Hindawi
Mobile Information Systems
Volume 2021, Article ID 9973681, 20 pages
https://doi.org/10.1155/2021/9973681

Research Article

Classifying Vehicle Activity to Improve Point of Interest Extraction

James Van Hinsbergh,[1] **Nathan Griffiths**,[1] **Phillip Taylor**,[1] **Zhou Xu**,[2] **and Alex Mouzakitis**[2]

[1]*Department of Computer Science, University of Warwick, Coventry, UK*
[2]*Jaguar Land Rover, Engineering Centre, Coventry, UK*

Correspondence should be addressed to James Van Hinsbergh; j.van-hinsbergh@warwick.ac.uk

Received 27 March 2021; Accepted 12 August 2021; Published 3 September 2021

Academic Editor: Alessandro Bazzi

Knowledge of drivers' mobility patterns is useful for enabling context-aware intelligent vehicle functionality, such as route suggestions, cabin preconditioning, and power management for electric vehicles. Such patterns are often described in terms of the Points of Interest (PoIs) visited by an individual. However, existing PoI extraction methods are general purpose and typically rely on detecting periods of low mobility, meaning that when they are applied to vehicle data, they often extract a large number of false PoIs (for example, incorrectly extracting PoIs due to stopping in traffic), reducing their usefulness. To reduce the number of false PoIs that are extracted, we propose using features derived from vehicle signals, such as the selected gear and status of doors, to classify candidate PoIs and filter out those that are irrelevant. In this paper, we (i) present Activity-based Vehicle PoI Extraction (AVPE), a wrapper method around existing PoI extraction methods, that utilizes a postclustering classification stage to filter out false PoIs, (ii) evaluate the benefits of AVPE compared to three state-of-the-art general purpose PoI extraction algorithms, and (iii) demonstrate the effectiveness of AVPE when applied to real-world driving data.

1. Introduction

Point of Interest (PoI) extraction is useful for automatically discovering locations that are relevant to a user for a given application. For example, PoIs can provide an understanding of a person's daily routine, their frequently visited locations, and the type of journeys they undertake. With this knowledge, intelligent systems can be designed to customize a vehicle for a given trip, for example, altering the climate control or tailoring the media settings. Previous work on PoI extraction typically uses periods of low movement to detect PoIs, in applications such as detecting mobility patterns in a city [1, 2] or animal migration patterns [3]. When applied to vehicle applications, where low movement does not necessarily imply that a vehicle has stopped for a specific purpose of interest, this can lead to the generation of false PoIs. For vehicle applications, a PoI is considered to be a location where the vehicle has stopped for an intended purpose, whether that be to park, drop off a passenger, or visit a drive-through service.

The aim of AVPE is to find representative locations within a user's trajectories, with a focus on ensuring that all of the identified locations are correct, rather than necessarily being complete. Thus, AVPE aims to remove noise in the form of erroneous PoIs, even if this is at the cost of reducing the number of correct PoIs. For applications such as customer segmentation [4] or categorizing usage in a vehicle context [5], the presence of noise can significantly skew the results. Since previous general purpose PoI extraction methods rely on detecting periods of low mobility, in vehicle applications, where drivers are likely to encounter traffic, this can cause significant problems. For such applications, it is more important to have an aggressive approach to noise reduction, rather than ensuring that the complete set of true PoIs is extracted.

The scope of this work is to create a methodology for identifying representative locations in vehicular trajectories, using basic data available from the vehicle data bus. By using basic on-board data, which is common across vehicles, AVPE can be applied to different vehicles without requiring

additional sensors or external data, the latter of which may not be available in some geographic regions.

In this paper, we (i) present Activity-based Vehicle PoI Extraction (AVPE), a wrapper around existing PoI extraction methods that uses a postclustering classification stage to filter out false PoIs from the extraction process, (ii) evaluate AVPE against three state-of-the-art general purpose PoI extraction algorithms, and (iii) demonstrate its effectiveness when applied to real-world driving data. This paper extends our previous work in [6] by formalizing the AVPE method, considering Random Forest classification, in addition to Support Vector Machines, and evaluating AVPE with the Clustering-Based Stops and Moves of Trajectories (CB-SMoT) clustering algorithm [7], in addition to the Spatio-Temporal Activities (STA) [8] and Gradient-based Visit Extractor (GVE) [9] algorithms. Additionally, we have incorporated feature selection into AVPE, using the Kneedle algorithm [10] instead of relying on a manual user-defined process as used in [6]. We analyze the performance of AVPE using the CB-SMoT, STA, and GVE clustering algorithms for vehicle trajectory data and evaluate the method on both scripted and unscripted real-world driving data.

This paper is organized as follows. Section 2 reviews related work, and Section 3 presents AVPE, our proposed wrapper method for PoI extraction. In Section 4, we describe our experimental methodology, introduce our datasets, and detail the process followed to collect them. Section 5 presents the results of applying CB-SMoT, STA, GVE, and AVPE on vehicle data and provides a direct comparison between the effectiveness of each method. Finally, Section 6 concludes the paper.

2. Related Work

The process of PoI extraction typically starts with a GPS trajectory, which is a temporally ordered sequence of instances, where each instance has a timestamp, latitude, and longitude. A PoI is typically defined as a group of instances in a trajectory that exhibits little or no movement, implying a period of low mobility, in which an individual remains in the same location [7–9]. Given this definition, Palma et al. [7], Bamis and Savvides [8], and Thomason et al. [9] assume that all periods of low mobility are meaningful, which is not necessarily the case for vehicle data, where areas of low movement exist that are not relevant to a user, such as waiting in traffic. This results in existing general purpose PoI extraction algorithms generating multiple false PoIs when applied to vehicle data.

PoI extraction is normally used as a preprocessing step prior to another form of analysis or prediction. Many applications, such as destination prediction, rely on robust PoI extraction to provide acceptable performance [11–14]. PoI extraction can also be used to identify semantically relevant places for users and to highlight public attractions. For example, Keles et al. use a Bayesian approach that considers the duration of the stationary period, the day of the week, and the arrival time to predict the category of a PoI [15]. Similarly, inferring the activity performed at a given PoI is investigated by Furletti et al. [16], by linking PoIs to amenities using semantic data such as a users' maximum walking distance between a vehicle parking location and their intended destination, and the opening hours of facilities located near the PoI. Furletti et al. assume that it is not always possible to park directly at the intended location and so rely on a user-provided maximum walking distance as a threshold for the PoIs to consider. Semantically relevant places such as a user's home or work can also be considered when developing location-aware applications [11, 17].

Extracting PoIs is becoming increasingly important for location-aware applications, and several techniques have been applied to this problem, the most common being clustering. Multiple clustering algorithms exist that can be applied to PoI extraction, typically using density-based approaches [18]. Additionally, some clustering algorithms have been proposed specifically for PoI extraction, namely, CB-SMoT [7], STA [8], and GVE [9], which represent the current state-of-the-art.

DBSCAN is a widely used density-based clustering algorithm that has the advantage of not requiring the number of clusters to be specified in advance, which is useful for PoI extraction, since this is typically unknown [18]. Another advantage of DBSCAN over other general purpose clustering algorithms, such as k-means [19], is that it can cope with clusters of different shapes. DBSCAN uses two parameters, ϵ and min pts, that, respectively, determine the absolute distance used to calculate the neighborhood of an instance, and the minimum number of instances that a cluster should contain. DJ-Cluster extends DBSCAN by considering usefulness, in addition to accuracy, where the usefulness metric describes the proportion of extracted PoIs that are meaningful to the user [20, 21]. This requires users to confirm whether the discovered PoIs are correct and to rate their importance on a 5-point scale. In the standard formulation of DJ-Cluster, all PoIs that are rated 4 or above are considered to be meaningful. DJ-Cluster also reduces the computational complexity, when compared to DBSCAN, by adopting a density-joinable approach. DJ-Cluster joins any two clusters that have identical instances in the neighborhoods of both clusters, rather than performing an outward neighborhood search on each instance in the resulting neighborhood. D-Star extends DJ-Cluster, using a sliding window to create the neighborhood, allowing the algorithm to work online [22]. D-Star identifies duration-joinable clusters instead of the density-joinable approach used in DJ-Cluster. This joins clusters together based on their duration overlap, which can handle missing instances within a PoI. ST_DBSCAN also extends DBSCAN, considering nonspatial, spatial, and temporal aspects to generate clusters [23].

The Clustering-Based Stops and Moves of Trajectories (CB-SMoT) algorithm [7] calculates a distance threshold, ϵ, over each trajectory, in contrast to DBSCAN, which uses the same value for all trajectories. Using the mean and standard deviation of the distances between consecutive instances allows a normal distribution to be created, and ϵ is set to be equal to the inverse cumulative probability of the distribution. Recalculating the distance threshold over each trajectory is beneficial, since it is difficult to provide a suitable threshold without knowing the properties of every trajectory

in advance. CB-SMoT also allows areas of known PoIs to be input, so that identified stops can be categorized into both known and unknown PoIs. Density-based approaches are computationally expensive, making them less desirable for use in resource constrained applications. However, since CB-SMoT was designed for trajectory data and recalculates a suitable distance threshold without the need for advance knowledge of trajectories, it is included in this paper for comparison.

Techniques such as Spatio-Temporal Activities (STA) [8] and the Gradient-based Visit Extractor (GVE) [9] use a buffer containing a number of previous instances in the trajectory, which is used to consider the distance from the current instance. Both STA and GVE iterate through the instances in the trajectory, adding these instances into the current cluster. If the distance exceeds a predefined threshold, then the candidate instance is considered to be moving away from the location and consequently ends the current cluster. STA uses a static distance threshold, in comparison to GVE, which uses a gradient-based threshold considering the current length of the buffer. Once the distance exceeds the threshold, and the current cluster ends, both STA and GVE assess whether the current cluster is retained or discarded. STA retains the current cluster if the buffer is full, while GVE does not require the buffer to be full and only discards a cluster in cases where there is no time difference between the first and last instances in the cluster.

A number of techniques use static time and distance thresholds, including the works of Kang et al. [24] and Fu et al. [25]. However, these techniques have been shown to exhibit poor performance when there is even a limited amount of noise in the data [8]. STA and GVE overcome this issue by using averaging filters to compare subsequent instances [26]. Chen et al. also employ static thresholds on taxi trajectory data, where GPS readings are sampled every 15 seconds [27]. Event durations in more general vehicle data typically vary between a few seconds (for a drop-off) to several minutes (for a drive-through service), and so the approach adopted by Chen et al. is prone to missing entire events. Bhattacharya et al. use a bucketing technique with time and distance to infer speed (and acceleration) [28]. They consider two different types of location, a point-based PoI such as an office, where the users' movement is negligible, and an extended PoI, such as a market, where the user will move slowly. More recently, Bhattacharya et al. considered a line segment-based approach that uses kernel density estimation as part of a two-phase process [29]. However, these speed and direction-based algorithms are not suited to extracting PoIs from vehicle trajectories, because they require a list of surrounding PoIs, which may not be available.

PoIs can vary in duration and shape, and there is no single approach or parameter configuration that is appropriate for all application domains. For example, a clustering algorithm with parameters trained on walking trajectories may be able to identify when a person travelling on foot is at a PoI; however, it may not be effective at detecting a PoI within vehicular trajectories, such as when a vehicle is at a drive-through service. Moreover, existing clustering algorithms typically generate large numbers of false PoIs for vehicle data in environments that contain road infrastructure and traffic, and therefore, such techniques do not give an accurate representation of a user's PoIs [7–9].

Our hypothesis in this paper is that existing clustering algorithms are not suitable on their own to extract useful PoIs from vehicle trajectories. We propose that adding a classification wrapper around existing PoI extraction methods will significantly improve their effectiveness when applied to vehicle data. In other application domains, activity classification has been used for many tasks ranging from detecting daily household activities [30–34] to specialized models predicting sports moves [35, 36]. Our proposed wrapper method, AVPE, introduces the notion of activity classification for vehicles. We apply techniques from existing activity classification approaches, such as the use of acceleration data [37] and sensor fusion [38], to vehicle activity classification.

3. Activity-Based Vehicle PoI Extraction (AVPE)

In this paper, we present Activity-based Vehicle PoI Extraction (AVPE), a novel wrapper method that uses a classification stage to filter out false PoIs that are extracted by existing clustering algorithms when applied to vehicle data. In our context, PoIs are defined as instances where the vehicle has stopped for a specific task (such as picking up a passenger or using a drive-through service), and they should be distinguished from false PoIs (such as waiting in traffic or stopping at a barrier). AVPE is a wrapper around existing clustering algorithms, which cluster periods of low mobility from historical trajectory data, generating a set of candidate PoIs. In this paper, we consider CB-SMoT, STA, and GVE as base clustering algorithms. Since vehicles frequently stop for reasons that do not represent PoIs, these three clustering methods return a large number of false PoIs when applied to vehicle trajectories. The AVPE wrapper method aims to reduce the number of false PoIs, accepting that this may be at the cost of missing some of the true PoIs. Thus, the overall aim of AVPE is to ensure that any identified locations are correct and that there is no noise, rather than aiming for completeness. Prior to applying AVPE, trajectory data is preprocessed using CB-SMoT, STA, or GVE, and a time threshold is used to merge distinct clusters that are close to each other in time. Using the resulting clusters, and features extracted from vehicle signals, AVPE then classifies the activity of the vehicle into one of several predefined activity types, where some activity types (positive labels) represent true PoIs and others (negative labels) represent the common types of false PoI extracted by the clustering methods. AVPE is, therefore, able to determine whether a candidate PoI is relevant or not. The approach of defining of labels, and the separation into positive and negative labels, corresponding to true and false PoIs, respectively, is fundamental to AVPE. Similarly, well-defined transitions between labels are required to ensure consistency. While we provide an example set of labels and transitions in this paper (see Section 4.1), our focus is on the AVPE method, rather than on a particular set of labels. The vehicle signals can include binary (e.g.,

engine on/off), categorical (e.g., indicator status), and numerical (e.g., steering wheel angle) values. The signals from the vehicle are expanded into features, comprising the minimum, maximum, range, and average for each of the vehicle signals computed for each cluster, in addition to the time above average, standard deviation, and first derivative for all numerical signals, and the delta for specific binary signals.

An overview of AVPE is given in Figure 1. AVPE uses a combination of vehicle signals and GPS data, and we define an instance x_j at time j to be a tuple $x_j = \langle \text{lat}, \text{long}, V \rangle$ containing a latitude, lat, longitude, long, and a vector of vehicle signal values, V. AVPE is retrospective in that it is used after journeys have been completed. While it is possible to adapt AVPE to use a naive time-based clustering approach to classify vehicle activity in real-time, this is not considered further in this paper.

AVPE requires training on a labelled set of data before it can be used on unseen trajectories. We assume that a set of labels, D, is defined, where the positive labels, $D^+ \subset D$, are activities that are of interest and correspond to true PoIs, and negative labels, $D \setminus D^+$, correspond to false PoIs that should be filtered out. Labelling is performed on each instance individually. To label the training data, the vehicle signals and both the activity labels and transition definitions are used to assign a label for each instance. At the start of training AVPE the training data, $\mathcal{T}_{\text{train}}$, is input, and the trajectories are clustered, with adjacent clusters up to λ seconds apart being merged together. These clusters are then used to train a classifier, ψ. The training algorithm iteratively increases the number of features selected by the feature selection algorithm and performs cross validation to obtain the area under the curve (AUC) [39]. If the current AUC is greater than the previously seen one, the record of the best classifier and feature set combination is updated accordingly. The best overall performing classifier and feature set that have been identified are output by the training algorithm. Algorithms 1 and 2 describe the preprocessing stage of AVPE (including the postclustering merging of clusters) and the vehicle feature extraction, respectively, with Table 1 defining the notation used within this paper. Algorithm 3 details the training process of AVPE, while the deployment version of AVPE, which is used to classify new trajectories, is described in Algorithm 4. The deployment algorithm takes five inputs: (i) the set of trajectories from which to extract PoIs, (ii) a threshold for merging clusters that are close together in time, (iii) a pretrained classifier (created using Algorithm 3), (iv) the feature set required by the pretrained classifier, and (v) the choice of clustering algorithm with pretrained parameters.

3.1. Base Clustering of Trajectories. AVPE begins with a preprocessing stage, as defined in Algorithm 1. This starts by generating clusters from each GPS trajectory, using only spatial and temporal information. This is achieved by inputting the data into an existing clustering algorithm, preferably an algorithm that discards outliers instances, since these will not be PoIs. Even though the clustering stage only uses spatial and temporal data, the vehicle data exists within each instance and so is available for use in the later stages of AVPE. In this paper, we consider CB-SMoT [7], STA [8], and GVE [9] as representative clustering algorithms.

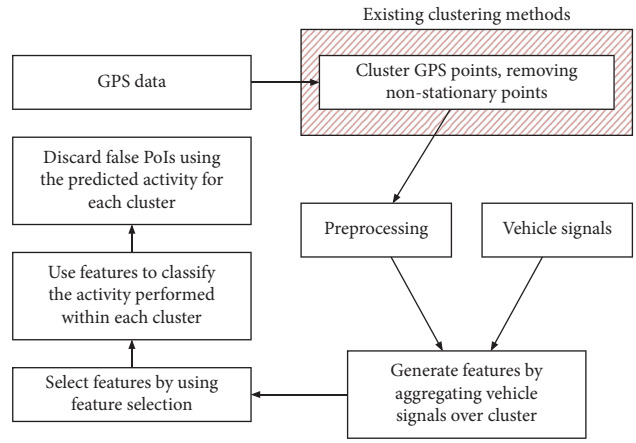

Figure 1: Overview of AVPE.

Clustering algorithms typically have parameters that can significantly alter the output that is generated. In this paper, to optimize the parameters for each clustering algorithm, we perform simulated annealing [40] using the training set. To compare the performance of a given parameter combination, we aim to maximize the number of nondriving instances that are clustered, while minimizing the number of driving instances that are clustered. This performance is quantified using the Sørensen-Dice coefficient (set overlap) metric [41, 42], defined as

$$\text{QS} = \frac{2|A \cap B|}{|A| + |B|}, \tag{1}$$

where QS is the quotient of similarity, A is the set of instances in a ground truth cluster, and B is the set of instances in an extracted cluster. This metric is limited by the equal weighting given to all instances and may be viewed as simplistic. Other metrics, such as that proposed by Ward et al. [43], define specific error types, enabling each kind of error to be individually weighted. However, a previous work has shown that using a set overlap results in clustering parameters being identified by simulated annealing that give a higher overall classification performance [6].

3.2. Postcluster Merging. Due to the nature of the trajectories and the vehicle activities, multiple clusters can be generated that are part of the same event, for example, drive-through and traffic events. Such events can include short periods of movement, causing a new cluster to be started. Fragmented clusters will cause a drop in classification performance due to the aggregated vehicle signals used in AVPE being calculated over periods of time, which do not reflect the whole activity. Figure 2 shows an example scenario, in which road traffic causes a vehicle to stop 3 times, with short periods of slow movement between the stops (the slight differences in latitude and longitude at each stop are due to GPS jitter). The

Table 1: Notations used in this paper.

Notation	Description
$\mathcal{T} = \{t_1, \ldots, t_n\}$	The set of n trajectories
$t_i = [x_1, \ldots, x_{\lvert t_i \rvert}]$	The i^{th} trajectory within $\mathcal{T}$, a strictly ordered sequence of $\lvert t_i \rvert$ instances
$\mathcal{V}$	The set of vehicle signals
V_j	The values of the vehicle signals at time j
$x_j = \langle \text{lat}, \text{long}, V \rangle$	An instance x_j is a latitude and longitude position, lat, long, and the values V of vehicle signals at time j
$V^R_{a,b}$	A matrix of the real-valued signals in $\mathcal{V}$ for the time interval from a to b, i.e., $(V_a, V_{a+1}, \ldots, V_b)$
$V^N_{a,b}$	A matrix of the categorical (nominal) signals in $\mathcal{V}$ for the time interval from a to b
$V^B_{a,b}$	A matrix of the binary signals in $\mathcal{V}$ for the time interval from a to b
$s^i_{a,b} = [x_a, \ldots, x_b] \subseteq t_i$	A strictly ordered subsequence of instances $\{x_j \lvert a \le j \le b\}$ in trajectory t_i
$\mathcal{C}$	The set of clusters extracted from all trajectories in $\mathcal{T}$
$c^i_m = s^i_{a,b} = [x_a, \ldots, x_b] \subseteq t_i$	The m^{th} cluster of trajectory t_i, defined as a strictly ordered subsequence of instances $\{x_j \lvert a \le j \le b\}$, where a is the first instance and b is the last instance temporally
$\text{time}(x_j)$	A function that returns the time j of the instance x_j
$\text{head}(c^i_m)$	A function that returns the first instance in cluster c^i_m
$\text{last}(c^i_m)$	A function that returns the last instance in cluster c^i_m
$\text{delete}(c^i_m)$	A function that deletes the cluster c^i_m
$\text{split}(\mathcal{C}, k)$	A function that returns an array of training and validation clusters for a given number of folds, k
$\text{truth}(c^i_m)$	A function that returns the ground truth classification label for c^i_m
score (TP, FP, TN, FN)	A function that returns the AUC
$\text{filter}(\omega, F)$	A function that returns the feature values in F for the features that are present in feature set ω
$\oplus$	The $\oplus$ operator is used to denote the concatenation of two sequences
F	The set of features that can be extracted from $\mathcal{V}$
D	The set of possible classification labels
D^+	The set of positive classification labels, $D^+ \subseteq D$
ψ	A pretrained classifier
ω	The feature set used in the classifier ψ
ϕ	A prediction from the classifier ψ

inputs: $\mathcal{T}$, the set of n trajectories, $\{t_1, \ldots, t_n\}$
λ, the merge threshold
cluster, the chosen clustering algorithm, with pretrained parameters
output: $\mathcal{C}$, a set of preprocessed clusters

(1) $\mathcal{C} = \text{cluster}(\mathcal{T})$
(2) **if** $\lambda > 0$ **then**
  //merge adjacent clusters up to λ seconds apart
(3) **for** $t_i \in \mathcal{T}$ **do**
(4) **for** $c^i_m \in \mathcal{C}$ **do**
  //calculate time difference between current and previous cluster
(5) $q = \text{time}(\text{head}(c^i_m))$
(6) $p = \text{time}(\text{last}(c^i_{m-1}))$
(7) **if** $(q - p) < \lambda$ **then**
  //append all instances from start of previous cluster to end of current cluster
(8) $c^i_{m-1} = c^i_{m-1} \oplus s^i_{p+1,q-1} \oplus c^i_m$
(9) $\text{delete}(c^i_m)$
(10) **end**
(11) **end**
(12) **end**
(13) **end**
(14) **return** $\mathcal{C}$

Algorithm 1: Preprocess (cluster, $\mathcal{T}, \lambda$)—preprocessing stage of AVPE.

overall output should be a single traffic activity; however, all three clustering algorithms considered in this paper have the potential to separate this traffic event into 3 separate clusters, especially if the GPS coordinates contain inaccuracies or noise.

To rectify this, we propose merging clusters that are within a defined temporal threshold of each other as part of the preprocessing stage of AVPE. We define a merge threshold, λ, to be the minimum number of seconds that is needed to separate consecutive clusters. In Algorithm 1, we compare the time of the first instance in cluster c^i_m and the time of the last instance in cluster c^i_{m-1}. If the difference in time between these two instances is less than λ, then c^i_{m-1} and c^i_m will be merged. Clusters are merged by concatenating the

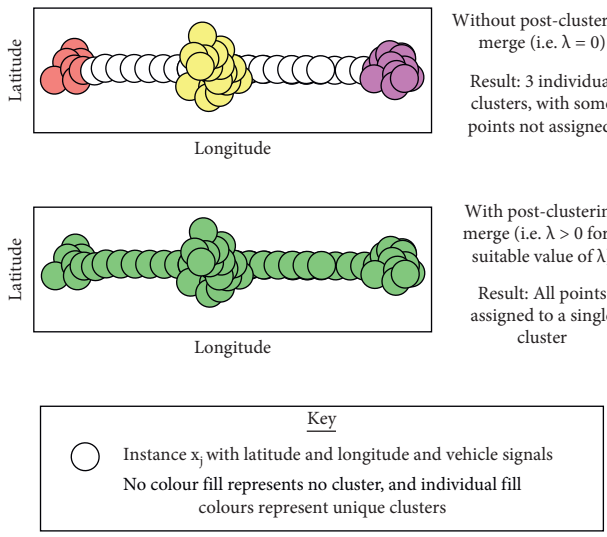

FIGURE 2: Example of a fragmented traffic event, and how post-cluster merging can mitigate this.

sequence of instances in the previous cluster, the current cluster, and any instances that are temporally between them. This helps reduce fragmented clusters (as illustrated in Figure 2), aiding the classification stage.

3.3. Signal Aggregation and Classification. The training stage of AVPE, as detailed in Algorithm 3, takes a set of training trajectories, $\mathcal{T}_{\text{train}}$, and a merge threshold, λ, along with the chosen methods for clustering, feature selection and classification, the Kneedle algorithm [10], and the number of folds to use for cross validation, k. The output of the training stage is a trained classifier, ψ^*, and the feature set used in the classifier, ω^*. The training algorithm first preprocesses the training trajectories (using Algorithm 1), assigning instances to clusters and merging nearby clusters together. With instances now assigned to clusters, the majority class (>50%) of the instances within each cluster determines the class label to be applied. Should a majority class containing greater than 50% of the instances do not exist, then the cluster is discarded.

With preprocessing being complete, the AVPE training algorithm uses an incremental search to find the best performing feature set, starting from training a classifier using a single feature, up to using all the available features. We adopt k-fold cross validation in the training stage to help reduce the bias. For each fold, we split the training data into a training and validation set, by assigning journeys to k partitions, where a single partition is used as the validation set (see line 3 in Algorithm 3). Features are then extracted for each cluster, as shown in Algorithm 2. Time and location are present in the data for each instance, but these signals are not used for the classification stage in AVPE, since they have already been used for clustering. The features extracted comprise the minimum, maximum, range, and average for each signal, along with the time above average, standard deviation, and first derivative for each numerical signal calculated over each cluster. Additionally, binary signals can also include a delta feature, which shows the relative change between the start and end of each cluster. For each cluster that is input, each of the signal vectors is concatenated to form a matrix, which is then input to element-wise operations (such as max, min, and mean in Algorithm 2) to calculate the aggregated value over the rows in the matrix. The calculated features for all clusters in the training set (in the current fold, $\mathcal{C}_{\text{train}}[k']$) are input into the chosen feature selection algorithm (such as Minimal Redundancy Maximal Relevance [44] or Principal Component Analysis [45]), along with the number of features to select. This will return a feature set to be used in the classification method. Feature selection is needed, since, by generating multiple statistical properties for each signal (e.g., the minimum, range, and average), there is potential for overlap between features, where multiple features can provide similar and redundant information. Additionally, we do not want to prejudge which features perform best, and different datasets for which AVPE might be applied may have different vehicle signals and features available. A classifier is then trained, using the chosen classification method and the feature set output by the feature selection algorithm. Using this newly created classifier, we iterate over each cluster in the validation set, using the chosen feature set to predict one of the activity labels. The prediction is compared to the ground truth for each label, and the count of true positives or false positives is incremented as appropriate. Once predictions have been made for all clusters in the validation set, the AUC is calculated and stored. This is repeated until all the features have been included in the feature set, and the resulting AUC values are input into the Kneedle algorithm [10], to identify the knee point of the curve. The knee point determines the feature set to use in AVPE, and the training stage returns the corresponding classifier and feature set. Alternative stopping criteria can be used for more robust feature selection, but since the feature selection method itself is not the focus of this paper, this simple approach was used.

3.4. Deployment. The deployment stage of AVPE is detailed in Algorithm 4. Data is collected from the vehicle in a batched manner, with AVPE being run on batches of trajectories as they become available (i.e., there is a buffer in which trajectories are stored, and once the buffer is full, the trajectories are processed, and the buffer is reset). The classifier resulting from the training stage (Algorithm 3) and the feature set used in this classifier are input to the deployment stage, along with the merge threshold, λ, and the chosen method for clustering, as used in the training stage. The trajectories in the buffer, $\mathcal{T}$, that are to be processed are also input. The preprocessing stage is identical to that used in the training stage of AVPE. After the trajectories have been preprocessed, and clusters have been created, the deployment stage of AVPE iterates through each cluster, calculating the feature values. These feature values are filtered according to the feature set used by the classifier and input into the classifier. A prediction for the cluster is given, and if this prediction is in the set of positive labels, D^+, then the cluster is added to the return set. This process is repeated for

input: c_m^i, the k^{th} cluster of trajectory t_i
output: F, a set of features calculated over all vehicle signals in cluster c_m^i
//get start and end time of cluster
(1) $a = \text{time}(\text{head}(c_m^i))$
(2) $b = \text{time}(\text{last}(c_m^i))$
//calculate features using element-wise operations over matrices
//calculate different features for real-valued, categoric and binary types
(3) $F = F \cup \max(V_{a,b}^R) \cup \min(V_{a,b}^R) \cup \text{mean}(V_{a,b}^R) \cup \text{range}(V_{a,b}^R)$
(4) $F = F \cup \text{stdev}(V_{a,b}^R) \cup \text{firstderivative}(V_{a,b}^R) \cup \text{timeabvmean}(V_{a,b}^R)$
(5) $F = F \cup \max(V_{a,b}^N) \cup \min(V_{a,b}^N) \cup \text{mean}(V_{a,b}^N) \cup \text{range}(V_{a,b}^N)$
(6) $F = F \cup \max(V_{a,b}^B) \cup \min(V_{a,b}^B) \cup \text{mean}(V_{a,b}^B) \cup \text{range}(V_{a,b}^B)$
//calculate element-wise delta operation over specific binary signals
(7) $F = F \cup \text{delta}(V_{a,b}^B \text{ where } V^B \in \text{deltaSignals})$
(8) **return** F

Algorithm 2: Features (c_m^i)—extracting features for a cluster.

inputs: $\mathcal{T}_{\text{train}}$, a set of n training trajectories, $\{t_1, \ldots, t_n\}$
λ, the merge threshold
cluster, the chosen clustering algorithm, with pretrained parameters
selection, the chosen feature selection algorithm
classificationMethod, the chosen classification method
kneedle, the Kneedle algorithm [10].
k, the number of folds to use for cross validation
output: ψ^*, a trained classifier
ω^*, the feature set used in the classifier ψ^*
(1) $\mathscr{C}_{\text{train}}, \mathscr{C}_{\text{validation}}, \psi^*, \omega^*, \text{AUC} =$
(2) $\mathscr{C} = \text{preprocess}(\text{cluster}, \mathcal{T}_{\text{train}}, \lambda)$
//split training and validation data
(3) $\mathscr{C}_{\text{train}}, \mathscr{C}_{\text{validation}} = \text{split}(\mathscr{C}, k)$
(4) **for** $F_{\text{num}} \in \text{count}(1, |F|)$ **do**
(5) TP, FP, TN, FN =
(6) **for** $k' \in \text{count}(1, k)$ **do**
//calculate features for each cluster (see Algorithm 2)
(7) **for** $c_m^i \in \mathscr{C}_{\text{train}}[k']$ **do**
(8) $F = F \cup \text{features}(c_m^i)$
(9) **end**
//select features and train classifier
(10) $\omega = \text{selection}(F_{\text{num}}, F)$
(11) $\psi = \text{train}(\text{classificationMethod}, \text{filter}(\omega, F))$
(12) **for** $c_m^i \in \mathscr{C}_{\text{validation}}[k']$ **do**
(13) $\phi = \psi(\text{filter}(\omega, \text{features}(c_m^i)))$
//compare the prediction to ground truth for each label
(15) **for** $d \in D$ **do**
(16) **if** $\phi = d \wedge \phi = \text{truth}(c_m^i)$ **then** $\text{TP}[d] += 1$
(17) **if** $\phi = d \wedge \phi \neq \text{truth}(c_m^i)$ **then** $\text{FP}[d] += 1$
(18) **if** $\phi \neq d \wedge \phi = \text{truth}(c_m^i)$ **then** $\text{TN}[d] += 1$
(19) **if** $\phi \neq d \wedge \phi \neq \text{truth}(c_m^i)$ **then** $\text{FN}[d] += 1$
(20) **end**
(21) **end**
end
//store the classifier, feature set and AUC
(22) $\psi^*[F_{\text{num}}] = \psi$
(23) $\omega^*[F_{\text{num}}] = \omega$
(24) $\text{AUC}[F_{\text{num}}] = \text{score}(\text{TP}, \text{FP}, \text{TN}, \text{FN})$
(25) **end**
//use the Kneedle algorithm to determine the number of features
(26) $F_{\text{num}}^* = \text{kneedle}(\text{AUC})$
(27) **return** $\psi^*[F_{\text{num}}^*], \omega^*[F_{\text{num}}^*]$

Algorithm 3: Activity-based vehicle PoI extraction (AVPE)—training stage.

inputs: $\mathcal{T}$, a set of n trajectories in the buffer, $\{t_1, \ldots, t_n\}$
λ, the merge threshold
ψ, the pre-trained classifier
ω, the feature set used in the classifier ψ
cluster, the chosen clustering algorithm, with pretrained parameters
output: $\phi \in D^+$, a set of clusters that are considered to be relevant
(1) $\mathcal{C}$ = preprocess (cluster, $\mathcal{T}, \lambda$)
(2) **for** $c_m^i \in \mathcal{C}$ **do**
//calculate features for the cluster from the vehicle signal values, features () is defined in Algorithm 2
//select feature values from the feature set and obtain prediction from classifier
(3) $\phi = \psi(\text{filter}(\omega, \text{features}(c_m^i)))$
//if the prediction is in the set of positive labels, add the cluster to our return set
(4) **if** $\phi \in D^+$ **then**
(5) $\mathcal{C}' = \mathcal{C}' \cup c_m^i$
(6) **end**
(7) **end**
//return a set of clusters that are considered to be relevant
(8) **return** $\mathcal{C}'$

Algorithm 4: Activity-based vehicle PoI extraction (AVPE)—deployment stage.

all clusters obtained from the trajectories in the buffer, and the set of clusters that are considered to be relevant is returned.

4. Experimental Methodology

In order to demonstrate and evaluate AVPE, we define a set of activity labels and transitions and select a set of vehicle signals to be used. In this section, we detail the implementation specifics of AVPE as evaluated in this paper, including the parameter values, data collection methodology, and attributes of the datasets used.

4.1. Activity Labelling and Transition Formulation. As described earlier in Section 3, the activity types and the transitions between these types are fundamental to AVPE. The set of labels and transitions provided in this paper is illustrative to enable our evaluation; however, the effectiveness of the specific definitions is dependent on the context, and they can be tailored depending on the application. For the PoI extraction task, we define 8 activity labels, guided by our industry partner in this work, when determining the activities of interest. Although introducing 8 labels increases complexity when compared to using binary classification (i.e., simply identifying true and false PoIs), classifying PoIs according to a more specific set of activities may be valuable in developing subsequent applications. This more nuanced set of classes may also help in understanding the reason why a given PoI was extracted, since a key motivation behind AVPE is that it can be used as a preprocessing step for applications such as destination prediction and categorizing vehicle usage. The class labels used in this paper are as follows, where (+ve) denotes that the instance should be considered as representing a PoI, i.e., a member of D^+, and (–ve) denotes that the instance should be considered as irrelevant for the purposes of PoI extraction, i.e., in $D \setminus D^+$.

(1) **Drive-through (+ve)**: an event that includes multiple stops and instances of slow movement, where the stops are for the driver to interact with a service.
(2) **Drop-off (+ve)**: an event in which the vehicle stops to allow passengers to exit.
(3) **Parked (+ve)**: a stationary period in which the vehicle is not driving, and this is the intent of the driver.
(4) **Pick-up (+ve)**: an event in which the vehicle stops to allow passengers to enter.
(5) **Barrier (–ve)**: an event in which the vehicle has to stop for the driver to interact in order to proceed past a closed barrier (such as a toll booth or parking barrier).
(6) **Driving (–ve)**: normal driving in free flowing traffic.
(7) **Manoeuvre (–ve)**: a period that involves slow movements with the possibility of stationary periods, high direction change, and reverse travel.
(8) **Traffic (–ve)**: where the vehicle has to move slowly as a consequence of external factors (such as roundabouts, traffic lights, congestion, or accidents).

We classify manoeuvre and barrier as negative labels, since, although they may indicate leaving or arriving at a PoI, they will always be adjacent to a positive label. We define separate labels for drop-off and pick-up as this can aid further applications that use the labelling, such as destination prediction.

In order to carry out the classification task, an accurate and consistent ground-truth labelling must be applied to the data. However, defining where the boundaries exist for each activity requires identification of the exact instance, in which transitions occur [30, 46]. Quantitative bounds were created to formalize the start and end instances for each label, alongside qualitative criteria (e.g., a drop-off event must include a passenger exiting the vehicle). We have defined a set of transitions for this paper, which can be used by other researchers in future investigations into PoI extraction. Examples of the transitions from the Driving activity to the

Table 2: Transition table from the driving activity to the next activity.

Next activity	Criteria
Barrier	When the vehicle first reaches the barrier, without any vehicle between itself and the barrier, and the vehicle speed first falls below 1 km/h
Drive-through	When the first toll booth (or order point) is reached and the vehicle speed first falls below 1 km/h
Drop-off	When the vehicle speed first falls below 1 km/h and a door to the vehicle is opened. Manual verification that a passenger is exiting the vehicle is required (from dashcam or seatbelt signals). The vehicle cannot be turned off for this label to be true
Manoeuvre	When the vehicle speed first falls below 1 km/h or reverse gear is selected. Manual verification that this is due to a manoeuvre is required
Parked	When the vehicle speed first falls below 1 km/h and the vehicle stops. The gear does not have to be in park, but manual verification that the stop is not due to a pick-up, drop-off or traffic is required
Pick-up	When the vehicle speed first falls below 1 km/h and a door to the vehicle is opened. Manual verification that a passenger is entering the vehicle is required (from dashcam or seatbelt signals). The vehicle cannot be turned off for this label to be true
Traffic	When the vehicle speed first falls below 5 km/h as a consequence of encountering congestion or road infrastructure that causes either 5 km/h not to be reached within 10 seconds or the vehicle speed to fall below 1 km/h within 10 seconds

other activities are shown in Table 2. Due to space limitations, we do not include the full set of transitions in this paper, but they are available at https://www.dcs.warwick.ac.uk/led. The use of such transition definitions ensures that the labelling process is reproducible and consistent across datasets. AVPE is agnostic to the set of labels used, and to use a different set of labels simply requires labelling of the ground truth according to the labels, and an appropriate set of transitions to be defined between the labels.

Table 3 shows the 22 vehicle signals that were used for activity classification. In this paper, the signals from the vehicle are expanded into the features described in Section 3 (minimum, maximum, range, etc.) resulting in a total of 99 features. The seatbelt status signals are the only binary signals for which a delta feature is generated. These signals were selected using domain expertise and guidance from our industry partner on common activities within a vehicle and how they relate to the available vehicle signals. For example, knowledge of the seatbelt and door status is key indicator of whether a passenger is entering or exiting the vehicle, and therefore, they are useful in identifying the current activity. Similarly, the lock status can be used as an indicator of a change of occupancy in the vehicle. Signals such as engine and stop-start status indicate whether the vehicle is stopping for a period of time, helping distinguish between manoeuvre and Parked events for example. Gear position, vehicle speed, and steering wheel angle can further provide insight into the vehicle's current activity. External data, such as traffic data from Application Programming Interfaces (APIs), and additional inertial measurements units could aid predictive performance; however, the aim of this paper is to use sensors that are already on the vehicle and are common across multiple vehicles, a motivation given by our industry partner. Additional sensors add cost to a vehicle, and traffic data APIs rely on data connectivity, which may not be available in some regions, and therefore, these are not considered in this paper.

4.2. Experimental Parameters. To use the AVPE algorithm, we are required to instantiate the algorithm with a number of parameters, including the set of activity labels (D), a value for the merge threshold (λ), a classification algorithm, and a feature selection algorithm. The activity labels used are defined in the previous subsection, and we investigate a range of merge thresholds (λ), namely, 0, 5, 10, and 20 seconds. We consider the Random Forest and Support Vector Machine (SVM) classification algorithms, since Random Forest classifiers have previously been used for transportation mode recognition [47–50], and SVMs have previously been shown to be effective for activity prediction [30, 51, 52]. When training the classifiers, we used a value of $k = 10$, for the k-fold cross validation. For simplicity, both classifiers use the default parameters in the library implementation used (we used the Weka library implementations of the Random Forest and SVM classifiers [53]), since tuning the classification is not the focus of this paper, and we found the default values to have reasonable performance. We used Minimal Redundancy Maximal Relevance (mRMR) for feature selection, since it has been shown to provide a compact subset of features that improves classification accuracy on both discrete and continuous data [44]. Both the classifier and feature selection methods can be replaced with alternatives if required (such as Bayesian Inference [54] or Principle Component Analysis [45], respectively), since our approach is agnostic with respect to the methods used.

4.3. Data Collection. In order to evaluate AVPE, we defined a data collection methodology and collected two datasets, namely, a scripted scenario dataset and a pattern-of-life dataset (the full scripted scenario dataset is available online at https://www.dcs.warwick.ac.uk/led. For privacy reasons, we are not able to publish it). The scripted scenario dataset comprises specific routes and activities (such as a pick-up) and certain locations. Each route follows a specified set of instructions and is repeated multiple times. This dataset has been made public for others to utilize in further research on vehicle PoI extraction and contains both GPS data and vehicle signals. The scenario dataset is used to train the classifier and evaluate the performance of AVPE. For our evaluation in this paper, the dataset is separated into training and testing sets over the journeys, with 65 journeys (5 routes) in the training set and 52 journeys (4 routes) in the test set. Cross validation occurs within the training set only (by separating it into training and validation sets), meaning

Table 3: Vehicle signals used for activity classification.

Signal	Type
Boot status (open/closed)	Binary
Door status (open/closed) [driver, passenger, rear right, rear left]	Binary
Combined seatbelt status	Categorical
Engine (on/off)	Binary
Gear position	Categorical
Indicator status	Categorical
Lock status	Categorical
Roof position	Categorical
Seatbelt status (buckled/unbuckled) [driver, passenger, rear right, rear left]	Binary
Steering wheel angle	Numerical
Stop-start status	Categorical
Vehicle speed	Numerical
Window position [driver, passenger, rear right, rear left]	Categorical

that all of the journeys in the test set are unseen, meeting the out-of-sample principle. The pattern-of-life dataset comprises journeys and activities undertaken, though an individual was using their own personal vehicle. The pattern-of-life dataset contains 76 journeys from 2 individuals, and all of the journeys are used for testing, since the classifier is trained on the scenario data, to demonstrate that AVPE is applicable to unscripted driving data.

For the scenario dataset, a set of 9 predefined routes were used with 2 different vehicles, with slight variations in routes between the vehicles. The first vehicle was a 2-door 4-seater convertible SUV, and the second was a 5-door 5-seater estate car. Each route was repeated 8 times in the first vehicle and 5 times in the second vehicle. This resulted in a dataset containing 117 journeys and 153, 698 instances, totaling over 1, 190 kilometers travelled and around 44 hours of data. Journey times were varied between peak daytime (07:00–10:00 and 16:00–19:00), nighttime (23:00–05:00), and off-peak daytime. The routes include sections of major and minor roads, with the shortest route lasting around 6 minutes on average, and the longest route around 40 minutes. The distribution of the journey durations and distances in the scripted scenario dataset is summarized in Table 4. Open air and multistorey car parks were used at shopping centers, railway stations, and a university campus, along with roadside parking. Other road structures included in the routes are drive-through services and barrier-controlled private roads. These routes ensure that data from a diverse set of road types and traffic conditions were collected. In all journeys, the vehicle contained a driver and passenger, with some journeys having 2 passengers. The front passenger seat was always used, and the rear passenger could sit in either outer seat. Not all routes contained every event type, and the frequency of events varied as can be seen in Figure 3, which shows the distribution of events within the dataset. The duration of a single event is dependent on the type of event, as shown in Figure 4, with the majority of events being around 20 seconds on average. Since there are differences in the event durations, the distribution of instances per event type differs slightly from the frequency distribution of event types, as can be seen in Figure 5 (when compared to Figure 3). Pattern-of-life data was also collected, with participants using the vehicles as part of their daily routine. Vehicle signals were collected for every journey in the data collection period, and grouped by participant ID. In this paper, to evaluate AVPE, we use 4 weeks of data from 2 different participants (1 week for each participant and vehicle combination) for our analysis. The pattern-of-life dataset contains 76 journeys and 101, 063 instances, totaling over 1, 215 kilometers and around 28 hours of driving. There were no scripted routes in this dataset; however, the routes still contain a mix of major and minor roads. The shortest journey lasted just over 3 minutes, and the longest journey took just over 100 minutes. The distribution of the journey durations and distances in the pattern-of-life dataset is summarized in Table 5. There were no requirements on any journey in the pattern-of-life dataset to have any passengers or contain specific activity types. A Vector GL2000 logger was used to record GPS data and signals from the vehicle Controller Area Network (CAN) bus. GPS data was recorded at 1 Hz, while CAN signals were broadcast to the logger and were generally recorded at a higher rate but were downsampled to 1 Hz as per the GPS data. We selected 22 vehicle signals that were considered relevant for predicting activities. The signals are a mix of binary, categorical, and real-valued attributes, as described in Section 3 and listed in Table 3. Every second, the last observed value for each of the CAN signals is used. Values older than 3 seconds for CAN signals and 60 seconds for GPS signals are discarded, and instances with missing values are removed from the dataset. In order to label the ground truth, we used dashcam footage to manually identify which activity is being performed for each instance.

5. Results

In this section, we first discuss the results of the state-of-the-art clustering algorithms, namely, CB-SMoT [7], STA [8], and GVE [9] when applied to our scenario dataset. Following this, we present the results of the activity classification stage and analyze the performance of AVPE as a whole.

5.1. Base Clustering of Trajectories. In this analysis, we use the parameters for each of the selected clustering algorithms that produced the highest set overlap. GVE [9] produces the highest number of instances, along with the most clusters. STA [8] gives the fewest clusters, and CB-SMoT [7] returns

Table 4: Summary of route durations and distances in the scripted scenario dataset.

Route	Duration (s)				Distance (m)			
	Minimum	Maximum	Average	Total	Minimum	Maximum	Average	Total
1	1558	3112	2433	31626	17686	30167	24219	314850
2	697	1139	917	11917	4860	6117	5318	69132
3	409	772	574	7458	3763	8760	5035	65459
4	706	1720	1077	14004	4766	4933	4832	62820
5	1620	2496	1996	25946	16219	27789	20775	270071
6	1031	2211	1451	18865	8412	8811	8496	110446
7	1621	2716	1967	25570	10815	14024	12082	157061
8	1046	1744	1348	17529	8731	8954	8869	115302
9	291	506	378	4908	1781	2116	1950	25349
	291	**3112**	**1349**	**157823**	**1781**	**30167**	**10175**	**1190490**

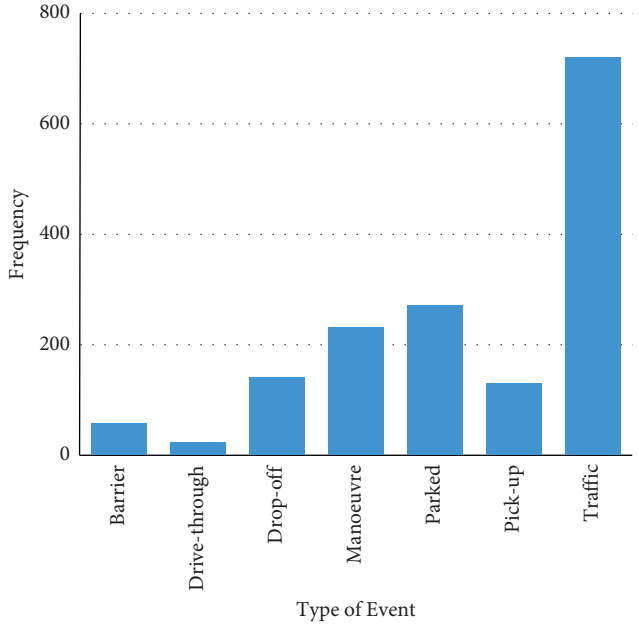

Figure 3: Distribution of the labels in the dataset, with the driving label (1248 events) omitted to aid readability.

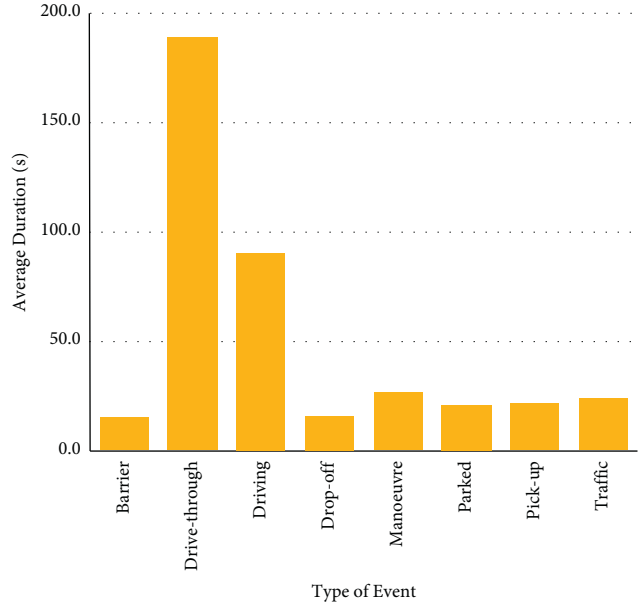

Figure 4: Average duration of events per label.

the fewest instances. When investigating these results further, we find some interesting properties, as discussed below.

GVE missed 28 clusters (or events), of which 2 were labelled pick-up and 26 parked. All of the missed clusters were less than 48 seconds in duration, with the highest durations being the pick-up labels. The parked labels that are not recorded are 2–13 seconds in duration, with the majority being located in multistory car parks with weak GPS signal, therefore showing false readings with a wide range of movement between consecutive instances. The other missed parked clusters all have a short duration, and this is likely to be a factor in the cause of these missed clusters. The missed pick-up events are 20–48 seconds in duration and occur directly after long parked events. The slight increase in movement compared to that of the parked events appears to prevent these pick-ups from being captured. Similarly to GVE, 31 clusters are missed by STA, 29 of which are parked events. The remaining missed clusters are 2 pick-up events, which are the same ones discussed above for GVE. The missed parked events are between 2 and 14 seconds in duration, and a similar combination of cluster length and GPS inaccuracy causes them to be missed. CB-SMoT misses 30 events in total, of which 26 are parked clusters, which varied in length between 4 and 15 seconds. Additionally, 2 drive-through events, lasting over 5 minutes, and 2 drop-offs were missed. Due to a different distance threshold being calculated for each trajectory, CB-SMoT is more sensitive to small movements. Similar trends to GVE and STA are also evident, where poor GPS reception gives the impression of high movement. Unlike the other clustering algorithms, CB-SMoT discards all clusters for two complete journeys. This behavior is not beneficial to PoI extraction, as multiple true PoIs are lost. CB-SMoT separates the input data into individual trajectories, where a distance threshold is calculated for each trajectory using the inverse cumulative probability. The distribution is created using the mean and standard deviation of the distances between consecutive instances. Both discarded journeys included a long wait at a drive-through service in addition to 2 drop-off events. This caused the mean of the distances to be low, whilst keeping a relatively high standard deviation. Using the best performing

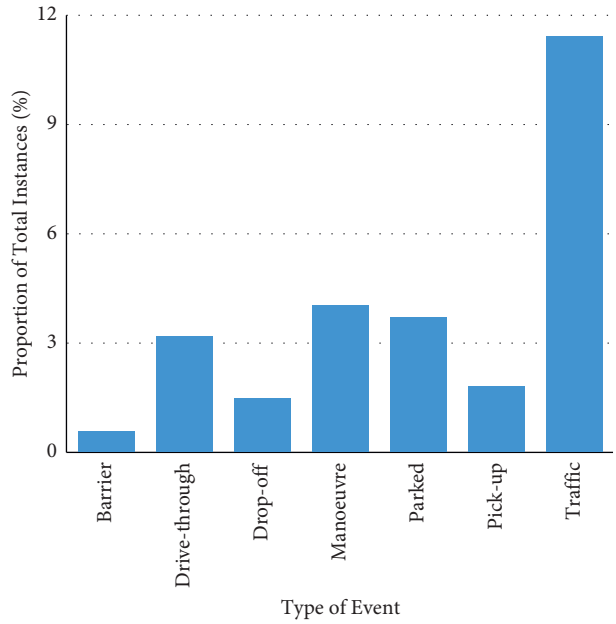

Figure 5: Proportion of the total instances per label, with the driving label (74%) omitted to aid readability.

Table 5: Summary of durations and distances in the pattern-of-life dataset per participant.

Participant	Duration (s)				Distance (m)			
	Minimum	Maximum	Average	Total	Minimum	Maximum	Average	Total
A	487	2948	1621	45387	1424	35930	22562	631735
B	189	6005	1162	55772	446	95626	12169	584102
	189	**6005**	**1392**	**101159**	**446**	**95626**	**17366**	**1215837**

parameters with this distribution caused a negative distance threshold to be generated, which resulted in no events being extracted for the two journeys.

When increasing the merge threshold, the number of false PoIs greatly decreases. This is due to combining fragmented clusters around true PoIs. However, as the number of false PoIs decreases, intuitively, the number of merged clusters also increases. Therefore, having a large merge threshold may not be beneficial, since this can distort the aggregated values in the cluster and reduce the number of true PoI extractions. Table 6 summarizes the performance of each clustering algorithm with 0-, 5-, 10-, and 20-second merge thresholds. There is a slight increase in false PoIs as the merge threshold in increased, since merging can combine multiple true PoIs resulting in a loss of precision. GVE is the favored algorithm of the three considered, since the priority in selecting a clustering algorithm is to minimize the number of false PoIs, and since we cannot recover them in the further stages of AVPE. The beneficial impact of merging clusters is most evident for GVE when using a 5-second merge threshold. Merging clusters within 5 seconds of each other reduces the number of false PoIs by 27.3%, without any further increase in false PoIs. Given that CB-SMoT discarded 2 entire journeys, it is unlikely to be the best algorithm to use for the clustering stage; however, for completeness, we consider all three clustering algorithms in our evaluation of AVPE.

5.2. Activity Classification and PoI Filtering. For our evaluation, we use the clustered scenario data from the best performing parameters for each of the clustering algorithms discussed above. For classification, we compare the use of Random Forest and SVM classifiers both using mRMR feature selection, as detailed in Section 3. Minimizing the number of features used is important as this will (i) avoid overfitting to the training data, (ii) increase the generality and simplicity of the classifier, and (iii) require less data to be collected and processed on the vehicle. With these factors in mind, it is important to consider the trade-off between the number of features and performance benefits. We used 10-fold cross validation on the training data to determine a suitable number of features to select when applying classification to the test data, using the Kneedle algorithm [10] as shown in Algorithm 3. The number of features selected varied between 11 and 30 for the Random Forest, and 6–34 for the SVM classifiers. The mean and standard deviation for the number of features were 15.8 and 6.0 for the Random Forest, and 16.0 and 7.5 for the SVM classifiers, respectively. Using Random Forest and CB-SMoT, the number of features increased as the merge threshold increased, while no such pattern exists for GVE and STA, in which both required fewer features than CB-SMoT. In contrast, the SVM classifier used the most features with CB-SMoT at $\lambda = 5$ and $\lambda = 10$. Similarly, a higher number of features are used for

Table 6: Summary of the performance of the clustering algorithms.

Algorithm	Merge threshold, λ (s)	# Clusters	# Missed PoIs	# False PoIs
CB-SMoT	0	896	30	578
CB-SMoT	5	705	32	412
CB-SMoT	10	638	32	362
CB-SMoT	20	560	36	293
GVE	0	1089	28	795
GVE	5	857	28	578
GVE	10	716	28	441
GVE	20	599	31	327
STA	0	845	31	569
STA	5	794	31	520
STA	10	696	32	425
STA	20	590	32	319

both GVE and STA when using 5- and 10-second merge thresholds.

Once classifiers are trained with the selected number of features, we apply them to the test set. Table 7 details the accuracy and AUC achieved for each combination on the test set. We consider AUC as our main performance metric as, unlike accuracy, AUC is not biased towards class imbalance. The confusion matrices in Figures 6 and 7 show a breakdown of the per-class accuracy for the Random Forest and SVM classifiers, respectively. It can be seen that STA does not cope well with drop-off and pick-up events, with a high misclassification rate between them. However, this is due to STA generally using fewer features, and therefore, the informative seatbelt features are not present. When using the SVM classifier, we can see that the drop-off and pick-up accuracy is much higher when combined with CB-SMoT or GVE clustering, when using λ values of 5 and 10. Additionally, a high misclassification rate between barrier and drive-through events can be seen; however, this is not unexpected given the similarity of these activities. From the confusion matrices, it is also apparent that the Random Forest classifier incorrectly predicts the traffic class at a much higher frequency than the SVM classifier. Although the AUC values when using the Random Forest classifier are generally higher than those when using the SVM classifier, the confusion matrices show that the SVM classifier appears to be better over all classes. Table 7 shows that the predictions using GVE clustering give the highest accuracy and AUC with both classifiers. Using the Random Forest classifier, GVE with no merging gives the highest AUC, whereas when using the SVM classifier, GVE with a 5-second merge achieves the highest AUC. We, therefore, take GVE with $\lambda = 0$ and GVE with $\lambda = 5$ as the highest performing combinations for the Random Forest and SVM classifiers, respectively. We use these combinations on unscripted pattern-of-life data later in this section to evaluate the applicability of AVPE to real-world data.

Table 8 lists the features that appear in the top 10, and their respective frequencies as selected by mRMR, over all base clustering methods and merge thresholds. The most prominent signals that appear are the vehicle speed, current selected gear, and the passenger door status. For the vehicle speed signal, the average feature is always in the top 10, with other combinations using the maximum, minimum, standard deviation, and time above average features. The feature for the average of the currently selected gear is always present in the top 10 across all combinations, with no other features derived from this signal appearing. The same applies to the passenger door status, with all combinations including the average feature within the top 10, in addition to some combinations using the minimum and range features. Features for the steering wheel angle signal are seen in the top 10 in all but the GVE and $\lambda = 0$ combination. The majority of combinations use the standard deviation feature of the steering wheel angle; however, some combinations also use the maximum, range, and time above average features. The average of combined seatbelt count signal is used in all combinations except for CB-SMoT with $\lambda = 5$ or $\lambda = 10$. Other signals with features within the top 10 are the indicator status, the engine running status, the stop start status, the passenger seatbelt status, the driver window position, and the central locking status. Figure 8 shows the improvement that AVPE gives over the three existing state-of-the-art algorithms alone. We define the percentage reduction as

$$\text{reduction}_{\alpha} = 100 - \frac{\#P^{*}_{\alpha}}{\#P_{\alpha}}, \tag{2}$$

where $\#P^{*}_{\alpha}$ is the number of PoIs output by AVPE, $\#P_{\alpha}$ is the number of PoIs output by the base clustering method, and α is the type of PoI to count (i.e., false or missed). The removal of false PoIs comes at the cost of increasing the number of missed true PoIs. While it is important to consider the reduction in true PoIs, the overall aim of AVPE is to provide a correct set of identified locations, rather than a complete set, and therefore, a reduction in true PoIs is acceptable.

When using the Random Forest classifier (see Figure 8(a)), with GVE and $\lambda = 0$, we see that 99.0% of false PoIs are removed, while losing 37.8% of true PoIs compared to the base algorithm. Using STA and $\lambda = 5$ removes 98.2% of false PoIs at a cost of losing 40.4% of true PoIs. Finally, using CB-SMoT with $\lambda = 0$ removes 94.3% of false PoIs, being the lowest reduction of the three clustering methods, with 35.5% of true PoIs being lost.

Figure 8(b) shows the same metrics for the SVM classifier. We see similarities to using the Random Forest

Table 7: Classification results, where * denotes the highest performing parameter combination, in terms of AUC, for each clustering algorithm and classifier combination.

Classifier	Algorithm	Merge threshold, λ (s)	# features	Accuracy	AUC
Random forest*	CB-SMoT	0	14	0.775	0.969
Random forest	CB-SMoT	5	15	0.718	0.955
Random forest	CB-SMoT	10	22	0.776	0.964
Random forest	CB-SMoT	20	30	0.717	0.956
Random forest*	GVE	0	23	0.810	0.978
Random forest	GVE	5	11	0.739	0.964
Random forest	GVE	10	12	0.737	0.957
Random forest	GVE	20	16	0.779	0.972
Random forest	STA	0	11	0.687	0.950
Random forest*	STA	5	13	0.766	0.971
Random forest	STA	10	12	0.765	0.961
Random forest	STA	20	11	0.698	0.940
SVM	CB-SMoT	0	8	0.746	0.931
SVM	CB-SMoT	5	22	0.770	0.933
SVM*	CB-SMoT	10	34	0.780	0.933
SVM	CB-SMoT	20	16	0.695	0.925
SVM	GVE	0	12	0.812	0.949
SVM*	GVE	5	21	0.864	0.959
SVM	GVE	10	20	0.828	0.947
SVM	GVE	20	16	0.803	0.952
SVM	STA	0	11	0.740	0.930
SVM*	STA	5	13	0.792	0.949
SVM	STA	10	13	0.789	0.943
SVM	STA	20	6	0.728	0.918

classifier, with GVE removing the most false PoIs, followed by STA and CB-SMoT. In general, the amount of false PoIs removed is lower, along with a reduced loss of true PoIs, apart from the case of CB-SMoT. Overall, the reduction of false PoIs is achieved to a reasonably high extent when using both Random Forest and SVM classifiers.

5.3. Applying AVPE to Pattern-of-Life Data. We apply AVPE to unscripted pattern-of-life data to investigate its generality and evaluate its effectiveness when applied to data from normal day-to-day driving, which may not exhibit the same patterns as the scenario routes. Additionally, we compare the performance of AVPE on the two different vehicles used. The classifier was trained using the scenario data, with a SVM as the classification method (since, in general, it outperformed Random Forest with the scenario data) and mRMR feature selection, as detailed in Section 3. The input clusters were generated using GVE with $\lambda = 5$, and both vehicles are used in the training and testing sets.

Participant A had 37 ground truth PoIs that were obtained in Vehicle 1. Of these, 24 were correctly extracted by AVPE, along with 7 false PoIs, and there were 13 missed PoIs. The missing PoIs were all parked events, while the false PoIs were barrier, manoeuvre, and traffic events. When using Vehicle 2, Participant A had 27 ground truth PoIs, with AVPE resulting in 3 false PoIs and 11 missed PoIs. The false PoIs were comprised of slow manoeuvre and traffic events. Once again, all missed PoIs were parked events, of which 3 were due to missing data as a result of poor GPS signal.

Based on the Vehicle 1 data for Participant B, 61 ground truth PoIs were identified. Using AVPE, there were 40 correct PoIs extracted. There was only a single false PoI, but 21 PoIs were missed in the process, 19 of which were parked events along with a single drop-off and a single pick-up event. In the case of Vehicle 2, 50 ground truth PoIs were identified for Participant B. After applying AVPE, 38 correct PoIs were extracted, along with 5 false PoIs. There were 2 false PoIs at a barrier, in which the stop-start queuing nature has similar qualities to a drive-through, 2 when performing slow manoeuvres, and another in traffic. Therefore, 12 PoIs were missed, 11 of which were parked events with a single drop-off event.

The results are summarized in Figure 9(a), which shows that AVPE with a SVM classifier and GVE ($\lambda = 5$) provides reasonable accuracy on pattern-of-life data. The number of false PoI extractions is very low, indicating that the classifier removes false PoIs with high accuracy. However, the number of missing PoIs is larger than expected, especially since the majority are parked events. Overall, it is clear that AVPE continues to provide a significant reduction in false PoIs, which is the aim of the method. Figure 9(b) illustrates this, with over 94.9% of false PoIs removed in the pattern-of-life dataset. This does come at a cost, however, with 22.4%–40.7% of PoIs missed from the output. These results are comparable to the performance seen in the scenario data (see Figure 8), showing the applicability of the method to unscripted driving.

To gain a deeper understanding of the decrease in performance, we consider the detail behind each missed event. The most common error, accounting for most of the missed PoIs, is the cluster size being too large. In multiple instances, a manoeuvre precedes the parked event, and a single cluster covers

FIGURE 6: Confusion matrices for random forest classifiers trained on the CB-SMoT, GVE, and STA clusters with different merge thresholds. (a) CB-SMoT ($\lambda = 0$). (b) GVE ($\lambda = 0$). (c) STA ($\lambda = 0$). (d) CB-SMoT ($\lambda = 5$). (e) GVE ($\lambda = 5$). (f) STA ($\lambda = 5$) (g) CB-SMoT ($\lambda = 10$). (h) GVE ($\lambda = 10$). (i) STA ($\lambda = 10$) (j) CB-SMoT ($\lambda = 20$). (k) GVE ($\lambda = 20$). (l) STA ($\lambda = 20$).

the majority of both. Due to the presence of features indicating a manoeuvre, such as the use of reverse gear and large steering movement, these clusters are classified as a manoeuvre event. The length of the manoeuvre event in the cluster is generally longer than the duration of the parked event; hence, the features (such as averages) are biased towards the qualities of a manoeuvre. Other variations include slow driving, drop-off, and traffic events, which are similar in nature, with 46 out of 57 (80.7%) missed PoIs being due to this type of error. A lack of GPS signal affected 8 (14.0%) of the missed PoIs. Due to the lack of any available GPS coordinates, data with missing instances are discarded before clustering. The final 3 errors (5.3%) are due to the GVE clustering algorithm not identifying the PoI as an area of low spatial movement and subsequently not generating a cluster. These could also be a result of GPS inaccuracies, giving the impression of greater movement.

5.4. Discussion. The misclassification that exists between drop-off and pick-up activities is partly caused by the cluster starting too late or ending too early, resulting in informative signals, such as the change in seatbelt status, being lost. Some investigation into extending the clusters for a given duration prior to the first instance was conducted; however, this was found to decrease the classification performance. It is possible that extending the cluster for all vehicle signals increases the difficulty in predicting the correct activity, since, for example, the average speed of the vehicle will increase dramatically if the cluster is extended prior to stopping. Additionally, the feature selection method could be failing to select informative features when these are calculated over extended clusters, since the vehicle signals now contain values from prior to the activity of focus. Further investigation of cluster expansion, using a lookback and lookahead, may help address this issue.

Intuitively, we might expect to see improved numerical performance if we consider activity classification as a binary problem, rather than having the 8 class labels used in this paper. An important motivation for AVPE is to be useful for applications such as destination prediction or driver profiling, and the vehicle activities used in this paper are defined

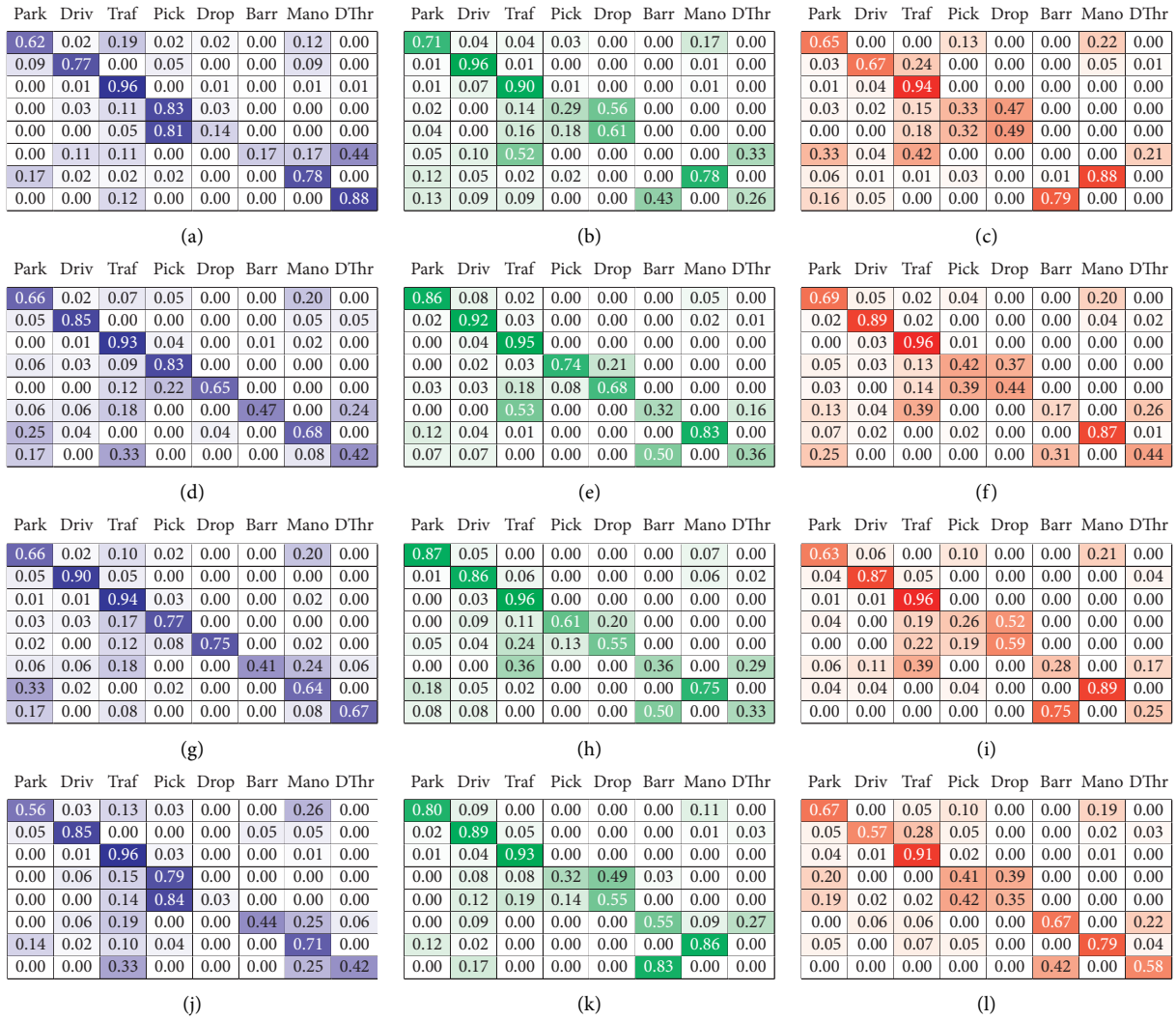

Figure 7: Confusion matrices for SVM classifiers trained on the CB-SMoT, GVE, and STA clusters with different merge thresholds. (a) CB-SMoT ($\lambda = 0$). (b) GVE ($\lambda = 0$). (c) STA ($\lambda = 0$). (d) CB-SMoT ($\lambda = 5$). (e) GVE ($\lambda = 5$). (f) STA ($\lambda = 5$) (g) CB-SMoT ($\lambda = 10$). (h) GVE ($\lambda = 10$). (i) STA ($\lambda = 10$) (j) CB-SMoT ($\lambda = 20$). (k) GVE ($\lambda = 20$). (l) STA ($\lambda = 20$).

Table 8: Frequency of features within the top 10 selected by mRMR for all 12 base clustering and merge threshold combinations.

Signal	Feature	Frequency
Gear position	Average	12
Vehicle speed	Average	12
Passenger door status	Average	12
Combined seatbelt status	Average	10
Steering wheel angle	Standard deviation	9
Stop-start status	Minimum	7
Driver window position	Average	7
Lock status	Average	7
Steering wheel angle	Range	5
Steering wheel angle	Time above average	4
Passenger door status	Minimum	4
Passenger door status	Range	4
Lock status	Minimum	4
Vehicle speed	Time above average	3
Engine (on/off)	Range	3
Driver window position	Range	3
Indicator status	Average	2

Table 8: Continued.

Signal	Feature	Frequency
Steering wheel angle	Maximum	2
Vehicle speed	Maximum	2
Vehicle speed	Minimum	2
Driver window position	Maximum	2
Vehicle speed	Standard deviation	1
Engine (on/off)	Minimum	1
Stop-start status	Range	1
Passenger seatbelt status	Delta	1

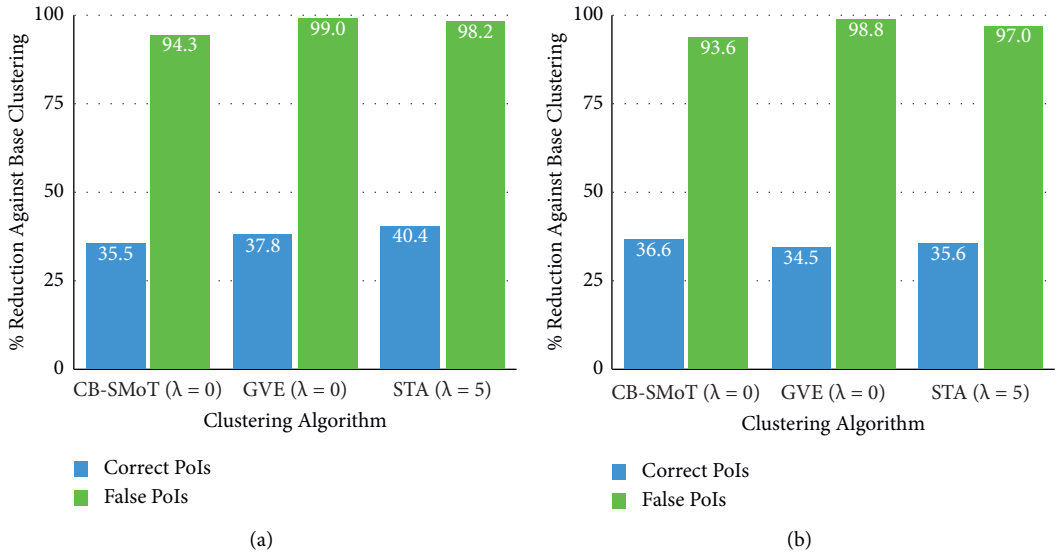

Figure 8: Reduction in PoIs when using AVPE compared to the base clustering methods. (a) Random forest. (b) SVM.

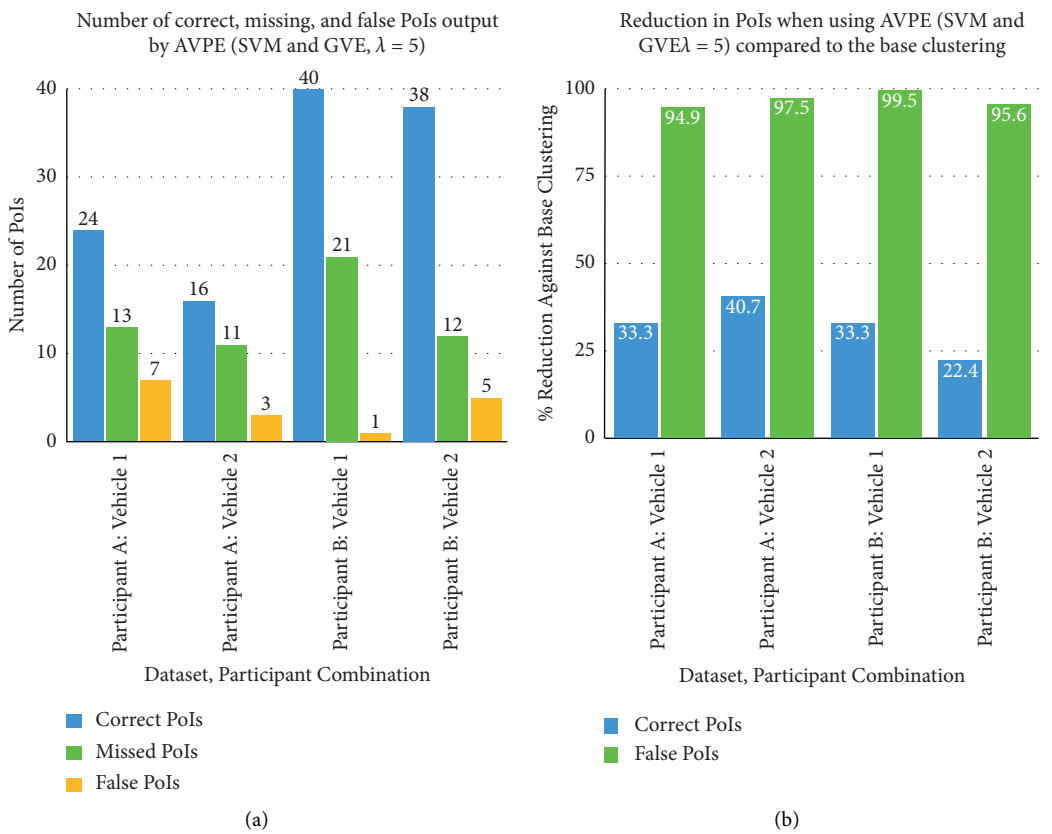

Figure 9: Results of the AVPE methodology applied to the pattern-of-life data, using 21 features.

with this in mind, based on guidance from our industry partner. Due to this, performing a binary classification is not appropriate, since an application such as destination prediction can make use of the reason why a period of low movement was classified as a PoI or not (e.g., stopping in traffic, although not considered a PoI, could be a key indicator for the next destination of the vehicle). Similarly, combining drop-off and pick-up activities would lose valuable information in the same way, since the difference between these two activities can have a significant impact on which future destinations are most likely.

6. Conclusion

In this paper, we show that AVPE, our proposed wrapper method that uses a classification stage based on vehicle data to filter out false PoIs, improves performance over the existing state-of-the-art clustering algorithms when extracting PoIs from vehicle data. We compared the performance of three base clustering algorithms, namely, CB-SMoT, STA, and GVE, and discussed the high amount of false PoIs output. We found that GVE with a 0-second merge threshold and a 5-second merge threshold gave the best performance in AVPE when using a Random Forest and SVM classifier, respectively.

In our scenario data, we observed that 94.3%–99.0% of false PoIs can be removed at a cost of 35.5%–40.4% of true PoIs being lost using a Random Forest classifier, using each of the three clustering algorithms. Similarly, when using an SVM classifier with each of the three clustering algorithms, 93.6%–98.8% of false PoIs can be removed while losing 34.5%–36.6% true PoIs. These figures show a reasonable trade-off between reduction in false PoIs against loss of true PoIs. When applied on an unscripted pattern-of-life dataset, AVPE saw comparable performance to that on the scenario data, with over 94.9% of false PoIs being removed. This shows that AVPE, which aims to ensure that any extracted PoIs are correct rather than aiming for completeness, gives a significant improvement over the current state-of-the-art clustering algorithms when used alone. This improvement can help assist the development of applications such as destination prediction [55–58] and identifying ride sharing opportunities [59, 60], which benefit from accurate PoI data.

Future work will investigate the impact of external data, such as Geographic information system (GIS) data, on the classifier to provide more context to the current surroundings of the vehicle. For example, if the vehicle is consistently near a drive-through service for the duration of the PoI, this could be used to inform the predicted activity.

Data Availability

The full scripted scenario dataset is available online at https://www.dcs.warwick.ac.uk/led. For privacy reasons, the pattern-of-life dataset is not published.

Conflicts of Interest

There are no potential conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by Jaguar Land Rover and the UK-EPSRC grant EP/N012380/1 as part of the jointly funded Towards Autonomy: Smart and Connected Control (TASCC) Programme.

References

[1] T. Andrade and J. Gama, "Identifying points of interest and similar individuals from raw gps data," in *Mobility Internet of Things*, pp. 293–305, Springer, Cham, Switzerland, 2020.

[2] R. A. Hamid and M. Sadik Croock, "A developed gps trajectories data management system for predicting tourists' poi," *Telkomnika*, vol. 18, no. 1, pp. 124–132, 2019.

[3] M. Luisa Damiani, H. Issa, and F. Cagnacci, "Extracting stay regions with uncertain boundaries from gps trajectories: a case study in animal ecology," in *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 253–262, Dallas, TX, USA, November 2014.

[4] X. Li, Q. Zhang, Z. Peng, A. Wang, and W. Wang, "A data-driven two-level clustering model for driving pattern analysis of electric vehicles and a case study," *Journal of Cleaner Production*, vol. 206, no. 1, pp. 827–837, 2019.

[5] M. Cañigueral and J. Meléndez, "Flexibility management of electric vehicles based on user profiles: the arnhem case study," *International Journal of Electrical Power & Energy Systems*, vol. 133, no. 1, pp. 1–17, 2021.

[6] J. Van Hinsbergh, N. Griffiths, P. Taylor, A. Thomason, Xu Zhou, and A. Mouzakitis, "Vehicle point of interest detection using in-car data," in *Proceedings of the ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery*, pp. 1–4, Seattle, WA, USA, November 2018.

[7] A. T. Palma, V. Bogorny, B. Kuijpers, and L. Otavio Alvares, "A clustering-based approach for discovering interesting places in trajectories," in *Proceedings of the 2008 ACM Symposium on Applied Computing*, pp. 863–868, Ceará, Brazil, March 2008.

[8] A. Bamis and A. Savvides, "Lightweight extraction of frequent spatio-temporal activities from GPS traces," in *Proceedings of the 2010 31st IEEE Real-Time Systems Symposium*, pp. 281–291, San Diego, CA, USA, December 2010.

[9] A. Thomason, N. Griffiths, and V. Sanchez, "Identifying locations from geospatial trajectories," *Journal of Computer and System Sciences*, vol. 82, no. 4, pp. 566–581, 2016.

[10] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a "kneedle" in a haystack: detecting knee points in system behavior," in *Proceedings of the 2011 31st International Conference on Distributed Computing Systems Workshops*, pp. 166–171, Minneapolis, MN, USA, June 2011.

[11] X. Cao, C. Gao, S. Christian, and Jensen, "Mining significant semantic locations from GPS data," *Proceedings of the VLDB Endowment*, vol. 3, no. 1–2, pp. 1009–1020, 2010.

[12] J. Krumm and E. Horvitz, "Predestination: where do you want to go today?" *Computer*, vol. 40, no. 4, pp. 105–107, 2007.

[13] R. Simmons, B. Browning, Y. Zhang, and V. Sadekar, "Learning to predict driver route and destination intent," in *Proceedings of the IEEE Intelligent Transportation Systems Conference*, pp. 127–132, Toronto, ON, Canada, September 2006.

[14] Yu Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from GPS trajectories,"

in *Proceedings of the International Conference on World Wide Web*, pp. 791–800, Madrid, Spain, April 2009.

[15] I. Keles, M. Schubert, Peer Kröger, S. Šaltenis, S. Christian, and Jensen, "Extracting visited points of interest from vehicle trajectories," in *Proceedings of the International ACM Workshop on Managing and Mining Enriched Geo-Spatial Data*, pp. 1–6, Chicago, IL, USA, May 2017.

[16] B. Furletti, P. Cintia, C. Renso, and L. Spinsanti, "Inferring human activities from GPS tracks," in *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*, pp. 1–8, Chicago, IL, USA, August 2013.

[17] M. Lv, L. Chen, Z. Xu, Y. Li, and G. Chen, "The discovery of personally semantic places based on trajectory data mining," *Neurocomputing*, vol. 173, no. Part 3, pp. 1142–1153, 2016.

[18] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, Portland, OR, USA, August 1996.

[19] J. A. Hartigan and M. A. Wong, "Algorithm as 136: a K-means clustering algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100–108, 1979.

[20] C. Zhou, D. Frankowski, P. Ludford, S. Shekhar, and L. Terveen, "Discovering personal gazetteers: an interactive clustering approach," in *Proceedings of the ACM International Workshop on Geographic Information Systems*, pp. 266–273, Arlington, VA, USA, November 2004.

[21] C. Zhou, D. Frankowski, P. Ludford, S. Shekhar, and L. Terveen, "Discovering personally meaningful places," *ACM Transactions on Information Systems*, vol. 25, no. 3, p. 12, 2007.

[22] K. Nishida, H. Toda, and Y. Koike, "Extracting arbitrary-shaped stay regions from geospatial trajectories with outliers and missing points," in *Proceedings of the ACM SIGSPATIAL International Workshop on Computational Transportation Science*, pp. 1–6, Seattle, WA, USA, 2015.

[23] D. Birant and A. Kut, "ST-DBSCAN: an algorithm for clustering spatial-temporal data," *Data & Knowledge Engineering*, vol. 60, no. 1, pp. 208–221, 2007.

[24] J. Hee Kang, W. Welbourne, B. Stewart, and G. Borriello, "Extracting places from traces of locations," in *Proceedings of the ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*, pp. 110–118, Philadelphia, PA, USA, 2004.

[25] Z. Fu, Z. Tian, Y. Xu, and C. Qiao, "A two-step clustering approach to extract locations from individual GPS trajectory data," *ISPRS International Journal of Geo-Information*, vol. 5, no. 10, p. 166, 2016.

[26] S. Vhaduri and C. Poellabauer, "Hierarchical cooperative discovery of personal places from location traces," *IEEE Transactions on Mobile Computing*, vol. 17, no. 8, pp. 1865–1878, 2018.

[27] Y. Chen, H. Yu, and L. Chen, "A spatiotemporal cluster method for trajectory data," in *Proceedings of the 2015 International Conference on Computer Engineering and Networks*, pp. 3–10, Harbin, China, December 2015.

[28] T. Bhattacharya, L. Kulik, and J. Bailey, "Extracting significant places from mobile user GPS trajectories: a bearing change based approach," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pp. 398–401, Redondo Beach, CA, USA, November 2012.

[29] T. Bhattacharya, L. Kulik, and J. Bailey, "Automatically recognizing places of interest from unreliable GPS data using spatio-temporal density estimation and line intersections," *Pervasive and Mobile Computing*, vol. 19, pp. 86–107, 2015.

[30] N. C. Krishnan and D. J. Cook, "Activity recognition on streaming sensor data," *Pervasive and Mobile Computing*, vol. 10, no. Part B, pp. 138–154, 2014.

[31] L. Liao, D. Fox, and K. Henry, "Location-based activity recognition," in *Proceedings of the 2006 Neural Information Processing Systems Conference*, pp. 1–8, Vancouver, British Columbia, Canada, December 2006.

[32] Y. Liu, Y. Mu, K. Chen, Y. Li, and J. Guo, "Daily activity feature selection in smart homes based on pearson correlation coefficient," *Neural Processing Letters*, vol. 51, no. 2, pp. 1771–1787, 2020.

[33] P. Rashidi, D. J. Cook, L. B. Holder, and M. Schmitter-Edgecombe, "Discovering activities to recognize and track in a smart environment," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 4, pp. 527–539, 2011.

[34] J. Ye, G. Stevenson, and S. Dobson, "Kcar: a knowledge-driven approach for concurrent activity recognition," *Pervasive and Mobile Computing*, vol. 19, pp. 47–70, 2015.

[35] A. Akin, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, and H. Paul, "Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: a survey," in *Proceedings of the 23th International Conference on Architecture of Computing Systems 2010*, pp. 1–10, Hannover, Germany, February 2010.

[36] A. Bulling, U. Blanke, and Bernt Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Computing Surveys*, vol. 46, no. 3, pp. 1–33, 2014.

[37] Ó. D. Lara, A. J. Pérez, M. A. Labrador, and J. D. Posada, "Centinela: a human activity recognition system based on acceleration and vital sign data," *Pervasive and Mobile Computing*, vol. 8, no. 5, pp. 717–729, 2012.

[38] X. Hong, C. Nugent, M. Mulvenna, S. McClean, B. Scotney, and S. Devlin, "Evidential fusion of sensor data for activity recognition in smart homes," *Pervasive and Mobile Computing*, vol. 5, no. 3, pp. 236–252, 2009.

[39] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[40] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[41] L. R. Dice, "Measures of the amount of ecologic association between species," *Ecology*, vol. 26, no. 3, pp. 297–302, 1945.

[42] T. A. Sørensen, *A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species and its Application to Analyses of the Vegetation on Danish Commons*, vol. 5, pp. 1–34, Kongelige Danske Videnskabernes Selskab, Copenhagen, Denmark, 1948.

[43] J. A. Ward, L. Paul, and H. W. Gellersen, "Performance metrics for activity recognition," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 1, p. 6, 2011.

[44] H. Hanchuan Peng, F. Fuhui Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[45] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[46] S. Amin, Y. Ren, and F. D. Salim, "Information gain-based metric for recognizing transitions in human activities," *Pervasive and Mobile Computing*, vol. 38, pp. 92–109, 2017.

[47] J. Li, X. Pei, X. Wang, D. Yao, Y. Zhang, and Y. Yue, "Transportation mode identification with gps trajectory data

and gis information," *Tsinghua Science and Technology*, vol. 26, no. 4, pp. 403–416, 2021.

[48] Z. Lu, Z. Long, J. Xia, and C. An, "A random forest model for travel mode identification based on mobile phone signaling data," *Sustainability*, vol. 11, no. 21, pp. 1–21, 2019.

[49] M. A. Shafique and E. Hato, "Classification of travel data with multiple sensor information using random forest," *Transportation Research Procedia*, vol. 22, no. 1, pp. 144–153, 2017.

[50] B. Wang, L. Gao, and Z. Juan, "Travel mode detection using gps data and socioeconomic attributes based on a random forest classifier," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 5, pp. 1547–1558, 2018.

[51] T. Huỳnh, U. Blanke, and B. Schiele, "Scalable recognition of daily activities with wearable sensors," in *Proceedings of the International Symposium on Location- and Context-Awareness*, pp. 50–67, Oberpfaffenhofen, Germany, September 2007.

[52] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," in *Proceedings of the Conference on Innovative Applications of Artificial Intelligence*, pp. 1541–1546, Pittsburgh, PA, USA, July 2005.

[53] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, Burlington, MA, USA, 2016.

[54] P. A. Flach and N. Lachiche, "Naive bayesian classification of structured data," *Machine Learning*, vol. 57, no. 3, pp. 233–269, 2004.

[55] L. Chen, M. Lv, and G. Chen, "A system for destination and future route prediction based on trajectory mining," *Pervasive and Mobile Computing*, vol. 6, no. 6, pp. 657–676, 2010.

[56] T. Minh Tri Do and D. Gatica-Perez, "Where and what: using smartphones to predict next locations and applications in daily life," *Pervasive and Mobile Computing*, vol. 12, pp. 79–91, 2014.

[57] J. Hu, S. Cai, T. Huang et al., "Vehicle travel destination prediction method based on multi-source data," *Automotive Innovation*, vol. 4, pp. 315–327, 2021.

[58] J. McInerney, S. Stein, A. Rogers, N. R. Jennings, and Jennings, "Breaking the habit: measuring and predicting departures from routine in individual human mobility," *Pervasive and Mobile Computing*, vol. 9, no. 6, pp. 808–822, 2013.

[59] N. Bicocchi and M. Mamei, "Investigating ride sharing opportunities through mobility data analysis," *Pervasive and Mobile Computing*, vol. 14, pp. 83–94, 2014.

[60] M. Enzi, S. N. Parragh, D. Pisinger, and M. Prandtstetter, "Modeling and solving the multimodal car-and ride-sharing problem," *European Journal of Operational Research*, vol. 293, no. 1, pp. 290–303, 2021.