016. Interactive mapping for large, open demographic data sets using familiar geographical features. *Journal of Maps* 12, 4 (2016), 676–683.

[13] Automobile Association Plc. 2017. *Fuel price report (May 2017)*. Technical Report May. The AA, Basingstoke. 2 pages. http://www.theaa.com/driving-advice/driving-costs/fuel-prices

[14] Rosario Salpietro, Luca Bedogni, Marco Di Felice, and Luciano Bononi. 2015. Park Here! A Smart Parking System Based on Smartphones' Embedded Sensors and Short Range Communication Technologies. In *Proc. WF-IoT*. IEEE.

[15] Abraham Savitzky and Marcel JE Golay. 1964. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry* 36, 8 (1964), 1627–1639.

[16] Bernhard Schölkopf and Alexander J. Smola. 2001. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA.

[17] Donald C. Shoup. 2006. Cruising for Parking. *Transport Policy* 13, 6 (2006), 479–486.

[18] Leon Stenneth, Ouri Wolfson, Bo Xu, and Philip S. Yu. 2012. PhonePark: Street Parking Using Mobile Phones. *Proc. Mobile Data Management* (2012).

[19] United Nations. 2014. *World Urbanization Prospects*. Technical Report. http://esa.un.org/unpd/wup/Highlights/WUP2014-Highlights.pdf

[20] 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.