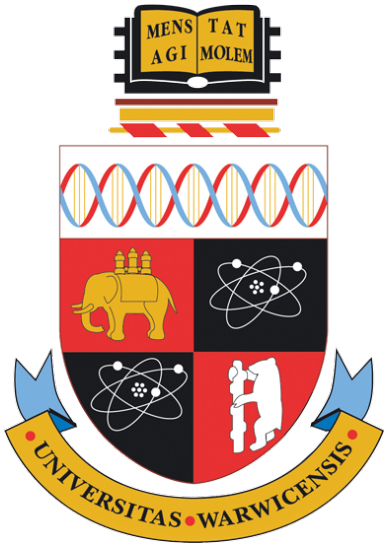
WARWICK
THE UNIVERSITY OF WARWICK




# Enhancing vehicle destination prediction using latent trajectory information

by

**James Van Hinsbergh**

**Thesis**

Submitted to the University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

**Doctor of Philosophy**

**Department of Computer Science**

April 2022

# Contents

# List of Tables

# List of Figures

# Acknowledgments

Firstly, I would like to give my upmost thanks to my supervisor Nathan Griffiths, who has given me direction, guidance and support throughout my time as a PhD student. His feedback and discussions have helped me shape the work in this thesis. I would also like to extend my thanks to my advisors, fellow members of the TASCC project, and to the participants that made the datasets I collected possible. I would like to thank members of the Department of Computer Science, in particular Roger Packwood, who has always been there to provide technical assistance with a smile, and helped facilitate the hardware to run my research on.

To my friends at the University of Warwick, in particular David Purser for his unparalleled support and encouragement, Matthew Bradbury, for the useful discussions and great company, and Ian Tu, for the enjoyment you created throughout the project. To Luke Evans, JinJue Lin and Louisa Lovatt, for the exciting experiences and the great friendships that kept me smiling throughout my studies, especially when times were hard. I would also like to thank Liam Steadman, Richard Kirk, and Helen McKay for their friendship and support throughout my time at the university. Additionally, I would like to thank Charlotte Frost for her motivation and support to get me through the last few months.

Finally, I am extremely grateful to my family, who have always believed in me and supported me in all my endeavours. Kim, Rob and Dan, your love and unconditional support has made this thesis possible. To Joanna, your never-ending belief always kept me going. Thank you for providing me with the encouragement needed to undertake this huge task, and for keeping me smiling throughout the process.

# Declarations

Parts of this thesis have been previously published by the author in the following papers.

[162] James Van Hinsbergh, Nathan Griffiths, Phillip Taylor, Alasdair Thomason, Zhou Xu, and Alex Mouzakitis. Vehicle point of interest detection using in-car data. In *Proc. of the ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery*, pages 1–4, 2018

This workshop paper describes the initial investigation conducted for Chapter 4, and is the basis for our PoI extraction method, AVPE.

[163] James Van Hinsbergh, Nathan Griffiths, Phillip Taylor, Zhou Xu, and Alex Mouzakitis. Classifying vehicle activity to improve point of interest extraction. *Mobile Information Systems*, 2021(1), September 2021

This paper presents AVPE, our method for PoI extraction for vehicle trajectories. Content from this paper forms the majority of Chapter 4.

[164] James Van Hinsbergh, Nathan Griffiths, Zhou Xu, and Alex Mouzakitis. Using trajectory sub-clustering to improve destination prediction. *Mobile Information Systems*, 2022(1), October 2022

This paper presents DPTS, our novel method for sub-clustering trajectories to improve destination prediction, and forms the majority of Chapter 6.

# Abstract

Intelligent transportation systems have the potential to provide road users with a range of useful applications, including vehicle preconditioning, traffic flow management and intelligent parking recommendations. The majority of these applications can benefit from knowledge of vehicle activities (common situations that a vehicle encounters e.g. traffic), along with the upcoming destinations that a vehicle will visit. We focus on the trajectories that vehicles provide, and the data contained within them, in order to ascertain information about the patterns in individuals' mobility data.

Machine learning has been used in many different vehicle applications, and we focus on using these techniques to predict the activity of a vehicle and its future destinations. Clustering methods can be applied at the level of trajectories or the individual instances within them, and we explore both of these alternatives in this thesis. Additionally, we explore several classification approaches to predict activities and destinations. In developing our methods, we make use of a combination of both geospatial and temporal data along with on-board vehicle sensor data.

This thesis presents novel methods for filtering stay points to identify points of interest and applying destination prediction to vehicle trajectories. Existing methods for stay point detection are not specific to vehicles, and therefore any region of low mobility is potentially considered to be of interest. We propose a novel method for filtering the extracted stay points to identify points of interest, using vehicle data to predict vehicle activities. The predicted activities are further used to represent trajectories as sequences of annotated locations, to inform the detection of similarities between journeys. Finally, this thesis presents a novel method for using additional properties of a trajectory to cluster trajectories into groupings of similar trajectories with the aim of improving the accuracy of destination prediction. We evaluate our proposed methods on a set of vehicle datasets, varying in purpose and the data available.

# Sponsorships and Grants

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.

# Acronyms

**API** Application Programming Interface.

**AUC** Area Under the Curve.

**AVPE** Activity-based Vehicle Point of Interest Extraction.

**BDP** Besse *et al.*'s Destination Prediction method.

**BIC** Bayesian Information Criterion.

**CAN** Controller Area Network.

**CB-SMoT** Clustering-Based Stops and Moves of Trajectories.

**DBSCAN** Density-Based Spatial Clustering of Applications with Noise.

**DPTS** Destination Prediction by Trajectory Sub-Clustering.

**GIS** Geographic Information System.

**GLONASS** Global Navigation Satellite System (product name).

**GMM** Gaussian Mixture Model.

**GNSS** Global Navigation Satellite System (generic).

**GPS** Global Positioning System.

**GVE** Gradient-based Visit Extractor.

**HMM** Hidden Markov Model.

**LED** Location Extraction Dataset.

**mRMR** Minimal Redundancy Maximal Relevance.

**OS** Ordnance Survey.

**POI** Point of Interest.

**POLD** Pattern of Life Dataset.

**SSPD** Symmetrized Segment-Path-Distance.

**STA** Spatio-Temporal Activities.

**SUV** Sport Utility Vehicle.

**SVM** Support Vector Machine.

**TASCC** Towards Autonomy: Smart and Connected Control.

**UWx** Unique Participant Identifier, number x.

**V2I** Vehicle-to-infrastructure.

**V2V** Vehicle-to-vehicle.

# Symbols

| | |
|---|---|
| $\mathcal{T}$ | The set of $n$ trajectories, $\{t_1, ..., t_n\}$ |
| $t_i$ | The $i^{th}$ trajectory within $\mathcal{T}$, a strictly ordered sequence of $\|t_i\|$ instances, $[x_1, ..., x_{\|t_i\|}]$ |
| $\mathcal{V}$ | The set of vehicle signals |
| $V_j$ | The values of the vehicle signals at time $j$ |
| $x_j$ | An instance $x_j$ is a latitude and longitude position, $lat, long$, and the values $V$ of vehicle signals at time $j$, $\langle lat, long, V_j \rangle$ |
| $V^R_{a,b}$ | A matrix of the real-valued signals in $\mathcal{V}$ for the time interval from $a$ to $b$, i.e., $(V_a, V_{a+1}, ..., V_b)$ |
| $V^N_{a,b}$ | 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 $b$ |
| $s^i_{a,b}$ | A strictly ordered subsequence of instances $\{x_j \| a \leq j \leq b\}$ in trajectory $t_i$, $[x_a, ..., x_b] \subseteq t_i$ |
| $\mathcal{C}$ | The set of clusters extracted from all trajectories in $\mathcal{T}$ |
| $c^i_m$ | The $m^{th}$ cluster of trajectory $t_i$, defined as a strictly ordered subsequence of instances $\{x_j \| a \leq j \leq b\}$, where $a$ is the first instance and $b$ is the last instance temporally, $s^i_{a,b} = [x_a, ..., x_b] \subseteq t_i$ |
| $\oplus$ | The $\oplus$ operator is used to denote the concatenation of two sequences |
| $F$ | The set of features that can be extracted from $\mathcal{V}$ |
| $L$ | The set of possible activities |
| $L^+$ | The set of positive activities, $L^+ \subseteq L$ |
| $\psi$ | A pre-trained classifier |
| $\omega$ | The feature set used in the classifier $\psi$ |
| $\phi$ | A prediction from the classifier $\psi$ |
| $\lambda$ | The merge threshold |
| $P$ | A probability value for a trajectory, $t$, fitting the most likely GMM in the set of GMMs, $X$ |
| $M$ | A sparse matrix containing the clustering parameters, with an implicit ordering to inform the order of clustering iterations |
| $\alpha$ | A parameter value for current iteration of clustering |
| $d$ | The distance function for clustering, which is used to calculate $D$ |
| $D$ | A dissimilarity matrix for the input set of trajectories |
| $\kappa$ | The maximum number of components for a GMM |
| $\mu$ | The maximum number of instances for training a GMM |

| | |
|---|---|
| $\theta$ | The parameter value to use for the decision threshold, i.e., a probability value in the static mode and a multiplier in the dynamic mode |
| $\theta_{dynamic}$ | A boolean flag indicating whether to use the dynamic (true) or static (false) mode for the decision threshold |
| $X$ | A set of $n$ GMMs trained for each cluster, $\{\chi_1, ..., \chi_n\}$ |
| $\chi_j$ | A pre-trained GMM for the cluster $c_j$ |
| $\nu$ | The best (lowest) Bayesian Information Criterion value for a GMM |
| $\mathcal{T}_{c_j}$ | A set of trajectories in cluster $c_j$ |
| $S_i$ | A vector of stay points |
| $\mathcal{S}$ | A set of vectors of stay points |
| $s$ | The stay point, defined as a strictly ordered subsequence of instances $\{x_j \mid a \leq j \leq b\}$, where $a$ is the first instance and $b$ is the last instance temporally. |
| $\Upsilon_t$ | A sequence of pre-processed data points for trajectory $t$, comprising of the location, activity and duration |
| $a$ | An activity mapping |
| $a$ | A set of activity mappings, $\{a_1, ..., a_r\}$ |
| $\omega$ | The activity mapping used in the classifier $\psi$ |

# Chapter 1

# Introduction

Human mobility is a vast area of research that is prevalent in society today [30, 66, 83, 108, 144, 150]. Advances in technology have made location-aware devices become commonplace [74], resulting in a wealth of data on individuals, with companies yet to find suitable applications for all the available data [2]. Examples of the use of location data include social media platforms, such as Facebook and Twitter, along with car insurance companies, and search engines, where the data collected will tailor or enrich the service the user receives [12, 112, 131]. This thesis will focus on human mobility within the vehicle domain, meaning that we only consider human mobility patterns from vehicle usage. In particular, we investigate private vehicles. We also consider data collected from taxis to allow for direct comparison to existing work.

With the rapid development of intelligent vehicles in recent years [31], the demand for location-aware applications will only increase as individuals become more dependant on the smart functionality they are now accustomed to from other services. This demand for intelligent vehicle functionality can come in a range of applications, for example, real-time parking information [28, 36, 176] and restaurant recommendations at the destination [46, 70, 89] are useful functions that an individual might desire from their vehicle. These applications all rely on human mobility patterns and utilising these to predict user-based information including their important locations and frequent trajectories. Applications such as ride-sharing opportunities [26, 47] and predicting the taxi demand in a given area [179] can harness the information contained within human mobility patterns. We define a point of interest, in the context of vehicle applications, as a point of low mobility (stay point) where the vehicle has stopped for an intended purpose e.g. to park or pick-up a passenger.

Applications that can predict unknown values typically rely on machine learning, where a pre-trained classifier can read input data and assign a label based on previous data. Important considerations include assessing the cost in terms of both computation time and storage, and finding a trade-off between this cost and performance [153]. Machine learning has been used for vehicle applications such as predicting future destinations [9, 23, 99, 149, 174, 175], estimating traffic flow [40, 115, 116, 132], and fault detection [69, 87, 140, 146].

Some of these applications utilise vehicle telemetry data, which is typically obtained from the Controller Area Network (CAN) bus, or equivalent. The CAN-bus adopts a message-based protocol, allowing each device to broadcast messages at their own defined intervals [58], and ensuring that all signals are received. This allows for high-frequency data capture, giving precise measurements for the vehicle's systems. Data loggers can record this information, along with GPS, allowing for the collection of high quality vehicle trajectories.

The focus of this thesis is the destination prediction process. This process encompasses several steps: data collection, feature curation, trajectory and stay point extraction, automated filtering and automated prediction. The following subsection details the contributions of this thesis, which advance and improve the state-of-the-art techniques used in the destination prediction process.

## 1.1 Problem Statement & Contributions

This thesis aims to propose new data-driven approaches that harness the latent information contained in vehicle trajectories, whether that be from vehicle telemetry, spatial or temporal data. Specifically, the problem statement is: **Can we devise new techniques that either utilise the data in a different way, or consider different data, within vehicle trajectories to improve applications such as point of interest extraction and destination prediction?** In order to investigate our proposed problem, this thesis provides the following contributions.

1. **Removing false positive points of interest within vehicle trajectories.** Previous investigations into point of interest (PoI) extraction did not consider points of interest specifically within vehicle trajectories. Vehicle trajectories have different characteristics to other types of trajectories, the most important of which is that stopping in a location does not imply that this is the intent of the individual, as it may in fact be a consequence of getting to their destination. An example of this could be stopping at a toll both, which is necessary to get to the destination, but in itself is unrelated to the purpose of the journey. The consequence of this is that current state-of-the-art approaches generate false PoIs [18, 126, 158]. We propose Activity-based Vehicle Point of Interest Extraction (AVPE), an algorithm for extracting points of interest within vehicle trajectories, that filters out false positives by classifying the activity of the vehicle. We define vehicle activity to be common situations or tasks in which the vehicle encounters, for example, waiting in traffic or performing a manoeuvre.

2. **Using sequences of activity-annotated stay points to predict**

**destinations.** Within a vehicle trajectory, there may be several activities performed at stay points, such as dropping off a passenger or making a turn-in-the-road. These activities can provide extra context about the trajectory they are within, and information on activities may help in predicting the destination of the vehicle. We propose a method to annotate and group stay points within vehicle trajectories, to compare sequences and predict destinations. As part of this method, we propose using multiple activity groupings for training, since the effectiveness of using activities to predict destinations may vary depending on the input data.

3. **Grouping vehicle trajectories by their additional properties.** Decomposing data into smaller groupings can result in the data within each group being more similar to each other. This notion can be applied to trajectory classification and, by extension, destination prediction. Further grouping of trajectories could result in a lower prediction error for the final destination. We propose Destination Prediction by Trajectory Sub-clustering (DPTS), which performs multi-layered clustering on trajectory data, which is an extension built on the work of Besse *et al.* [23]. Additionally, we propose two types of decision threshold to assess the certainty of prediction, and enable a more general prediction to be made if this threshold is not met.

4. **Collection of two vehicle datasets for point of interest and pattern of life experiments.** To enable the study of vehicle trajectories within this thesis, two datasets were collected, namely, the Warwick Location Extraction Dataset (LED) and the Warwick Pattern of Life Dataset (POLD). Both datasets contain samples from two different vehicles, and consist of GPS data alongside all the signals available on the vehicle. To create the LED, a specific data collection protocol was devised, with scripted routes and actions for investigating PoI extraction. The LED has been made publicly available for use in future research on PoI extraction[1]. The POLD consists of segmented data for multiple drivers over weekly or fortnightly periods. As with the LED, full vehicle data alongside GPS was available in this dataset. This dataset is not publicly available due to the sensitive nature of the participant's routines and locations, but has been used to support the evaluation in this thesis.

## 1.2 Structure

The thesis is structured in the following chapters.

[1] *https://www.dcs.warwick.ac.uk/led*

**Chapter 2** provides an overview into the related work surrounding vehicle trajectory data, including existing clustering methods for PoI extraction and approaches to destination prediction.

**Chapter 3** presents the Warwick Location Extraction Dataset (LED) and Pattern of Life Dataset (POLD), that have been collected as part of this work, along with other trajectory datasets that we utilise for our experiments. This chapter details the preparation and collection procedures used to create the LED and POLD.

**Chapter 4** presents Activity-based Vehicle Point of Interest Extraction (AVPE), a wrapper method around existing PoI extraction methods, that introduces the notion of vehicle activity classification, and uses the predicted activity as a method to filter out false positives. AVPE uses signals for the vehicle, such as the gear position and door status, within the proposed activity classification, which provides further insight into the intent of the driver of the vehicle. The performance of AVPE is compared to an existing state-of-the-art method on the LED, in addition to evaluating the performance on unscripted data from the POLD.

**Chapter 5** investigates the use of activity annotations within vehicle trajectories, to establish whether such annotations can assist with predicting the destination of the vehicle. The proposed method includes the concept of aggregating individual activities together into groups, to establish whether different combinations of activity labels are beneficial for different individuals. The impact of adding activity annotations and using these to aid destination prediction is compared to existing methods that solely rely on the geospatial data.

**Chapter 6** introduces Destination Prediction by Trajectory Sub-clustering (DPTS), an extension to an existing method, presented by Besse *et al.* [23], that aims to reduce the prediction error by further grouping trajectories into smaller clusters. DPTS includes a parameter matrix, in which further sub-clustering can be easily performed, along with a decision threshold that evaluates the sub-cluster classifier against that of the baseline, and can revert to the original prediction if the performance is not adequate.

**Chapter 7** concludes the work presented in this thesis and outlines future research within this area.

# Chapter 2

# Background & Related Work

Intelligent transportation systems present opportunities to enhance the way we use our vehicles. Modern vehicles come with onboard systems and numerous sensors providing a vast amount of data. With this data, opportunities arise to further understand vehicle trajectories, utilising on-board data to enrich the information surrounding a given trajectory.

In this chapter, we present the background and related work for this thesis. The topic of intelligent transportation systems is introduced in Section 2.1, before considering an overview of the research into human mobility in Section 2.2. After considering a general overview of intelligent transportation systems and human mobility, destination prediction is explored in Section 2.3. Section 2.4 introduces the technologies used in data collection and how these form trajectories. The related work surrounding point of interest extraction is described in Section 2.5, and finally we discuss existing research on activity recognition in Section 2.6.

## 2.1 Intelligent Transportation Systems

The subject of intelligent transportation systems (ITS) has been a key topic of research in recent years. As new technologies emerge, products with intelligent automation and prediction capabilities have increasingly become commonplace in society. Artificial intelligence and machine learning techniques underlie the development of these new technologies. A strong focus has been on home automation tools, leaving other areas without significant development.

Destination prediction is an active area of research, especially when used in conjunction with ITS, and is the focus of this thesis. Krumm and Horvitz investigated real-time destination prediction for car journeys whilst on the move [99]. They used Bayesian AI to predict the stopping point of a trajectory in a kilometre square region, taking a subset of available road and trajectory properties in order to build the learning model. Simmons *et al.* also developed a similar system, using Markov Models with a graph representation of the road system, which would predict the subsequent edge that a vehicle would take [149].

Knowing the destination in advance can assist with route planning, which aims to provide the best possible route for either the driver or the system as a whole. This links closely with traffic assessment and traffic flow efficiency, since techniques for each can have mutual benefit when used together. This area has seen multiple approaches for real-time re-routing. For example, Wang *et al.* [170] investigate the use of localised vehicle-to-infrastructure (V2I) communications to inform approaching vehicles of unexpected traffic conditions along the route, while Adler and Blue model drivers as cooperative agents with the goal of finding a solution that best accommodates the needs of all parties [5].

The assessment of traffic conditions has been considered by many researchers. Using vehicle-to-vehicle (V2V) communications, Ma *et al.* demonstrated that lane-changes and speed profiles could help characterise incidents with relatively high classification accuracy [116]. In 2011, Bauza and Gozalvez proposed Cooperative Traffic Congestion Detection, a mechanism that attempts to combine previous attempts into one with greater functionality. This approach includes information on traffic density, helping to improve overall prediction accuracy for scenarios with dense traffic conditions [21].

Whilst traffic assessment can provide advanced warning of problems in traffic flow, it cannot solve the initial congestion, a serious issue that can have consequences both economically and in terms of safety. Congestion is caused by exceeding the capacity of the road and by disruption in traffic flows. By using V2V communications, research has been carried out in multiple areas including streamlining the flow at signalled junctions, allowing cooperative merging on motorways, and improving driving efficiency when using multi-lane roads. Butakov *et al.* model each driver individually, allowing for a trade-off between fuel economy and journey time [35]. When considering the issue of traffic flow disruption, recent research has focused on maintaining suitable gaps to anticipate changes in condition, enabling early decisions and smooth changes in velocity [84, 142].

Techniques such as platooning, where multiple vehicles travel close to one another at a set speed, can be a method of improving traffic flows. Driving in a platoon can also reduce fuel consumption compared to driving separately, particularly for heavy-goods vehicles (HGVs). Due to this, platooning, most notably in HGVs, has become an active area of research interest. In 2010, Alam *et al.* carried out experiments on a Swedish highway, investigating the effects of a 2-truck platoon on fuel consumption [7]. Larson *et al.* proposed a framework to assist the formation of platoons [103], since this has been the focus of little research.

The key motivation for our focus on destination prediction in this thesis is that having knowledge of a vehicles destination could improve the effectiveness

of specific intelligent transportation systems, even if the prediction is only partially correct (i.e., only valid for the next few steps of the journey). Examples of applications that could benefit from these predictions include route planning, traffic assessment, traffic management and platooning, which we discuss in this section.

### 2.1.1 Route Planning

Route planning can encompass many aspects of agent based systems and vehicle communications. The goal is to advise the driver of a route which is going to be most desirable, whether in terms of travel time, traffic disruption or type of route.

Wang *et al.* considered re-routing vehicles at the point of approaching unforeseen congestion in an urban environment [170]. It was proposed that traffic lights could be altered, through fitting V2I communications hardware, allowing interaction between the lights and vehicles. The aim of the protocol proposed by Wang *et al.* was to provide a recommendation to mitigate the congested area, and not change the entire route. The decision to only change parts of the route was taken from an efficiency perspective as it is logical for the control of the traffic lights to only be concerned with congestion in their immediate vicinity due to the extra complexity and communication overheads that might otherwise be involved.

Wang *et al.* also considered the balance between selfish and altruistic solutions, enabling re-routing guidance to benefit the road system as a whole whilst also providing useful recommendations to each individual driver. This system view is shared in other research, with de Souza *et al.* highlighting the importance of maximising the spatial utilisation of the network [51]. Whilst such a recommendation system does not have control over the driver's actions, Adler and Blue reinforce that compromises must be made in order for a satisfactory solution to be obtained [5]. In their simulations, Wang *et al.* reported that overuse of selfish re-routing mechanisms had a significant negative impact on the overall traffic conditions [170].

It is common for the road network to be modelled as a graph [51, 92]. For example de Souza *et al.* [51] generated weights for edges, representing the current road classification, and used the k-nearest neighbours technique to classify the roads, based on an estimate of road density and the average speed. Upon detection of a congested road, an area of interest is declared and re-routing commences, with load balancing used to ensure that additional disruption is kept to a minimum. Following the approach that Wang *et al.* proposed, the road-side units do not attempt to modify the route outside of the affected area [170].

Overall, Wang *et al.* found that their method yielded an average of a 38% improvement in travel time [170] compared to de Souza *et al.* who achieved a reduction in stopped time, albeit with an increase in overall journey distance [51]. In the work of Kim *et al.*, Markov policies were used to help with vehicle routing [92]. Their investigations found that combining real-time and historical data could greatly reduce overall vehicle usage during periods of high congestion.

The issues surrounding route planning include finding a suitable trade-off between selfish and altruistic approaches, whilst providing useful recommendations in a real-time manner. Including historical data has shown improvements in reduced vehicle usage, however an increased amount of integration with this data could be beneficial. Other solutions that include road-side units in an urban environment have not considered this data and therefore further investigation into its use is warranted. In addition, new methods for traffic assessment may further increase the performance of the aforementioned systems, since early and more reliable detection can be useful. An application such as route planning could benefit from knowledge of vehicle's destinations in advance, enabling timely notifications to the driver including when important information regarding the destination is received or an alternate route is advised.

### 2.1.2 Traffic Assessment

Data from the continuous assessment of traffic conditions can support a range of applications relating to route planning and the development of transportation networks. Recently, research into effective traffic assessment mechanisms has been undertaken, particularly on multi-lane roads such as motorways. In 2009, Ma *et al.* investigated the use of Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs) to assess traffic conditions in real time [116]. Their work highlighted existing research into V2V and V2I communication and the methods that exist for automatically monitoring traffic conditions. Unlike previous work, Ma *et al.* proposed using lane-changes and speed profiles to characterise incidents. Overall, it was concluded that the SVM approach worked with relatively high performance in a simulated environment, especially when compared to the baseline California algorithm, a popular and easy-to-implement incident detection algorithm [128].

CoTEC (COperative Traffic congestion detECtion) was proposed in 2011 by Bauza and Gozalvez [21]. CoTEC is a traffic congestion detection mechanism that attempts to use a combination of techniques to provide complete coverage, including the ability to monitor conditions, obtain a consensus among V2V enabled vehicles and limit the transmission of redundant information. In 2014, Yuan *et al.* investigated the use of beacons to determine the status of

neighbouring vehicles [178], using a similar implementation to CoTEC.

The level of congestion can be calculated using traffic density and average speed, in which some approaches achieve consensus [21], and others append their own estimate to the transmission [178]. The traffic condition reports are propagated back from the front of the queue, employing duplicate detection, and only exchanging data in abnormal traffic conditions, therefore limiting the communication overheads [21, 178]. The final reports could be used either to alter the route guidance of approaching vehicles or to connect to infrastructure to provide long-range detection.

Overall, CoTEC can accurately detect congestion and give indicators of the intensity and length of congestion, allowing the potential for intelligent route modification using this collected information. The performance measures when compared against induction loops are generally of similar effectiveness, but CoTEC outperforms the infrastructure-based system if the loop separations are over 300m. However, it is worth noting that these results assume 100% penetration, which is not representative of real world scenarios. Decreased penetration results in underestimation thus resulting in significant degradation in performance. The work of Yuan *et al.* is also promising, with shorter delay times on multi-lane roads when simulated [178].

It is important to consider the trade-off between communication overhead and detection accuracy, and more work could be done to ensure that accuracy does not suffer as a consequence of reducing communication overhead. Additionally, using existing systems to boost the detection accuracy may be beneficial. For example, incorporating data from induction loops with data from vehicles may be effective. This would require the use of V2I communications linked with the induction loops, and the feasibility of this should be investigated. More importantly, the effect of decreased vehicle penetration and data loss on classification accuracy needs further investigation.

The research into traffic assessment and cooperative congestion detection can provide useful information to surrounding vehicles, however further benefits could be gained by providing this data to a wider range of vehicles. Vehicles with destination prediction capability could receive the traffic assessment data whilst in the earlier stages of their journey, resulting in vehicles having enough time to change route and not add to the current congestion.

### 2.1.3 Traffic Flow Efficiency

Whilst a lot of research has gone into assessing traffic conditions, it is also of equal importance to explore techniques that improve the flow of traffic. This may have a positive impact on traffic condition assessment and could also link between the assessment and route planning applications to improve individual

journeys. Research in this area mainly focuses on driver recommendations, giving these based on the output of predictive models.

Butakov *et al.* discussed a method for minimising the change in speed when approaching signalled intersections, assuming that the location and timing of traffic lights and traffic flow speed would be available ahead of time through V2I communication [35]. At the same time, a traffic operations centre would provide average traffic speed and density information through the same communication link. Based on this, they create a model for vehicle energy, which uses the forces applied to the vehicle at a given point to estimate the energy usage.

Asadi and Vahidi investigated a similar situation that used forces from the road and engine to track velocity [13], much like in the work of Butakov *et al.* [35]. Rule sets were used in order to calculate a target velocity that should be matched so that the vehicle arrival coincided with the green phase of a given light. Once again, this timing information was assumed to be available ahead of time, and broadcast so that vehicles could receive this data. In Butakov *et al.*'s approach, amber lights are modelled as red because under normal circumstances it is not desirable to travel through a amber light [35]. The system also used real driving data, such as acceleration and deceleration profiles, to add personalisation, calculating a trade-off between arrival time and fuel economy, which was tuned to each individual driver.

Butakov *et al.* used Monte Carlo simulations, generating 1000 cases with different parameters for both driver style and route characteristics (such as traffic light patterns). An average of a 29% reduction in energy consumption was found, with only 15 out of 1000 cases requiring more energy than without the recommendation. Asadi and Vahidi simulated multiple scenarios for suburban driving and multi-vehicle encounters [13]. They concluded that communicating the traffic light state to vehicles aided fuel consumption, with experimental results showing an average increase of 8.3 miles per gallon over 6 simulated vehicles.

Another use-case in which traffic lights can be used is to control the entry of vehicle onto a motorway. Scarinci *et al.* explored the problem of not being able to find a suitable gap to merge when joining a motorway [142]. Previous research has highlighted this issue as one of the key causes of breakdowns in traffic flows, which can be dangerous to road users. Current systems use inductive loops and traffic lights at the slip road but this does not involve any cooperation between vehicles.

Scarinci *et al.* proposed a system called Cooperative Merging Assistance (CoopMA) which aims to release platoons of vehicles into specifically created gaps on the motorway [142]. This is achieved by reducing the speed of a vehicle on the motorway through some method of vehicular communication. Gaps will be created between cooperating vehicles at the front of the platoon and the

last vehicle in the next platoon upstream. By calculating these trajectories it is possible to determine the time and space in which the platoon vehicles will need to compact. No intelligent vehicles are needed on the slip road, however V2I communications need to occur between the motorway vehicles and the infrastructure in order to get their position and to receive a request to slow down and create a gap. Multiple simulations were carried out, showing a positive effect on the reduction in congestion levels. However, techniques for dealing with the lack of cooperating vehicles being available or inaccuracies in estimations are not considered. An alternative approach to minimising speed and creating a gap where slip roads merge, is to ensure early and planned lane change manoeuvres, which could improve fuel efficiency and flow on multi-lane roads [84].

These approaches focus on either recommendations to the driver or instructions to the vehicle a short period of time prior to the arrival of the vehicle. Having prior knowledge of a vehicle's destination could allow these traffic flow applications to provide information to the vehicle further in advance, which could improve efficiency, allowing the driver or vehicle more time to act.

### 2.1.4 Platooning

Another benefit for traffic flows may be platooning, where groups of vehicles travel together at close distances. This can have a positive influence on fuel economy. Larson *et al.* investigated the coordination of platoon formations by developing a system of virtual controllers, normally located at junctions [103]. They analysed the amount of fuel saved using a large-scale simulation of part of the Autobahn network in Germany. The coordination process is only started if the fuel savings are greater than the overheads to form the platoon. The process does not necessarily alter route information, but instead slightly reduces a vehicle's speed in order to facilitate platoon formation. This can be done in real-time using only location, speed and destination information.

With a few thousand participants, the simulated data achieved a reduction in fuel consumption of over 5%. However, the authors noted that fuel consumption reduction generally increases with the number of participants. Alam *et al.* also investigated fuel consumption reduction in platoons, focusing on heavy goods vehicles (HGVs) [7]. They concluded that the amount of fuel consumption reduction was dependant on both the time gap and the load of the lead vehicle. The results showed that trucks of identical weight enabled the highest amount of fuel reduction, in which the smallest time gap was used. The time gap was a preset function as part of the user configurable adaptive cruise control (ACC), that alters speed to maintain the desired separation to the vehicle in front. The smallest gap gave a 7.7% reduction when platooning with

identical trucks. These tests were simulated with the use of a modelling tool called Dymola [32]. A further experiment was also carried out on a Swedish highway using two identical trucks.

If platoon formation is recommended, a framework for maintaining the control of the platoon is required. Qian *et al.* investigated a predictive framework to manage the formation of the platoon [133]. Models for the road and other objects must be considered since other vehicles, not in the convoy, will be present. A virtual structure is proposed, maintaining a reference path, upcoming obstacles, road curvature and speed limits. Amoozadeh *et al.* also conducted research regarding platoon management protocols [10]. Their proposed method supported three basic operations, allowing for merge, split and lane change functionality. To achieve other, more complex, operations the three basic operations are used in varying combinations. The authors constructed seventeen individual micro commands, which provide the underlining functionality of the protocol.

Deeper analysis of the results found that larger platoon sizes and smaller inter-platoon spacing have a positive affect on overall throughput. Additionally, the simulations found that the most costly operation by a large margin was the merge operation. Simulations demonstrated the value of their approach, although perfect communication will not happen in real scenarios, and the impact of this has not been quantified. Ozbilgin *et al.* investigated the trade-off between cost and effectiveness of data shared within platoons [125].

Platooning at traffic lights is also an area of consideration, and Gunther *et al.* developed a rule-based algorithm for this [68]. It takes a follow-leader principle but also considers non-V2X vehicles. SUMO [96] and OMNeT++ [166] are used to simulate this technique, considering the required penetration rate of vehicles with communication capabilities in order to have a significant increase in traffic efficiency. Other work shows increased throughput by displaying the time until the next green phase of the upcoming traffic light to the driver [13, 35]. To group approaching vehicles, V2X communication is used [142], however for this to be possible all vehicles must have V2X communication capabilities. The role of a vehicle is evaluated to identify its abilities and those of the surrounding vehicles, which will directly impact on what the vehicle can achieve [68]. To test their rule-based algorithm, Gunther *et al.* performed simulations of a 4-way traffic intersection, with different parameter configurations, changing the penetration rate, distance and time gap to the preceding vehicle. They found that the crossing time for platoon vehicles is substantially shorter than other vehicles since no delay between the light change and accelerating is exhibited.

Platooning can provide efficiencies both in terms of traffic flow and fuel (or energy in the case of electric vehicles). However, coordination of forming these platoons is important to enable these efficiencies, and this is something that

destination prediction could benefit. Knowledge of several vehicle's destinations in advance could enable forward planning in the formation of platoons, and have the potential to give recommendations to candidate vehicles ahead of time.

### 2.1.5 Summary

The literature surrounding intelligent transportation systems is vast and a large amount of research has been undertaken. Agent-based solutions and machine learning techniques are widespread across the domain, providing new functionality that improves safety, driver comfort and convenience. Vehicular communication systems hold the key to unlocking even greater potential for development within everyday vehicle use.

The element of collaboration between vehicles provides additional information, allowing for new functionality to be developed and existing systems to be improved. However, with communicating vehicles, new concerns regarding safety and security are raised. How such collaborative information is used and to what extent it can affect an individual vehicle or fleet of vehicles needs to be considered, including from a security perspective.

In much of the literature the communications channel is assumed to be perfect, with no delay or loss of data. In future work, extra focus should be put on how delayed information can affect the output or result of the application. Similarly, sensor readings are also assumed to be of high precision, especially in cooperative manoeuvres like platooning. Further exploration on how sensor errors and inaccuracies influence system performance is required, and new techniques may be required to compensate for these issues.

Due to the nature of the research, a large amount of simulation is conducted in previous work, but little real-world data is used and limited real-world experiments are carried out. Tests to establish the extent to which simulations are representative and provide reliable results should be conducted, to establish whether unexpected results would be observed when taken from simulation to the real-world.

Overall, there is great potential in these applications surrounding intelligent vehicles, all of which could benefit from advance knowledge of the intended destination of each vehicle. For example, using destination prediction could suggest an appropriate time for multiple vehicles to depart, to maximise the time that trajectories overlap, planning a platoon to be formed in advance as opposed to relying on ad-hoc encounters. This provides the motivation for this thesis, focusing on destination prediction approaches, and methods that assist this application. Specifically, the scope is focused solely on personal vehicles, in which the trip purpose of the journey is left open. We considered data collected

from taxis in Chapter 6, to allow for direct comparison to existing work. The benefits of destination prediction is not limited to these applications, and can also aid other intelligent applications such as intelligent energy management of the vehicle and parking recommendations at the destination (or even automatic pre-booking or reservations). Prior to looking at destination prediction in more depth, it is important to understand human mobility, considering concepts such as mobility motifs, as examining these patterns provides motivation for destination prediction.

## 2.2 Human Mobility

Human mobility is a broadening field of research [30, 66, 144, 150], which gives the general context for this thesis. Research into human mobility provides useful analysis that can influence areas such as urban and transportation planning. Studies have compared mobility patterns across cities, utilising smart-card data for the cities transportation networks [182]. Understanding human mobility patterns can have a profound impact on numerous applications including destination prediction. Recent research has found that the distribution of trip distance, the number of visited locations and the total travel distance for a given time interval (commonly referred to as the radius of gyration) serve as the key indicators for human mobility patterns [30, 66, 150]. Schneider *et al.* found that trips of a shorter duration generally exhibit high regularity and are commonly associated with tasks such as commuting to work or buying groceries. Schneider *et al.* claim that the average person only visits a small number of locations on a daily basis, and that 90% of the population visit less than 7 distinct locations per day, according to the surveys in Chicago and Paris analysed in their work [144].

González *et al.* also claim that the majority of people generally spend most of their time in very few locations, and visit up to 50 additional locations with their remaining time [66]. Human mobility motifs is a concept that has been investigated by several authors [34, 39, 83, 144, 183]. For example, in a case study of human mobility in Singapore, Jiang *et al.* discuss common activity chains that also appear in Boston, Chicago and Paris travel surveys [82, 144]. These include a home-to-work based tour, which is a direct commute to and from work, a home-to-work tour that includes a third destination, such as a trip to shops (where work is the primary destination), and a home-to-work based tour with a work-based sub-tour, such as a trip out for lunch [39, 83]. These activity patterns can be abstracted into daily motifs. By analysing mobile phone data, Schneider *et al.* identified 17 daily motifs which account for 90% of the recorded trips [144]. Similar frequencies of the most common motifs have been reported by Büscher *et al.* [34], in their recent work which

analysed data from the German Mobility Panel, showing that the ordering (and frequency) of the most common motifs has remained consistent over the past 20 years [34]. Other research has found that the detection of infrequent motifs can act as an indicator of mass activities, such as sporting events or music concerts [108].

Travel diaries, and the trends or behaviours learnt from analysing them have been widely researched. From a six-week travel survey in Germany [152] to a field trial in Stockholm [8], various datasets have been analysed. The durations of such studies vary, however in general such studies have a minimum duration of two weeks [8, 143, 152, 156]. Additionally, automatic estimation of activity patterns have been researched, where Maeda *et al.* propose an approach to detect changes in the number of individuals for a given activity type [117]. Schlich *et al.* claim that people's travel behaviour is not completely constant but equally is not irregular [143], and a German travel survey found that the majority of people who took part have a consistent frequency of week-on-week grocery trips [152], in which Saturday is the most common day for this [8]. From studies conducted in Belgium and Sweden, Sunday appears to stand out from all other days of the week with reference to the type of trips undertaken [136], with an emphasis on family and social trips [8]. Seasonality has an impact on the properties of journeys, with public transport use for leisure activities increasing in the summer, alongside an increasing distance travelled for a given activity [156]. Alternative transportation methods to personal vehicles are found to be used increasingly in areas that are well connected [95] and by students attending university [48]. Interestingly, the research by Daisy *et al.* showed that students surveyed at a Canadian university opt for personal vehicles when travelling for sports, hobbies and other entertainment activities [48]. These types of trips have been shown to have a higher frequency on a Wednesday and Saturday in Allstrom *et al.*'s Stockholm field trial [8]. As might be expected, there is an increased stability of people's travel behaviours on work days [143]. Existing research into destination prediction has shown that temporal aspects, such as day-of-week or time-of-day can improve predictive performance [99].

In the 2018 England National Travel Survey [57], it was found that 61% of trips were taken by a personal vehicle, in which the most common purposes for a trip were for leisure (30%), to go shopping (20%), or for commuting (15%). Figure 2.1 illustrates these statistics, highlighting a imbalance between the types of trip undertaken. However, this imbalance does not need addressing, as these distinctions are important for destination prediction. A person in England makes an average of 986 trips in a year, where 255 trips are for leisure, 188 trips are for shopping and 144 for commuting accounts for 144 of an individual's trips, regardless of the mode of travel. The majority of

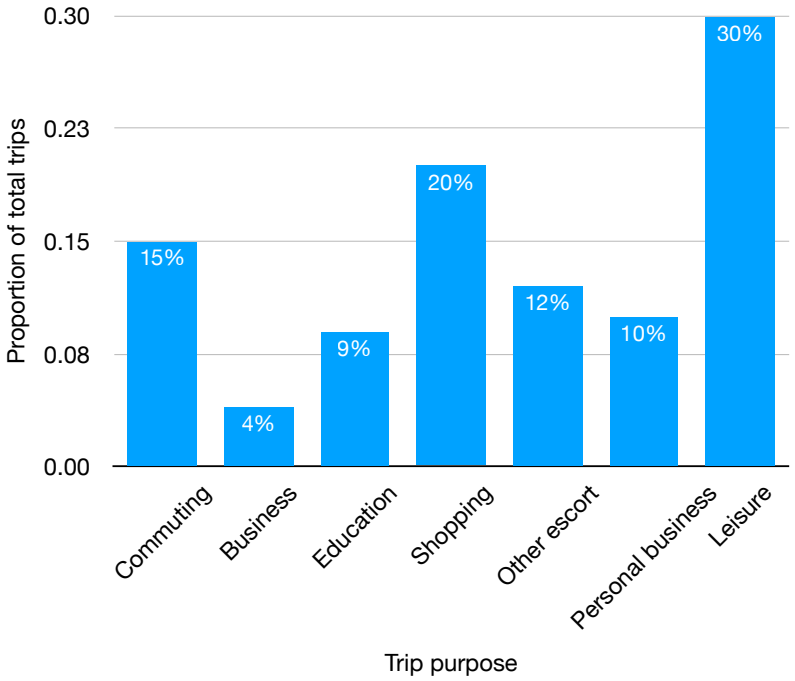



Figure 2.1: Distribution of trip purpose in a personal vehicle [57].

households (41%) have access to a single car or van, with 35% of households having multiple vehicles. This implies that in some cases there are multiple drivers for a single vehicle in a household.

## 2.3 Destination Prediction

Destination prediction has been the subject of much research, with recent work using historical GPS trajectories in order to predict an individual's next location or final destination. Markov models are widely used for destination prediction [16, 45, 73], with some methods considering multiple transport modes [9, 15, 44, 64] while others focus on vehicle trajectories [149, 175]. Other approaches include Bayesian inference [99, 110, 127, 187] to predict the intended destination of an individual, Gaussian mixture models [23, 172], decision tree learning [42, 44, 121], and support vector machines [170].

Many approaches to destination prediction focus on taxi data [23, 174], or other vehicle-based applications [99], but others are more general, where individual's walk or take public transport [9]. In this thesis, we focus of destination prediction of vehicles, in which we consider taxi data.

Research into predicting an individual's next location is similar to destination prediction, but focuses on predicting the next intermediate location, in contrast to the final destination. Ziebart *et al.* provide a good example of next location prediction, where they predict individual turns along a trajectory

[187].

Several destination prediction methods use a grid-based approach [99, 175, 187] or have a graph representation of the road network [88, 127, 149]. However, this often requires external data to enable pre-processing, such as map matching and prediction using information from external sources, such as GIS data [88]. Clustering is frequently used as an initial step in destination prediction, converting stay points (points of low mobility) into places [15, 23, 158, 159]. Throughout this thesis, we employ clustering techniques to gather stay points, and to transform stay points into place. We also consider a grid-based approach for destination prediction in Chapter 5.

There has been some research that attempts to address the data sparsity problem. For example, Xue *et al.* propose a method called Sub-Trajectory Synthesis (SubSyn), that decomposes trajectories into smaller segments and connects these to adjoining segments, creating synthetic trajectories [175]. This greatly increases the number of possible trajectories that are modelled from an input dataset, since it is rare to have an exhaustive set of input trajectories available. The available data can also be increased by considering multiple individuals, and how individuals complement each other, with similar trajectories increasing the value of the training data [65].

Krumm *et al.* and Xue *et al.* both use a 1km grid-based approach [99, 175], and Ziebart *et al.* evaluate their PROCAB algorithm on multiple grid sizes [187]. A coarse grid improves the matching performance, but is detrimental to destination prediction error, since the grid squares span a larger area. The opposite is seen with a fine grid, therefore an appropriate grid size should be selected to achieve an acceptable trade-off between trajectory matching performance and destination prediction error. A poor choice of grid size may cause separate destinations to be grouped together. The grid representation is extended in work by Chen *et al.*, in which grid cells are merged where adjacent cells have similar routes [41, 42]. The WhereNext algorithm uses a similar approach, where Monreale *et al.* propose T-Patterns, which are sequences of regions [121].

Alternatives to grid-based approaches include map matching or generating local graph representations of the road network [88, 127, 149]. Simmonds *et al.* use a mapping database to provide a road graph, in which link-goal pairs can be formed [149]. Their model can predict the next link, and subsequently infer the final destination [149]. Karimi *et al.* construct a tree-based structure using additional road-network data, alongside amenity information [88]. Similarly, Patterson *et al.* also propose a graph-based representation, which is constructed from a street map provided by the US Census Bureau [127].

Clustering techniques are used in many approaches as a means of extracting locations, with k-means [15, 45], hierarchical [23], and density-based [159]

clustering being used. Choi *et al.* adopt the k-means approach to cluster trajectory segments [45], similar to Ashbrook and Starner who map significant locations into clusters [15]. Ashbrook and Starner use a graph comparing the number of clusters to the number of locations, locating the knee point in order to select a suitable number of clusters [14, 15]. This has proven to be a popular method, and similarities are seen in several related approaches [9, 62, 99, 110, 127, 149]. Cho *et al.* extract intermediate instances by using a more computationally expensive Gaussian-means approach [44]. Conversely, Gambs *et al.*, use a density-based approach to generate the corresponding locations from their input data [64].

To predict destinations, we first have to separate the data into distinct trips, in which the first instance of a trip is the start location, and the final instance is the destination. Time thresholding is a widely used technique to achieve this [9, 15, 99], where the threshold is a minimum duration between two consecutive recorded instances (and instances are not recorded if an individual stays in the same place). Chen *et al.* [42] use a threshold of 2 minutes, Krumm *et al.* [99] and Alvarez *et al.* [9] use a threshold of 5 minutes, while Ashbrook and Starner opt for a 10 minute threshold [15]. The threshold value used varies, implying that it is non-trivial to find a suitable value.

Approaches to destination prediction also have varying input data, with some only using spatial information from within trajectories [23], while others use multiple external sources [88, 99]. For example, Krumm *et al.* use ground cover data [99], vehicle speed is used by Fukano *et al.* [62], Karimi *et al.* use data on local amenities [88], and others use temporal data [99, 149, 170]. Temporal data, such as the day-of-week or the hour-of-day, can act as an indicator of the next location, and have been shown to improve predictive performance [149]. Our aim in this thesis is to reduce the dependency on external data and, since temporal data is implicitly available within a trajectory record, our methods will focus on the use of data that is naturally contained within trajectory data. Additionally, in Chapter 5, we consider on-board vehicle data in our experiments.

Besse *et al.* propose a destination prediction method (which we refer to as BDP), which uses distribution-based models to match similar trajectories [23]. The training trajectories are grouped using hierarchical agglomerative clustering, with the distance between trajectories computed by the Symmetrized Segment-Path-Distance (SSPD) [22]. The Segment-Path distance is the mean of each of the points in one trajectory to the closest segment in the other trajectory. SSPD is defined as the mean of the sum of both Segment-Path distances between two trajectories. A detailed explanation of SSPD is given in Section 6.2.1. Using the clustered trajectories, Besse *et al.* train 2D Gaussian Mixture Models over each cluster, using the latitude and longitude to fit a

distribution to a sample of training coordinates. Once these models are trained, a likelihood can be assigned for each cluster in an unfolding trajectory, and its destination is predicted using the centroid of the most likely cluster. Besse *et al.* also propose using a weighted score, which uses auxiliary variables, such as the hour-of-day, with each variable associated with a weighting function to modify the GMM likelihood. Our proposed method in Chapter 6 avoids the need for defining such weighting functions and is easily extensible in terms of adding additional variables. Our method also results in smaller clusters that naturally take the auxiliary variables into account, which can be beneficial for interpreting predictions. Since our focus is on identifying suitable clusters from which to make predictions, we evaluate our approach against BDP with the unweighted simple score. BDP has several benefits over other methods since it does not rely on external data [88, 99], it does not require a mapping of the road network, which is computationally expensive to process [127, 149], and it does not discretise the space into a grid representation [99, 175, 187]. Given the above benefits, we use BDP as part of our approach in Chapter 6 While Besse *et al.* propose the use of auxiliary variables, these variables do not segregate the trajectories into more specific clusters, unlike our proposed approach in Chapter 6. Gaps in the literature also exist where trajectories could be clustered into more specific groupings, using criteria such as spatio-temporal attributes. While not the main aim of Chapter 6, our proposed method is extensible by design and allows multiple attributes to be used to narrow down specific trajectory groupings.

## 2.4 Data Collection

In this thesis, we focus on collecting geospatial data from vehicles, in conjunction with telemetry data from on-board sensors. This data allows us to characterise a trajectory both in terms of the location of the vehicle, the human inputs to the vehicle (such as opening a door, or pushing the accelerator), and the vehicle's resulting state (such as speed, fuel level etc.).

### 2.4.1 Trajectories

A trajectory is a sequence of data points. Typically, a geospatial trajectory consists of instances that contain a latitude, longitude and timestamp. A trajectory can have many forms, for example a trajectory can represent animal movements [180], or the arm movements of an individual [122]. Depending on its form, a trajectory may have different attributes, such as the sampling rate of each instance.

In this thesis, we focus on vehicular trajectories. Vehicular trajectories

always have a defined start and end point, i.e., when the vehicle is started and stopped respectively. Additionally, these vehicular trajectories may contain additional data from sensors on-board the vehicle, such as the steering wheel angle, the vehicle speed, and the currently selected gear. Formally, we define a trajectory of raw data, $t_i = [x_1, ..., x_{|t_i|}]$, to be a vector of instances. We define an instance $x_j$ at time $j$ to be a tuple $x_j = \langle lat, long, V \rangle$ containing a latitude, $lat$, longitude, $long$, and a vector of vehicle signal values, $V$.

### 2.4.2 Geospatial Data

There are a range of global navigation satellite systems (GNSS), through which a receiver can determine its latitude and longitude. The four most prevalent systems are BeiDou, Galileo, GLONASS and GPS, each of which have a different set of benefits and drawbacks. The accuracy of these systems is generally below 5 metres. Low-cost GNSS receivers are widely available and can be found in a variety of devices, such as smart phones, fitness trackers and digital cameras. Most new vehicles contain on-board navigation, and therefore contain a GNSS receiver.

Given the high-availability of devices with these capabilities, geospatial position has become an important data source, enabling a series of location-aware applications such as personalised advertising, asset tracking and destination prediction. Research has been carried out to use a combination of these systems in order to improve the precision [106]. In this thesis, we utilised a GNSS receiver in the vehicles that were used to collect both the LED and POLD.

### 2.4.3 On-board Vehicle Sensors

Modern cars are fitted with a vast array of sensors that can provide a high-level of detail about the vehicle. Typical sensors include the fuel level, vehicle speed and steering wheel angle. Different sensors may also sample data at different intervals, for example the steering wheel angle is typically updated more frequently than the fuel level.

The Controller Area Network (CAN) protocol was commonly used in the automotive industry, and is an illustration of the data buses that are typical. It is a carrier-sense, multiple access protocol with collision detection [138]. The CAN bus is a broadcast-message based protocol, where if multiple devices transmit data at the same time, the highest priority device is given access to the bus and the other devices back off. Messages are received by all devices on the bus. The data rate of the CAN bus can reach 1 Mbit/s, where different sensors will transmit messages at different rates. The CAN bus provides communication between vehicle subsystems, allowing functionality such as start/stop, and automatic release electronic parking brakes. Connecting a

logging device to the CAN bus allows us to capture all these signals and record them for later use, which is the process adopted in this thesis to collect the LED and POLD.

There are two main alternatives to the CAN bus for vehicle networks, namely FlexRay [118] and Automotive Ethernet [72]. FlexRay provides benefits over CAN, including support for both synchronous and asynchronous communication, a higher data rate, and clock synchronization [118]. Automotive Ethernet can provide benefits over both CAN and FlexRay with a higher communication bandwidth and increased compatibility with IoT and other smart devices [72].

## 2.5 Stay Point Extraction

The process of stay point 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 stay point is typically defined as a group of instances in a trajectory that exhibit little or no movement, implying a period of low mobility in which an individual remains in the same location [18, 126, 158]. Given this definition, Palma *et al.* [126], Bamis *et al.* [18] and Thomason *et al.* [158] each assume that all stay points are meaningful, which is not necessarily the case for vehicle data, where areas of low movement exist that are not relevant to an individual, such as waiting in traffic. In Chapter 1, we define a PoI to be a stay point where the vehicle has stopped for an intended purpose. This results in existing general purpose stay point extraction algorithms generating multiple false PoIs when applied to vehicle data.

Stay point 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 set of PoIs to provide acceptable performance [38, 99, 149, 181]. Although hard to quantify, it is necessary that the set of extracted PoIs are robust. Selecting too many PoIs needlessly increases the unnecessary data, which can make prediction harder due to increased noise. Conversely, missing PoIs may also have an adverse effect on prediction; if some PoIs that contain the most information are missed, the accuracy may be reduced. Stay point extraction can also be used to identify semantically relevant places for individuals 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 [90]. Similarly, inferring the activity performed at a given PoI is investigated by Furletti *et al.* [63], by linking PoIs to amenities using semantic data such as a individuals' 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 individual's home or work can also be considered when developing location-aware applications [38, 114].

### 2.5.1 Clustering Approaches

Extracting stay points 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 stay point extraction, typically using density-based approaches [56]. Additionally, some clustering algorithms have been proposed specifically for stay point extraction, namely CB-SMoT [126], STA [18] and GVE [158], which represent the current state-of-the-art.

Density-Based Spatial Clustering of Applications with Noise (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 stay point extraction since this is typically unknown [56]. Another advantage of DBSCAN over other general purpose clustering algorithms, such as k-means [75], is that it can cope with clusters of different shapes. DBSCAN uses two parameters, $\epsilon$ and $minpts$, that respectively determine the absolute distance used to calculate the neighbourhood 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 stay points that are meaningful to the individual [184, 185]. This requires individuals to confirm whether the discovered stay points are correct and to rate their importance on a 5-point scale. In the standard formulation of DJ-Cluster, all stay points that are rated 4 or above are considered 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 neighbourhoods of both clusters, rather than performing an outward neighbourhood search on each instance in the resulting neighbourhood. D-Star extends DJ-Cluster, using a sliding window to create the neighbourhood, allowing the algorithm to work online [123]. 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 stay point. ST_DBSCAN also extends DBSCAN, considering non-spatial, spatial and temporal aspects to generate clusters [27].

The Clustering-Based Stops and Moves of Trajectories (CB-SMoT) al-

gorithm [126] calculates a distance threshold, $\epsilon$, 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 $\epsilon$ 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. Additionally, CB-SMoT allows areas of known stay points to be input, so that identified stops can be categorised into both known and unknown stay points. 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 thesis for comparison.

Techniques such as Spatio-Temporal Activities (STA) [18] and the Gradient-based Visit Extractor (GVE) [158] 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.

Several techniques use static time and distance thresholds, including the work of Kang *et al.* [86] and Fu *et al.* [61]. However, these techniques have been shown to exhibit poor performance when there is even a limited amount of noise in the data [18]. STA and GVE overcome this by using averaging filters to compare subsequent instances [167]. Chen *et al.* employ static thresholds on taxi trajectory data, where GPS readings are sampled every 15 seconds [43]. Activity 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 activities. Bhattacharya *et al.* use a bucketing technique with time and distance to infer speed (and acceleration) [24]. They consider two different types of location, a point-based stay point such as an office, where the individuals' movement is negligible, and an extended stay point, such as a market, where the individual

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 [25]. However, these speed and direction-based algorithms are not suited to extracting stay points from vehicle trajectories because they require a list of nearby places (e.g. amenities and addresses), which may not be available.

### 2.5.2 Summary

Stay points can vary in duration and shape, and there is no single approach or parameter configuration that is appropriate for all application domains. Additionally, different domains do not consider all stay points to be PoIs (e.g., when a vehicle is waiting in traffic). 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 individual's PoIs [18, 126, 158].

In this thesis, we consider both stay points and PoIs, where a PoI is a stay point that is considered meaningful to the journey. To extract stay points from trajectories, we consider CB-SMoT, GVE and STA as representative algorithms, which are used in Chapter 4. In Chapter 4, we consider the differences between stay points and PoIs, and propose an approach to filter out false PoIs (e.g., waiting in traffic) from vehicular trajectories.

## 2.6 Activity Classification

Activity classification is the process of identifying the current activity being performed by a certain entity. In the case of vehicle activities that we refer to here, and in the remainder of this thesis, these are not activities in the context of human mobility e.g. leisure. Instead, the activity of the vehicle relates to common situations or tasks encountered, for example, waiting in traffic. The type of activity classification is dependant on the application, such that vehicle activity classification could relate to predicting the speed or movement of the vehicle. However, in this thesis we consider the activity of a vehicle to be more general, such as stopping in traffic, picking up a passenger or waiting at a barrier.

In other application domains, activity classification has been used for many tasks ranging from detecting daily household activities [97, 109, 111, 135, 177] to specialised models predicting sports moves [17, 33]. Other types of activities

include transportation mode and phone usage [101]. Many approaches have used computer vision as the main medium for activity recognition [119, 145], including classifying the actions of passengers in a vehicle [160, 161].

Hidden Markov Models (HMM) [20, 134] are commonly used, along with alternative techniques such as Support Vector Machines (SVM), Bayesian classifiers and neural networks [17, 67, 91, 104, 109, 186]. Techniques from existing activity classification approaches, such as the use of acceleration data [102] and sensor fusion [77], can be applied to vehicle activity classification.

A common challenge faced by activity classification is defining the exact boundaries of a given activity [33]. The issue of data segmentation has been widely researched, with the use of temporal windows as a potential solution [19, 155, 165]. Research suggests that some opt for static windows, however static windows present issues if different individuals' take a varied amount of time to perform the same activity. This may cause an activity to only be partially captured, or the segment to be overly large, when in fact the activity is shorter than anticipated. Dynamic windows have also been experimented with [165], where extra data from sensors can alter the window length.

## 2.7 Summary

This chapter has provided a background introduction to intelligent transportation systems, highlighting the research into route planning, traffic assessment, traffic flow efficiency and platooning. We have identified that destination prediction could be used to enhance these intelligent vehicle applications, thus motivating the work in this thesis.

We have introduced the concepts of trajectory data, and the technologies traditionally used to collect the geospatial data. These form the basis upon which we collect the LED and POLD datasets in Chapter 3. However, the key novelty of our dataset it to enrich the feature set for the destination prediction tasks by augmenting the geospatial data with on-board vehicle data. Stay point extraction, and the difference between a stay point and PoI has been discussed, along with common approaches used for this task. In Chapter 4, we propose a PoI extraction approach for vehicle trajectories, which shows an improvement over the state-of-the-art clustering algorithms (CB-SMoT, GVE and STA, as discussed in Section 2.5.1), that reduces the impact of false PoIs which are generated.

In Chapter 5 we study the effect of enhancing the features available for destination prediction by augmenting stay point data with activity classification data (which was discussed in Section 2.6). We discussed various approaches to destination prediction, considering the machine learning techniques used in previous research. In this thesis, we evaluate a range of approaches, including

Hidden Markov Models, Gaussian Mixture Models (GMMs), and decision tree classifiers, which we use in Chapters 5 and 6. In Chapter 6 we propose a multi-layered clustering approach that improves the destination prediction error over the initial phase of a trajectory.

# Chapter 3

# Datasets

In this chapter, we detail the datasets used in our research, including two that were collected during the course of the research, namely the Location Extraction Dataset (LED) and Warwick Pattern of life Dataset (POLD). The LED is used to evaluate our method in Chapter 4, and the POLD is used in Chapter 5, and supplements the analysis in Chapters 4 and 6. We also utilise two publicly available datasets, namely the Caltrain [130] and Porto [1] datasets, for Chapter 6 where we consider trajectory classification and destination prediction. The Caltrain and Porto datasets are used for Chapter 6, to evaluate our proposed method against the method proposed by Besse *et al.*, as these dataset are used in their initial evaluation.

The motivation behind the creation of the LED was that, to our knowledge, no other datasets exist that include both a full set of ground truth labels and a set of sensor data from on-board a vehicle for studying point of interest extraction. It was paramount to have the vehicle sensor data available to evaluate our proposed vehicle activity classification method. The LED consists of a number of scripted journeys and scenarios that ensure multiple activities are captured. Similarly, there are few available labelled pattern of life datasets, mainly due to privacy concerns of the individuals supplying the data. Examples exist such as the Nokia Mobile Data Challenge dataset [94], however this suffers from a lack of on-board vehicle signals in addition to the start and end of trajectories being truncated for privacy reasons. We collected the POLD to ensure access to the participants for further investigation and to obtain a full labelling, for use in our proposed destination prediction approaches. Additionally, collecting vehicle sensor data for general pattern of life driving allows us to evaluate the proposed vehicle activity classification on unscripted data, where the participants would drive their normal journeys, as if it were their personal vehicle.

## 3.1 Location Extraction Dataset (LED)

In order to evaluate the proposed point of interest extraction method, we defined a data collection methodology and collected a scripted scenario dataset,

namely, the Location Extraction Dataset (LED). The LED is provided for others to utilise in further research on vehicle PoI extraction. This section outlines the data collection procedure for the LED, along with the definitions of the activities and their transitions. Additionally, the routes in the LED are detailed and we present a summary of the dataset.

### 3.1.1 Data Collection

We used a set of predefined routes from which the LED was collected. The data on these routes were collected in 2 different vehicles, with slight variations in routes between the vehicles. The first vehicle, referred to as *the SUV*, was a 2-door 4-seater convertible SUV, and the second, referred to as *the estate*, was a 5-door 5-seater estate car. Data collection was performed with the support of our industry partner, who dictated vehicle choice and availability. Each route was repeated 8 times in the first vehicle and 5 times in the second vehicle. The difference in number of repeats was due to data loss after the data collection period had ended. To collect a representative sample, we scheduled journey times that varied between peak daytime (07:00-10:00 and 16:00-19:00), nighttime (23:00-05:00) and off-peak daytime. Additionally, routes include sections of major and minor roads. Open air and multi-storey car parks were used at shopping centres, 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. All journeys 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.

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 1Hz, while CAN signals were broadcast to the logger, and were generally recorded at a higher rate. These were combined and subsampled to the rate of the GPS signal. 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.

### 3.1.2 Activity Labelling & Transitions

We propose a method for applying activity labels and transitions to the collected data. The approach of defining of activities is fundamental to exploring point of interest extraction within vehicle trajectories, as we can distinguish the

current state of the vehicle into clear, distinct groups. Similarly, well-defined transitions between activities are required to ensure consistency. While we propose a set of activities and transitions for this dataset, the effectiveness of the specific definitions are dependant on the context and the definitions can be tailored depending on the application.

We define the following activities for the LED, in which we decided these from intuition and through discussion with our industry partner:

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

To ensure reproducibility, and application of this methodology to other datasets, an accurate and consistent ground-truth labelling must be applied to the data. However, defining where the boundaries exist for each activity is a non-trivial task, and requires identification of the exact instance in which transitions occur [97, 139]. Quantitative bounds were created to formalise the start and end instances for each activity (e.g., once the vehicle goes below 1km/h), alongside qualitative criteria (e.g., a drop-off activity must include a passenger exiting the vehicle). We have defined a set of transitions to be applied to the LED, which can be used by other researchers in future investigations into PoI extraction. This set of transitions ensures that activities cannot overlap.

Examples of the transitions from the Driving activity to the other activities are shown in Table 3.1. The full set of transitions are available in Appendix A.

Table 3.1: 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 1km/h. |
| Drive-through | When the first booth (or order point) is reached and the vehicle speed first falls below 1km/h. |
| Drop-off | When the vehicle speed first falls below 1km/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 activity to be true. |
| Manoeuvre | When the vehicle speed first falls below 1km/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 1km/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 1km/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 activity to be true. |
| Traffic | When the vehicle speed first falls below 5km/h as a consequence of encountering congestion or road infrastructure that causes either 5km/h not to be reached within 10 seconds or the vehicle speed to fall below 1km/h within 10 seconds. |

### 3.1.3 Route Breakdown

A detailed description of the routes is as follows, noting the variants (and their respective reasons) for *the estate* (if unspecified, there is no variant for the estate). Maps for each route (and their variants) are illustrated in Appendix B.

**Route 1:** This route commences at the university campus, from an open-air car park, with a single occupant in the front seat. The journey leaves the university campus, heads towards and subsequently joins a dual-carriageway, and exits at first subsequent slip-road. The route continues into the high street of a nearby town, where the front passenger is dropped off. The vehicle continues to a multi-storey car park, in which it parks, waits for a few minutes and then exits. The passenger is subsequently picked up from the town in a nearby residential street, before heading back to the university campus.

*The estate variant heads towards an open-air car park (instead of the multi-storey car park, due to the size of the vehicle), after the front passenger has been dropped off, where the car is parked and waits for a few minutes (the same actions as performed in the multi-storey car park). After the passenger is picked up, the route varies by heading, and terminating at a local superstore, instead of returning to the university campus to reduce the long periods of driving in which there is little of no extra information, as this reduces the demand on participants.*

**Route 2:** The second route commences at the university campus, in a multi-storey car-park. With a single front passenger on-board, the vehicle leaves the university campus for a nearby local train station, in which the front passenger is dropped off, before a turn-in-the-road is performed. The vehicle leaves the area of the station for a nearby residential road where it parked, and waits for a few minutes. After this, the vehicle performs a turn-in-the-road, heads back to the station to pick-up the passenger before returning to the university campus.

**Route 3:** This route is one of the shortest, commencing in a multi-storey car-park at the university campus. A single front passenger is present and the route leave the campus and heads towards a local residential area, where the passenger is dropped off. The route continues, looping around a residential area to return on the other side of the residential street, where the passenger is picked up before returning to the multi-storey car park.

*The estate variant skips the loop around the residential area and turns at the first roundabout, heading straight back to the pick-up point. This is due to a mistake in the data collection that was found after the collection period finished.*

**Route 4:** The fourth route is the only route that starts away from the

university campus, in a nearby town. With a single front passenger aboard, the route commences from on-street residential parking, and heads to a drive-through service. After using the service, the route continues to a nearby train station, to use the drop-off point, where the passenger is dropped off. Following this, the route returns to the original residential on-street parking area.

**Route 5:** This route is the first that includes multiple passengers, one of whom is in the front seat. The route begins on the university campus and takes the dual carriageway to a nearby town, similar to Route 1. Upon entering the town, the route follows the main road in to a residential area, where each passenger is dropped off in a different location within the residential area, before parking nearby in an on-street location. After waiting for a couple of minutes, the route resumes with a turn-in-the-road and proceeds to pick-up the two passengers from where each was initially dropped off. Following the pick-ups, the journey concludes by heading into the town centre and parking in a multi-storey car park.

*The estate variant starts in a different open-air car park at the university campus (as this car park has a barrier, and replaces the barrier activity we later miss from avoiding the multi-storey), and after picking up both passengers returns to the open-air car park at the university campus instead of the multi-storey in the town centre (due to the size of the vehicle).*

**Route 6:** The sixth route begins in on-street parking at the university campus. With a single passenger, the journey starts by driving to a restaurant on the outskirts of the campus, picking up a second passenger from an open-air car park and performing a turn-in-the-road. The journey continues on a dual carriageway into a commercial area, where a drive-through service is used. The journey finishes by heading back to the parking at the university where it commenced.

**Route 7:** This route starts at the university campus and proceeds into a nearby city, in which two passengers are present. The first passenger is dropped off at the city's train station before proceeding to a hotel where the second passenger is dropped off. The route continues to a small open-air car park, where a turn-in-the-road is performed. The vehicle then returns to the hotel, and subsequently the station, where each passenger is picked up. Finally, the vehicle heads to the multi-storey car park of a large store in the city centre, where it is parked, and the journey terminates.

*The estate variant sees the second passenger dropped off at a shopping centre on the outskirts of the city, before heading into the city to a church car park, in which the turn-in-the-road is performed. These were both varied due to vehicle size restrictions. Both passengers are then picked up from their respective drop-off locations. The end of the route is identical, heading to the multi-storey car park of a large store in the city centre.*

Table 3.2: Summary of route durations in the LED.

| Route | Max. Duration (s) | Min. Duration (s) | Avg. Duration (s) |
|---|---|---|---|
| 1 | 2918 | 1498 | 2281 |
| 2 | 1139 | 647 | 839 |
| 3 | 772 | 409 | 573 |
| 4 | 1720 | 706 | 1077 |
| 5 | 2475 | 1521 | 1909 |
| 6 | 2211 | 1031 | 1451 |
| 7 | 2716 | 1621 | 1967 |
| 8 | 1744 | 1046 | 1348 |
| 9 | 506 | 291 | 378 |

**Route 8:** The route starts at on-street parking within the university campus, with a single passenger. We exit the university campus and head to a nearby shopping centre, where the passenger is dropped off. The journey continues, to a busy roundabout where we loop back and return to the shopping centre to pick the passenger back up. Then, the route heads back to the university, visiting a university accommodation block and dropping the passenger off once again, before heading to an open-air car park and ending the journey.

**Route 9:** This is the shortest route, starting at an open-air car park at the university campus. The route starts with no passengers and proceeds to do a loop of the main campus, picking up a passenger outside the campus convenience store and further dropping them off outside the security building. The route concludes by heading back to the original open-air car park.

### 3.1.4 Dataset Statistics

The LED contains 117 journeys and 153,699 instances, over a duration of 42 hours 41 minutes and 39 seconds.

The shortest route lasted around 6 minutes on average, whereas the longest route took around 38 minutes. The distribution of the journey durations in the LED is summarised in Table 3.2.

Not all routes contained every activity, and the frequency of activities varied as can be seen in Figure 3.1, which shows the distribution of activities within the dataset. There is a activity bias, which we chose not to address, as in reality this will be present i.e. significantly more traffic instances that pick up instances. The duration of a single activity is dependant on the type of activity, as shown in Figure 3.2, with the majority of activities being around 20 seconds on average. Since there are differences in the activity durations, the distribution of instances per activity type differs slightly from the frequency distribution of activity types, as can be seen in Figure 3.3 (when compared to Figure 3.1).

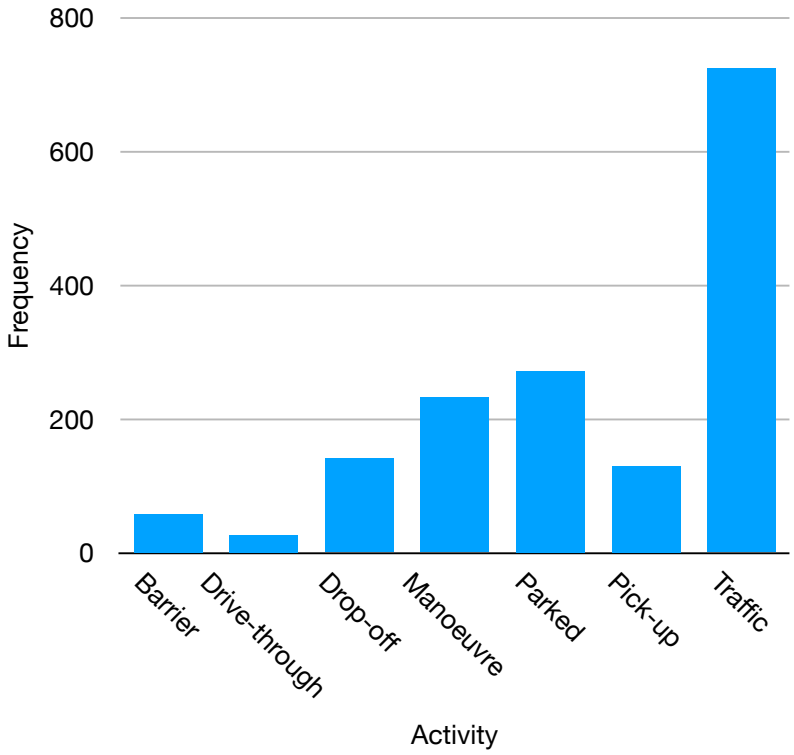



Figure 3.1: Distribution of the activities in the dataset, with the driving activity (1248 occurrences) omitted to aid readability.

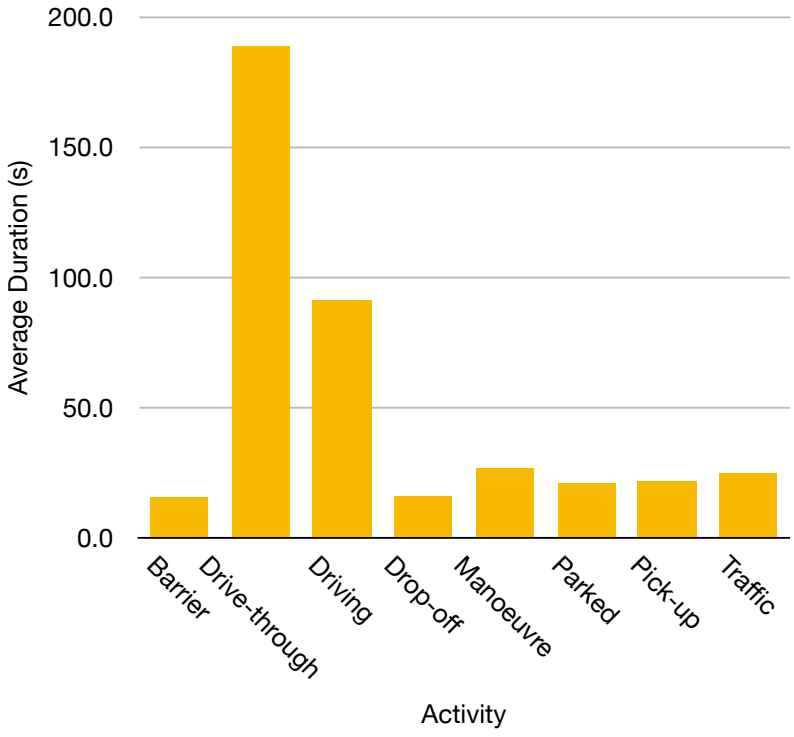



Figure 3.2: Average duration of activities.

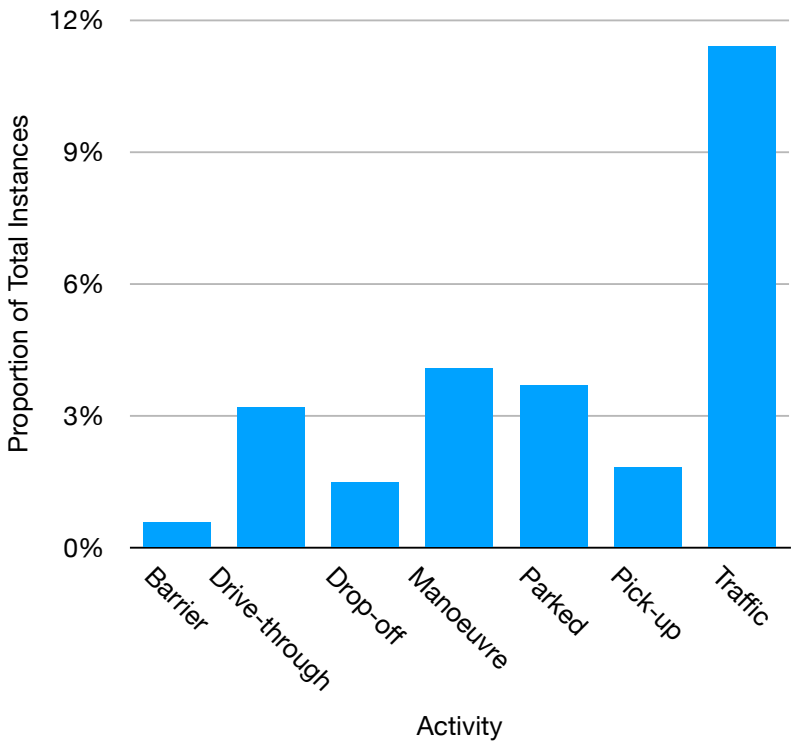
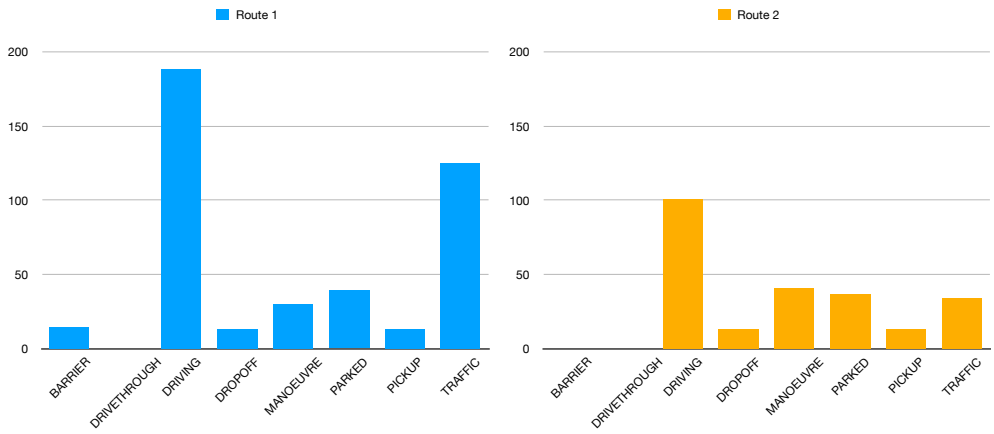


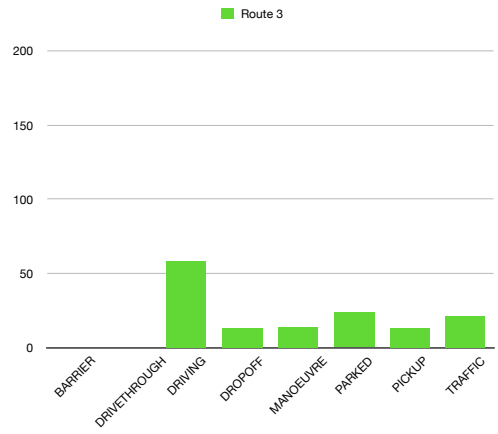

Figure 3.3: Proportion of the total instances per activity, with the driving activity (74%) omitted to aid readability.



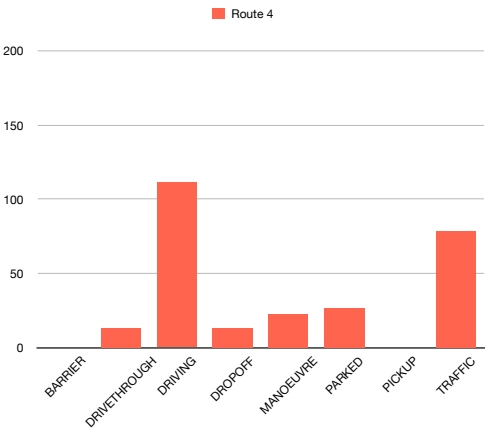

(a) Route 1



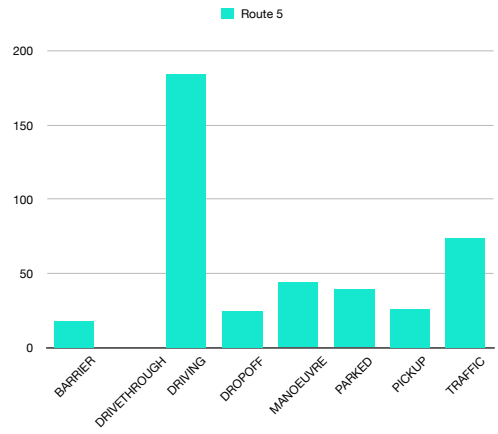

(b) Route 2



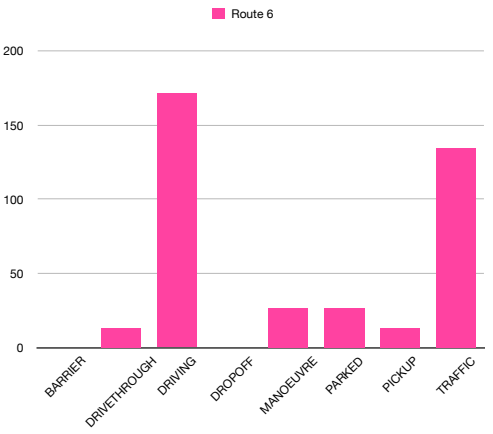

(c) Route 3



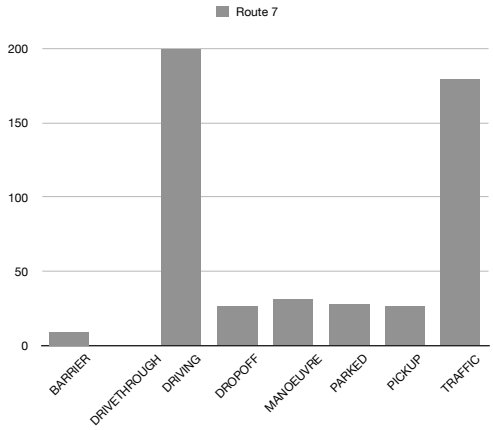

(d) Route 4



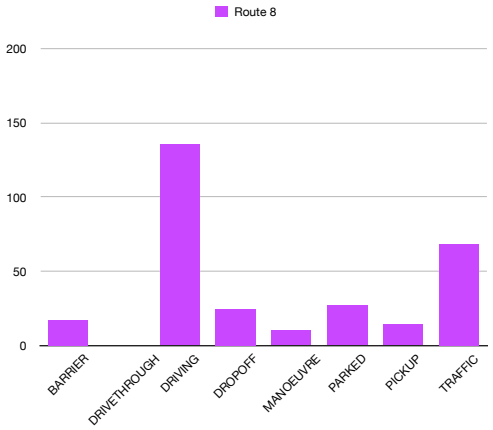

(e) Route 5



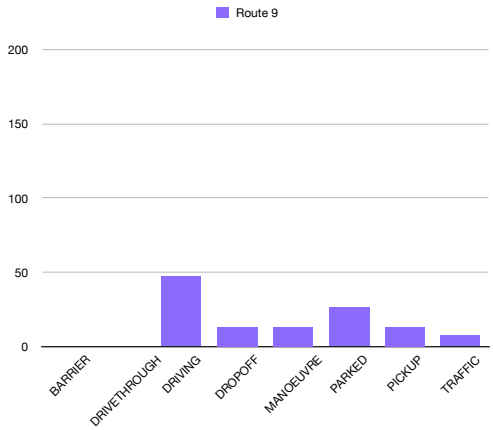

(f) Route 6




(g) Route 7




(h) Route 8




(i) Route 9

Figure 3.4: Distribution of activities per route in the LED.

Figures 3.4a–3.4i further decompose the distribution of activities by each individual route. We can see that the longer routes, namely routes 1, 5, 6 and 7 have an increased proportion of driving, which is expected due to their length. If we example the relationship between driving and traffic activities, we can see that routes 1, 4, 6 and 7 have the smallest difference, owing to these routes going through busier areas and therefore having an increased likelihood of encountering traffic (as well as enforced road infrastructure, e.g., time-delayed traffic lights). There are no routes that contain both barrier and drive-through activities, and all routes contain manoeuvres. Finally, we can see that all routes either have a pick-up or drop-off activity, but not necessarily both.

The dataset, and its corresponding activities are plotted in Figure 3.5. All activities, apart from driving are marked by coloured points on the plot. GPS jitter is present where the journey either started or ended in multi-storey car parks. From this plot, it is clear to see the three common areas in which the activities occur. Roads linking these areas do not contain any activities other than encountering traffic (and general driving). We can see hotspots where the majority of traffic activities occurred, mostly at junctions or roundabouts.

## 3.2 Warwick Pattern of life Dataset (POLD)

The Warwick Pattern of life dataset (POLD) is an unscripted driving dataset consisting of multiple participants undertaking their general day-to-day journeys. The POLD contains data from 5 participants, over 2 distinct vehicles, and is complete with a labelling for both the source and destination of each journey, along with the purpose of the journey. The vehicles used in this dataset are *the SUV* and *the estate*, are described in Section 3.1 where we described the LED. Each participant had sole use of the vehicle for between one and three weeks, before having a period of time without the vehicle. Given the finite and uncertain time in which the vehicle would be loaned to the project, pragmatic allocation based on participant's availability and constraints was required. Vehicle signals and GPS data were collected for every journey in the data collection period, and associated with their anonymised participant ID. We removed journeys that contained corrupted or invalid data as a pre-processing step, and combined and sub-sampled the GPS data with the vehicle signals to a rate of 1Hz, using the same process as for the LED.

### 3.2.1 Dataset Statistics

The POLD contains 826 journeys over a distance of 20230 kilometres travelled. The total travel duration of all journeys is a little over 407 hours. All journeys were in the United Kingdom, and are contained within a 386.2km by 424.1km

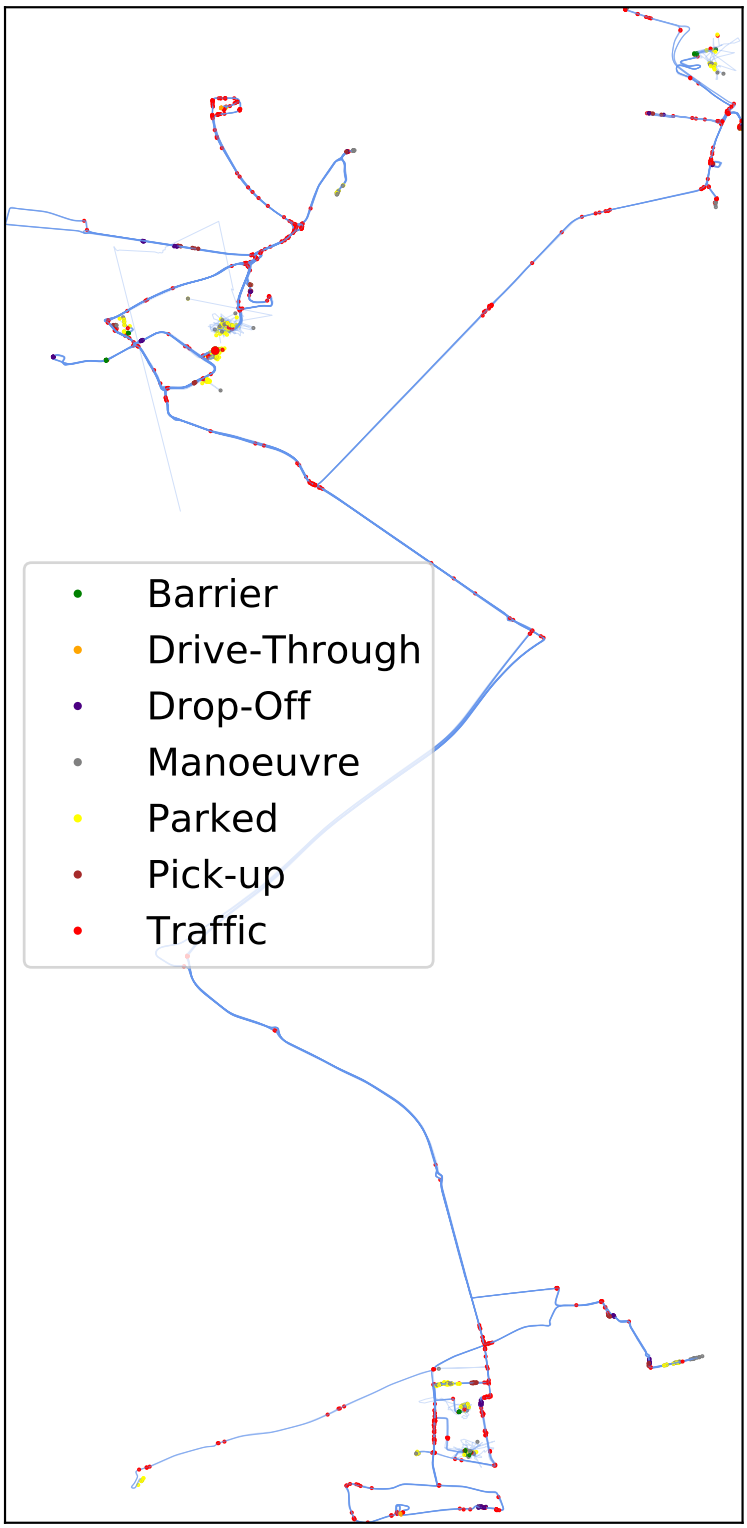



Figure 3.5: Plot of trajectories and activities within the LED.

Table 3.3: Individual Participant Statistics for POLD.

| Dataset | Number of Journeys | Total Journey Duration | Total Distance Travelled (km) | Average Journey Distance (km) | Maximum Journey Distance (km) |
|---|---|---|---|---|---|
| uw1 | 102 | 52 hrs 3 mins 0 secs | 2601.2 | 25.5 | 172.2 |
| uw2 | 113 | 55 hrs 48 mins 33 secs | 2907.4 | 25.7 | 227.6 |
| uw4 | 80 | 30 hrs 50 mins 5 secs | 1242.6 | 15.5 | 193.8 |
| uw6 | 305 | 147 hrs 50 mins 47 secs | 6702.8 | 22.0 | 308.1 |
| uw9 | 226 | 120 hrs 38 mins 45 secs | 6776.5 | 30.0 | 325.3 |

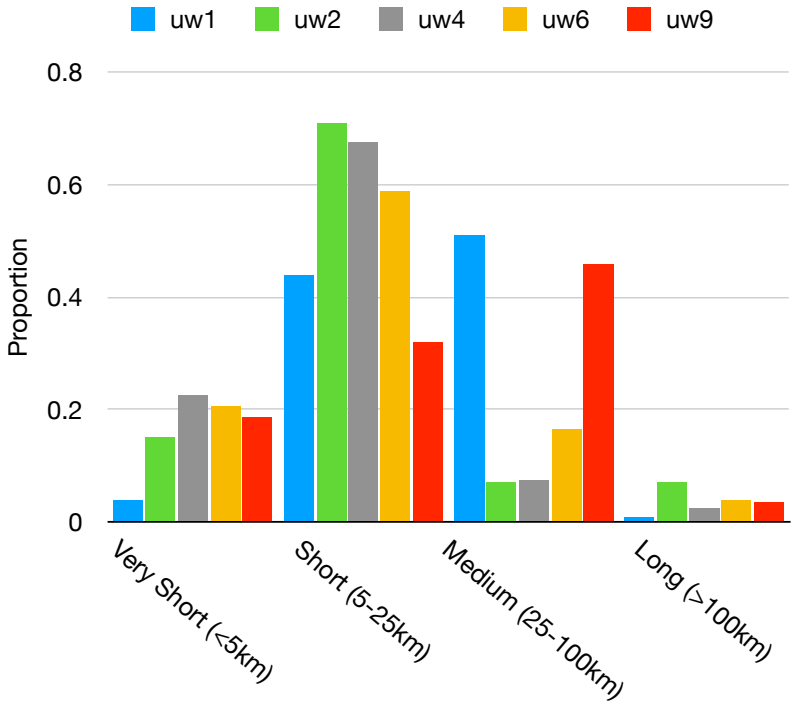



Figure 3.6: Distribution of number of journeys by total distance.

grid.

All 5 participants in the POLD were male, university educated with a Bachelor's or higher, and either studied or worked at a university at the time of collection[1].

A breakdown of participant data is shown in Table 3.3. The highest amount of journeys were undertaken by participant UW6, however participant UW9 travelled the furthest over the data collection period, covering over 6776 kilometres. We note a large spread between participants in terms of the duration, the distance travelled, and the number of journeys. UW6 and UW9 travelled considerably further, over a higher number of journeys than the other participants. However, most journeys average around 20–30km in length.

Figure 3.6 details the normalised distribution in journey lengths over the

[1]This was due to difficulties in participant recruitment.

Table 3.4: Number of unique endpoints and routes per participant.

| Dataset | Number of Unique Endpoints | Number of Unique Route Combinations |
|---|---|---|
| uw1 | 31 | 55 |
| uw2 | 26 | 48 |
| uw4 | 23 | 49 |
| uw6 | 85 | 163 |
| uw9 | 79 | 162 |

Table 3.5: Number of common endpoints per participant.

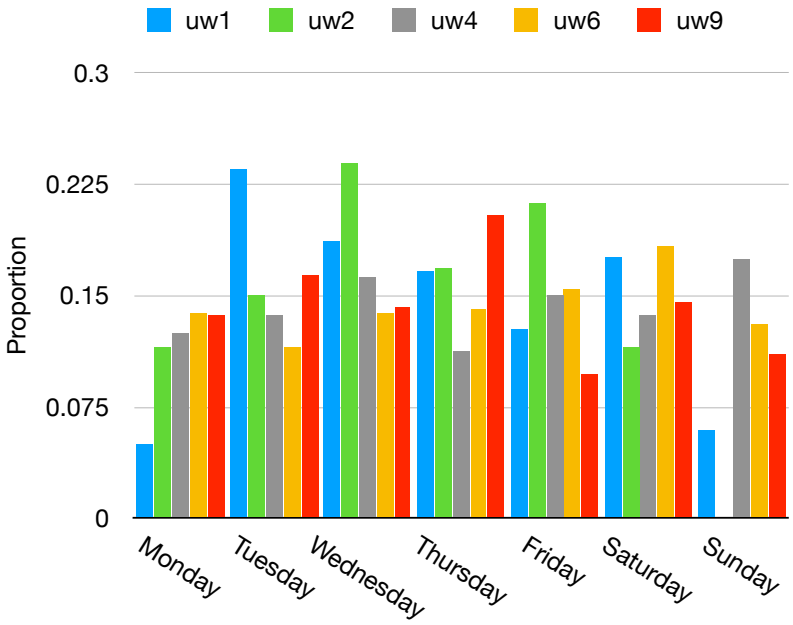
| | uw1 | uw2 | uw4 | uw6 | uw9 |
|---|---|---|---|---|---|
| uw1 | 31 | 2 | 1 | 3 | 3 |
| uw2 | | 26 | 2 | 5 | 4 |
| uw4 | | | 23 | 6 | 5 |
| uw6 | | | | 85 | 6 |
| uw9 | | | | | 79 |

participants. We have binned the data into 4 categories, very short (less than 5km), short (5–25km), medium (25–100km) and long (greater than 100km) journeys. As expected, the proportion of long journeys is the smallest for all participants. Participants UW2, UW4 and UW6 have the majority of their journeys between 5–25km, compared to UW1 and UW9 who have the majority of journeys being between 25–100kms.

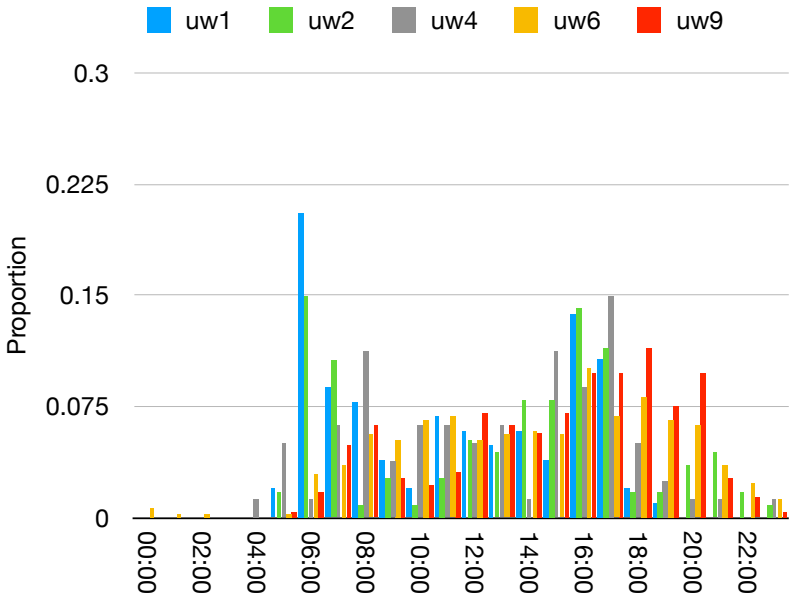
If we consider the number of days with journey data, UW6 has the most, with 114. The other participants range between 34–65 days. We define the number of days as the amount of days where the participant has undertaken at least one journey.

When evaluating the temporal aspects of the data, we can see different trends between participants. UW1 and UW2 have a reduced number of journeys on the weekend compared to the weekday, especially on Sunday, where UW2 has no journeys in the dataset. UW4 and UW6 see an increased amount of journeys at the weekend, with Sundays and Saturdays having the majority of journeys for UW4 and UW6 respectively. If we focus on the hour in the day (see Figure 3.7b), we can see that 6am is when the most journeys commence for UW1 and UW2. Participants UW4, UW6 and UW9 have the most frequency hours of commencing a journey at 5pm, 4pm and 6pm respectively. No journey commenced after 8pm for UW1 and only UW6 has commenced a journey between midnight and 3am. No journeys in the dataset begin at 3am.

Table 3.4 summarises the number of unique endpoints for each participant, where an endpoint can be either the start or end of the journey. UW1, UW2, and UW4 have between 23–31 endpoints, while UW6 and UW9 have a considerably higher amount of 85 and 79 respectively. This shows that UW6 and UW9




(a) Day of week.




(b) Hour of day.

Figure 3.7: Temporal distribution of journeys per participant in the POLD.

have travelled to more locations than the other participants, which could be a consequence of a less structured routine or the increase amount of data on these participants. The number of common endpoints is shown in Table 3.5. We can see that UW6 has 6 endpoints in common with both UW4 and UW9, whereas in contrast UW1 and UW4 share the least similarity in locations, with only one common endpoint in their datasets.

### 3.2.2 Ethical Considerations

Due to the nature of the personal information contained within the dataset, ethical considerations had to be made. Prior to the start of collection, an application was written to the Warwick Biomedical & Scientific Research Ethics Committee (BSREC) outlining the specifics of the data collection task. The application includes details of the manner in which the data collection would take place, provided appropriate information sheets and consent forms for each participant, in addition to addressing data privacy and the safeguards put in place to protect the data. The data collection received full BSREC approval[2].

### 3.2.3 Limitations

There is an insufficient amount of data on each participant to extensively model their pattern of life. Due to only having a single project vehicle at a time, participants had 1, 2 or 3 consecutive-week periods with the vehicle, with gaps of several weeks in-between having the project vehicle. This can cause trends journeys to be hidden or even lost entirely. Participant availability and other factors also caused a disparity in the number of weeks collected per participant, with some having considerably more than others. One of the project vehicles was not suitable for the normal day-to-day use of most of the participants (e.g., being unsuitable for fitting child seats), causing participants to not use the project vehicle for some journeys. The dataset suffers from a lack of participants. We collected data from other participants, but due to a combination of a lack of journeys, and difficulties in obtaining a ground truth for the journeys, they were omitted from the dataset. The dataset also suffers from a limited sample in terms of gender and other demographic factors. Additionally, the age range of the participants misses out the older bracket.

For future data collection of this type, we suggest the following recommendations:

i multiple project vehicles should be available for data collection to occur in parallel;

[2]BSREC Application Reference: REGO-2017-2036, approved 18 October 2017

ii the suitability of the project vehicle should be considered, and ensure that all participants can use it as their daily vehicle;

iii the length of consecutive weeks should be fixed for the entire collection period, and be a minimum of 4 weeks;

iv participants should have specific criteria to log on each journey, that can be entered via a mobile application or in-vehicle device;

v vehicle re-fuelling should be completed by the participants, to better model their pattern of life[3].

## 3.3 Caltrain Dataset

The Caltrain dataset is a taxi trajectory dataset consisting of journeys from approximately 500 taxis, all within the San Francisco Bay Area [130]. We take a subset of this dataset, which is a 6.327 $\times$ 6.827km grid, containing all journeys commencing from the Caltrain Station in San Francisco, as pre-processed by Besse *et al.* [23]. We refer to this subset as the Caltrain dataset for the remainder of this thesis. In this dataset, 4,127 trajectories are present over a period of 24 days, starting on the 17th May and ending on the 9th June 2008. The dataset contains trajectories with a data point every minute, resulting in a dataset of 44832 instances.

Figure 3.8 shows a plot of all of the trajectories within the dataset. From this, the reader will find the roads in San Francisco form a grid-like structure, with few roads containing bends. Additionally, the majority of journeys seem to be to the north of the station, with a small dense area containing most of the destinations. There are only a few journeys to the east of the station, as there is only a small area east until you reach San Francisco Bay, a large body of water.

On further examination of the dataset, the average journey is 3km in length and 9 minutes 24 seconds in duration. Figure 3.9a illustrates a fairly even distribution over the number of weekdays, with Sunday standing out with noticeably fewer journeys than the others. On examining the hour of the day (see Figure 3.9b), we see journeys peak at 6pm, with 8–10am and 5–8pm being the busiest hours in general. As expected, 12am–7am has the fewest journeys, with hardly any in between 1–5am. Figure 3.10 details the distribution in length of journeys. The data shows that most journeys are between 2–3km in length, and few are over 8km.

[3] Although this creates unreasonable financial complications.

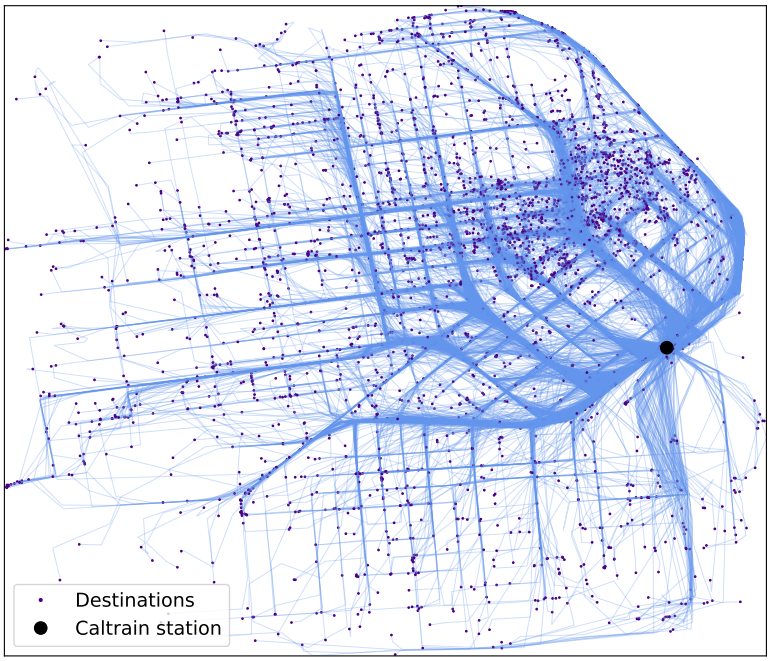


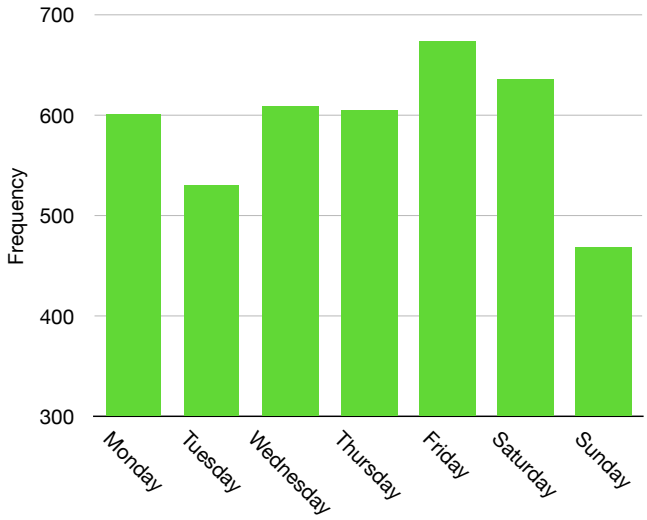

Figure 3.8: Plot of trajectories and destinations within the Caltrain dataset [23].

## 3.4 Porto Dataset

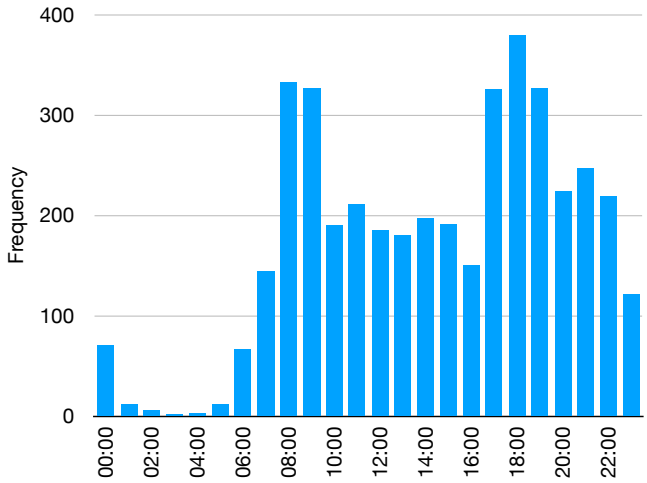
The Porto dataset is a taxi trajectory dataset from 442 taxis running in the city of Porto [1]. The data collection period consisted of a complete year (from 01/07/2013 to 01/07/2014). Besse *et al.* [23] generated a subset of this dataset, and we refer to this subset as the Porto dataset in the remainder of this thesis. The dataset contains 19,423 journeys that start from the Sao Bento station within the city, and end within a 8.116 $\times$ 8.068km grid. The dataset contains 645096 instances, with a sampled rate of one data point every 15 seconds.

Unlike the Caltrain dataset, Figure 3.11 shows a contrasting structure to the road network for the Porto dataset. Although there are a few straight roads, most of the road network consists of multiple bends and a less consistent approach to the space of nearby roads and junctions. In general, most journeys occur north of station, with the majority of destinations situated nearby. With a few exceptions, the destinations become sparse towards the outer boundaries of the grid.

Despite the larger area, the Porto dataset has on average a smaller journey length and duration of 2.4km and 8 minutes 3 seconds. Figures 3.12a and 3.12b highlight the distribution in days and hours of the journeys, seeing similar trends in time, with peak hours around 8–11am and 1–4pm, and a lull in activity from 1–6am. Examining the distribution in the day of the week, we find an increase in journeys on Sundays, with Fridays and Saturdays being the




(a) Day of week.




(b) Hour of day.

Figure 3.9: Temporal distribution of journeys in the Caltrain dataset.

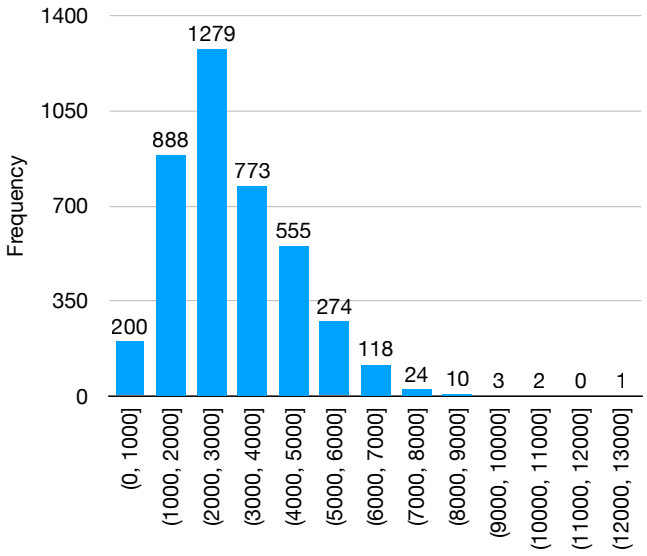



Figure 3.10: Distribution of journey length in the Caltrain dataset.

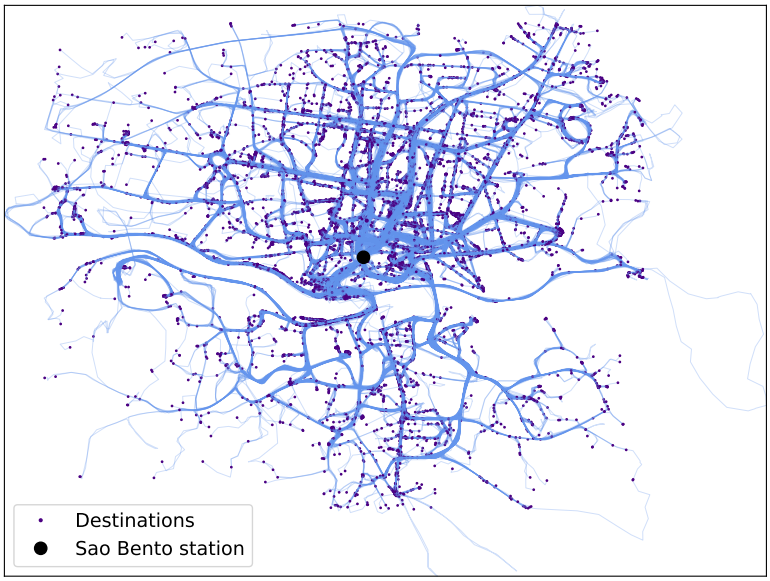
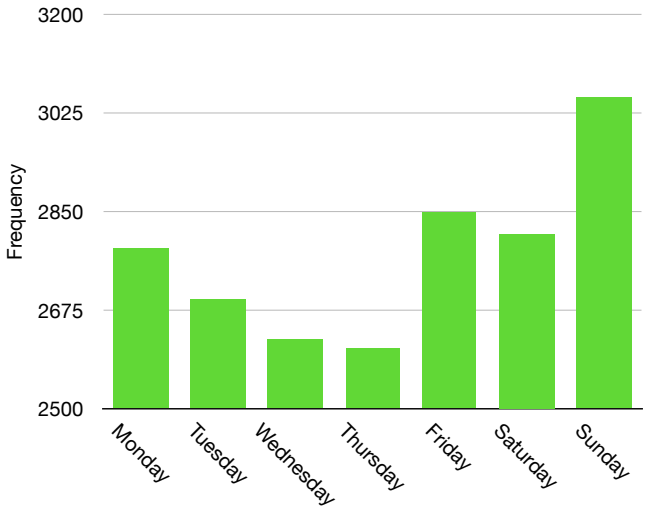
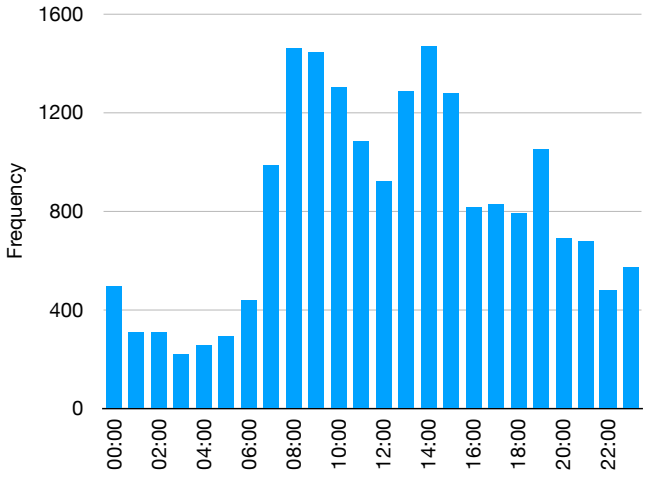



Figure 3.11: Plot of trajectories and destinations within the Porto dataset [23].




(a) Day of week.




(b) Hour of day.

Figure 3.12: Temporal distribution of journeys in the Porto dataset.

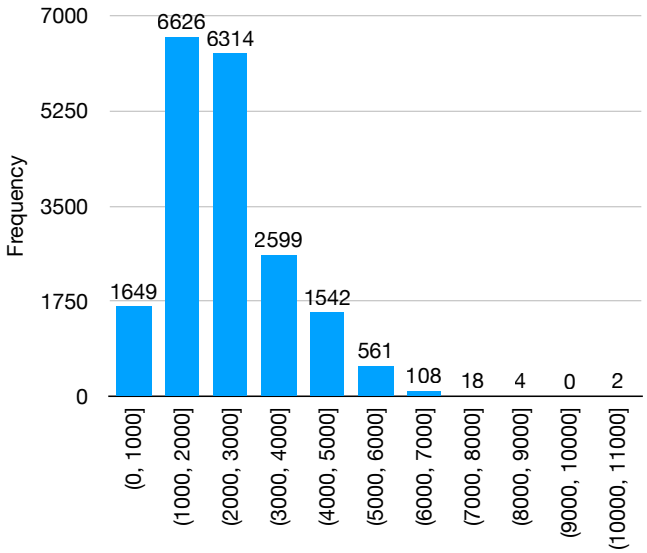



Figure 3.13: Distribution of journey length in the Porto dataset.

most popular among taxi journeys. Many of the journeys are between 1–3km in length (see Figure 3.13), with a relatively small proportion over 6km.

## 3.5 Summary

In this chapter we discussed four vehicle trajectory datasets, detailing the collection methodology for the two that were collected for this thesis. We described the motivations behind collecting our datasets, and presented the drawbacks of the POLD, with suggestions of improvements for future collection. The main benefits of the LED and POLD that we collected are that a wealth of on-board vehicle data exists, along with a ground truth of activities and destinations for the LED and POLD respectively. We use the LED for our investigation into point of interest extraction in Chapter 4, and subsequently the POLD to evaluate the proposed method on normal driving data. Our work on destination prediction in Chapter 5 uses the POLD. Chapter 6 uses the POLD, Caltrain and Porto datasets to evaluate our proposed destination prediction method. A key difference the Caltrain and Porto datasets exhibit is the lack of on-board vehicle data. This presents a challenge in Chapter 6, resulting in the evaluation focusing on the spatio-temporal properties of the trajectories.

# Chapter 4

# Activity-based Vehicle PoI Extraction

Knowledge of drivers' mobility patterns is useful for enabling context-aware intelligent vehicle functionality, as discussed in Chapter 2. Such patterns are often described in terms of the points of interest (PoIs) visited by an individual. However, existing stay point 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. This chapter presents Activity-based Vehicle PoI Extraction (AVPE), a wrapper method around existing stay point extraction methods, that utilises a post-clustering classification stage to filter out false PoIs. We evaluate the benefits of AVPE compared to three state-of-the-art general purpose stay point extraction algorithms and demonstrate the effectiveness of AVPE when applied to real-world driving data.

## 4.1 Introduction

Point of interest (PoI) extraction is useful for automatically discovering locations that are relevant to an individual 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 customise a vehicle for a given trip, for example altering the climate control or tailoring the media settings. Previous work on stay point extraction typically uses periods of low movement to detect PoIs, in applications such as detecting mobility patterns in a city [11, 71] or animal migration patterns [49]. 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 stay point where the vehicle has stopped for an intended purpose, whether that be to park, drop-off a passenger,

or to 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 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 [107] or categorising usage in a vehicle context [37], the presence of erroneous PoIs can significantly skew the results. 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 are 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 chapter, we (i) present Activity-based Vehicle PoI Extraction (AVPE), a wrapper around existing stay point extraction methods that uses a post-clustering classification stage to filter out false PoIs from the extraction process, (ii) evaluate AVPE against three state-of-the-art general purpose stay point extraction algorithms, and (iii) demonstrate its effectiveness when applied to real-world driving data. We analyse 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 chapter is organised as follows. Section 4.2 presents AVPE, our proposed wrapper method for PoI extraction. In Section 4.3, we describe our experimental methodology, the parameters and the vehicles used in our evaluation, and detail the specifics of the datasets. Section 4.4 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 4.5 concludes the chapter.

## 4.2 Activity-based Vehicle PoI Extraction (AVPE)

In this chapter, we present Activity-based Vehicle PoI Extraction (AVPE), a novel wrapper method which 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 stay points where the vehicle has stopped for a specific task (such as picking up a passenger or using a drive-through service), and should be distinguished from false PoIs (such as

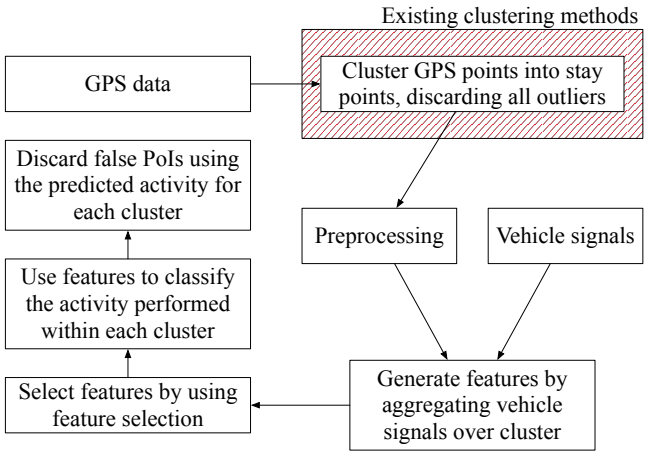



Figure 4.1: Overview of AVPE.

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 chapter, we consider CB-SMoT, STA and GVE as base clustering algorithms, that were introduced in Section 2.5.1. These algorithms were selected as they were developed for stay point extraction (and successfully used) by Bamis *et al.* [18], Palma *et al.* [126] and Thomason *et al.* [157, 158]. 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 which helps prevent the presence of fragmented clusters (explained in Section 4.2.2). 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 (as defined in Section 3.1.2), where some activity types (positive activities) represent true PoIs and others (negative activities) represent the common types of false PoI extracted by the clustering methods, which are introduced below. AVPE is therefore able to determine whether a candidate PoI is relevant or not. The approach of using predefined activities, and the separation into positive and negative activities, corresponding to true and false PoIs respectively, is fundamental to AVPE. Similarly, well-defined transitions between activities are required to ensure consistency. While we use an example set of activities and transitions in this chapter (see Chapter 3 for a description of the activities and transition definitions), our focus is on the AVPE method,

rather than on a particular set of activities. 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 additional 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 4.1. AVPE uses a combination of vehicle signals and GPS data, and, as defined in Section 2.4.1, an instance $x_j$ at time $j$ is a tuple $x_j = \langle lat, long, V \rangle$ 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 thesis.

AVPE requires training on a labelled set of data before it can be used on unseen trajectories. We assume that a set of activities, $L$, is defined, where the positive activities, $L^+ \subset L$, are activities that are of interest and correspond to true PoIs, and negative activities, $L^- = L \setminus L^+$, 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 activities and transition definitions are used to assign a activity for each instance. At the start of training AVPE the training data, $\mathcal{T}_{train}$, is input and the instances within the trajectories are clustered using the spatio-temporal data, only keeping periods of low mobility (which are referred to as stay points). Adjacent clusters up to $\lambda$ seconds apart are then merged together. These clusters are then used to train a classifier, $\psi$. 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) [29]. If the current AUC is greater than any that has been previously seen, the record of the best classifier and feature set combination is updated accordingly. The best overall performing classifier and feature set identified are output by the training algorithm. Algorithms 1 and 2 describe the pre-processing stage of AVPE (including the post-clustering merging of clusters), and the vehicle signal feature extraction respectively, with Table 4.1 defining the functions used within the algorithms in this chapter. 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 pre-trained classifier (created using Algorithm 3), (iv) the feature set required by the pre-trained classifier, and (v) the choice of clustering algorithm with pre-trained parameters.

Table 4.1: Functions used when defining AVPE.

| Notation | Description |
|---|---|
| $time(x_j)$ | A function that returns the time $j$ of the instance $x_j$ |
| $head(c_m^i)$ | A function that returns the first instance in cluster $c_m^i$ |
| $last(c_m^i)$ | A function that returns the last instance in cluster $c_m^i$ |
| $delete(c_m^i)$ | A function that deletes the cluster $c_m^i$ |
| $split(\mathcal{C}, k)$ | A function that returns an array of training and validation clusters for a given number of folds, $k$ |
| $truth(c_m^i)$ | A function that returns the ground truth classification label for $c_m^i$ |
| $score(TP, FP, TN, FN)$ | A function that returns the AUC |
| $filter(\omega, F)$ | A function that returns the feature values in $F$ for the features that are present in feature set $\omega$ |

### 4.2.1 Base Clustering of Trajectories

AVPE begins with a pre-processing stage, as defined in Algorithm 1. This starts by generating clusters (in which a cluster is a strictly ordered sub-sequence of instances) of instances 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 outlier 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. As discussed earlier in this chapter, we consider CB-SMoT [126], STA [18] and GVE [158] as representative clustering algorithms.

Clustering algorithms typically have parameters that can significantly alter the output that is generated. To optimise the parameters for each clustering algorithm, we perform simulated annealing [93] using the training set. To compare the performance of a given parameter combination, we aim to maximise the number of non-driving instances that are clustered, while minimising the number of driving instances that are clustered. This performance is quantified using the Sørensen-Dice coefficient (set overlap) metric [53, 151], defined as,

$$QS = \frac{2|A \cap B|}{|A| + |B|}, \tag{4.1}$$

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.* [171], define specific error types, enabling each kind of error to be individually weighted.

We conducted an initial investigation [162], prior to proposing AVPE, which used both set overlap and the Ward metric [171] to measure the performance in our parameter search. This was performed on a subset of the LED, and we

found that using set overlap results in clustering parameters being identified that give a higher overall classification performance than those found when using the Ward metric. Therefore, in the remainder of this thesis we use set overlap as the metric from which parameters are determined.

**Algorithm 1:** *preprocess*($cluster(), \mathcal{T}, \lambda$) — Pre-processing stage of AVPE.

**inputs :** $\mathcal{T}$, the set of $n$ trajectories, $\{t_1, ..., t_n\}$
$\lambda$, the merge threshold
$cluster()$, the chosen clustering algorithm, with pre-trained parameters

**output :** $\mathcal{C}$, a set of pre-processed clusters

$\mathcal{C} = cluster(\mathcal{T})$
**if** $\lambda > 0$ **then**
  `// merge adjacent clusters up to` $\lambda$ `seconds apart`
  **for** $t_i \in \mathcal{T}$ **do**
    **for** $c_m^i \in \mathcal{C}$ **do**
      `// calculate time difference between current and previous cluster`
      $q = time(head(c_m^i))$
      $p = time(last(c_{m-1}^i))$
      **if** $(q - p) < \lambda$ **then**
        `// append all instances from start of previous cluster to end of current cluster`
        $c_{m-1}^i = c_{m-1}^i \oplus s_{p+1,q-1}^i \oplus c_m^i$
        $delete(c_m^i)$
      **end**
    **end**
  **end**
**end**
**return** $\mathcal{C}$

### 4.2.2 Post-cluster Merging

Due to the nature of the trajectories and the vehicle activities, multiple clusters can be generated that are part of the same activity, for example drive-through and traffic activities. Such activities 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 4.2 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 overall output should be a single traffic activity, however all three clustering algorithms considered in this chapter have the potential to separate this traffic

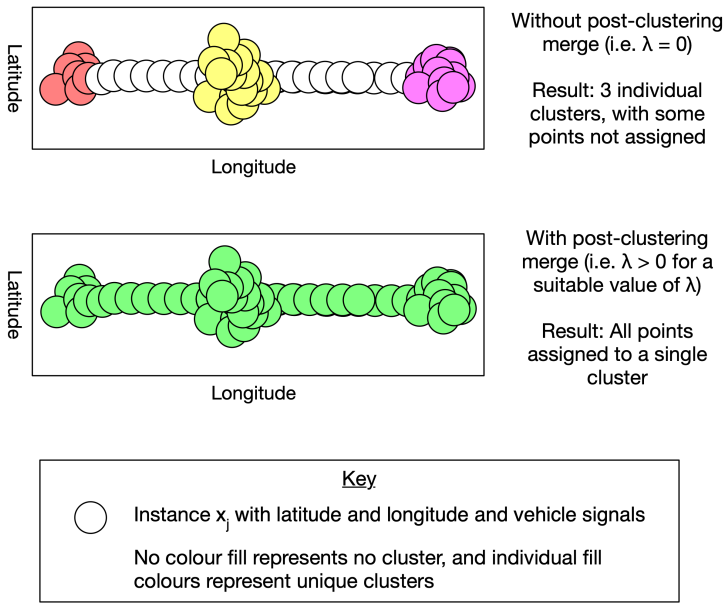



Figure 4.2: Example of a fragmented traffic activity, and how post-cluster merging can mitigate this.

activity 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 pre-processing stage of AVPE. We define a merge threshold, $\lambda$, 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 $c_m^i$ and the time of the last instance in cluster $c_{m-1}^i$. If the difference in time between these two instances is less than $\lambda$ then $c_{m-1}^i$ and $c_m^i$ will be merged. Clusters are merged by concatenating the sequence of instances in the previous cluster, the current cluster and any instances that are temporally between them. This helps to reduce fragmented clusters (as illustrated in Figure 4.2), aiding the classification stage.

### 4.2.3 Signal Aggregation and Classification

The training stage of AVPE, as detailed in Algorithm 3, takes a set of training trajectories, $\mathcal{T}_{train}$, and a merge threshold, $\lambda$, along with the chosen methods for clustering, feature selection and classification, and the number of folds to use for cross validation, $k$. The output of the training stage is a trained classifier, $\psi^*$, and the feature set used in the classifier, $\omega^*$.

The training algorithm first pre-processes 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 activity to be applied.

**Algorithm 2:** $features(c_m^i)$ — Extracting features for a cluster.

**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 $a = time(head(c_m^i))$
2 $b = time(last(c_m^i))$

`// calculate features using element-wise operations over matrices`
`// calculate different features for real-valued, categoric and binary types`
3 $F = F \cup max(V_{a,b}^R) \cup min(V_{a,b}^R) \cup mean(V_{a,b}^R) \cup range(V_{a,b}^R)$
4 $F = F \cup stdev(V_{a,b}^R) \cup firstderivative(V_{a,b}^R) \cup timeabvmean(V_{a,b}^R)$
5 $F = F \cup max(V_{a,b}^N) \cup min(V_{a,b}^N) \cup mean(V_{a,b}^N) \cup range(V_{a,b}^N)$
6 $F = F \cup max(V_{a,b}^B) \cup min(V_{a,b}^B) \cup mean(V_{a,b}^B) \cup range(V_{a,b}^B)$
`// calculate element-wise delta operation over specific binary signals`
7 $F = F \cup delta(V_{a,b}^B$ where $V^B \in deltaSignals)$
8 **return** $F$

Should a majority class containing greater than 50% of the instances not exist, then the cluster is discarded.

With pre-processing 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 are 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,

**Algorithm 3:** Activity-based Vehicle PoI Extraction (AVPE) — Training stage.

**inputs :** $\mathcal{T}_{train}$, a set of $n$ training trajectories, $\{t_1, ..., t_n\}$
$\lambda$, the merge threshold
$cluster()$, the chosen clustering algorithm, with pre-trained parameters
$selection()$, the chosen feature selection algorithm
$classificationMethod()$, the chosen classification method
$k$, the number of folds to use for cross validation

**output :** $\psi^*$, a trained classifier
$\omega^*$, the feature set used in the classifier $\psi^*$

$\mathcal{C}_{train}, \mathcal{C}_{validation}, \psi^*, \omega^*, AUC = [\,]$
$\mathcal{C} = preprocess(cluster(), \mathcal{T}_{train}, \lambda)$
`// split training and validation data`
$\mathcal{C}_{train}, \mathcal{C}_{validation} = split(\mathcal{C}, k)$
**for** $F_{num} \in count(1, |F|)$ **do**
  $TP, FP, TN, FN = [\,]$
  **for** $k' \in count(1, k)$ **do**
    `// calculate features for each cluster`
    **for** $c_m^i \in \mathcal{C}_{train}[k']$ **do**
      $F = F \cup features(c_m^i)$
    **end**
    `// select features and train classifier`
    $\omega = selection(F_{num}, F)$
    $\psi = train(classificationMethod(), filter(\omega, F))$
    **for** $c_m^i \in \mathcal{C}_{validation}[k']$ **do**
      $\phi = \psi(filter(\omega, features(c_m^i)))$
      `// compare the prediction to ground truth`
      **for** $l \in L$ **do**
        **if** $\phi = l \wedge \phi = truth(c_m^i)$ **then** $TP[l]$ += 1
        **if** $\phi = l \wedge \phi \neq truth(c_m^i)$ **then** $FP[l]$ += 1
        **if** $\phi \neq l \wedge \phi = truth(c_m^i)$ **then** $TN[l]$ += 1
        **if** $\phi \neq l \wedge \phi \neq truth(c_m^i)$ **then** $FN[l]$ += 1
      **end**
    **end**
  **end**
  `// store the classifier, feature set and AUC`
  $\psi^*[F_{num}] = \psi$
  $\omega^*[F_{num}] = \omega$
  $AUC[F_{num}] = score(TP, FP, TN, FN)$
**end**
`// determine # features with the Kneedle algorithm`
$F^*_{num} = kneedle(AUC)$
**return** $\psi^*[F^*_{num}], \omega^*[F^*_{num}]$

$\mathcal{C}_{train}[k']$) are input into the chosen feature selection algorithm (such as Minimal Redundancy Maximal Relevance [129] or Principal Component Analysis [4]), 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 pre-judge 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 activities. The prediction is compared to the ground truth for each activity, 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 [141], 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 chapter, this simple approach was used.

### 4.2.4 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 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, $\lambda$, 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 pre-processing stage is identical to that used in the training stage of AVPE, generating clusters from the trajectories and merging consecutive clusters that occur within the defined time threshold. After the trajectories have been pre-processed 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

**Algorithm 4:** Activity-based Vehicle PoI Extraction (AVPE) — Deployment stage.

**inputs :** $\mathcal{T}$, a set of $n$ trajectories in the buffer, $\{t_1, ..., t_n\}$
$\lambda$, the merge threshold
$\psi$, the pre-trained classifier
$\omega$, the feature set used in the classifier $\psi$
$cluster()$, the chosen clustering algorithm, with pre-trained parameters

**output :** $\mathcal{C}'$, 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   $\phi = \psi(filter(\omega, features(c_m^i)))$
    `// if the prediction is in the set of positive activities, add the cluster to our return set`
4   **if** $\phi \in L^+$ **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** $\mathcal{C}'$

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 activities, $L^+$, then the cluster is added to the return set. This process is repeated for all clusters obtained from the trajectories in the buffer, and the set of clusters that are considered to be relevant is returned.

## 4.3 Experimental Methodology

In order to demonstrate and evaluate AVPE, we use a set of activities and transitions (as defined in Chapter 3), and select a set of vehicle signals to be used. In this section, we detail the implementation specifics of AVPE as evaluated in this chapter, including the parameter values, data collection methodology and attributes of the datasets used.

### 4.3.1 Activity Labelling

As described earlier in Section 4.2, the activity types and the transitions between these types are fundamental to AVPE. The set of activities and transitions defined in Chapter 3 are illustrative and are to enable our evaluation, however,

Table 4.2: 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 |

they can be tailored depending on the application. For the PoI extraction task, we use the 8 activities described in Section 3.1.2. Although introducing 8 activities 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 can be valuable in developing subsequent applications. This more nuanced set of activities 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 categorising vehicle usage. Given this, we do not consider a binary classification. The activities used in this chapter are separated into positive and negative activities (as defined in Section 4.2). The positive activities are **Drive-through**, **Drop-off**, **Parked** and **Pick-up**, whereas the negative activities are **Barrier**, **Driving**, **Manoeuvre** and **Traffic**. We classify manoeuvre and barrier as negative activities since, although they may indicate leaving or arriving at a PoI, they will always be adjacent to a positive activity. We define separate activities for drop-off and pick-up since this can aid further applications that use the activities, such as destination prediction. Chapter 3 provides full descriptions of the activities and Appendix A documents all of the transition tables between each activity.

Table 4.2 shows the 22 vehicle signals that were used for activity classification, which were selected using both intuition and domain expertise. In this chapter, the signals from the vehicle are expanded, using common statistical properties, into the features described in Section 4.2 (minimum, maximum, range etc.) resulting in a total of 99 features. As discussed in Section 4.2.3, we

Table 4.3: Simulated annealing parameter space.

| Algorithm | Parameter | Increment | Start Range |
|---|---|---|---|
| CB-SMoT | $\mathcal{Q}_{area}$ | 0.01 | $0 \leq n \leq 1.0$ |
| CB-SMoT | $t_{min}$ | 1 | $0 \leq t \leq 10$ |
| GVE | $\alpha$ | 0.1 | $0 \leq \alpha \leq 2.5$ |
| GVE | $\beta$ | 3 | $1 \leq \beta \leq 50$ |
| GVE | $n_{points}$ | 3 | $1 \leq n \leq 50$ |
| GVE | $t_{max}$ | 20 | $10 \leq t \leq 1440$ |
| STA | $\mathcal{N}_{buf}$ | 2 | $1 \leq n \leq 15$ |
| STA | $\mathcal{D}_{thresh}$ | 0.1 | $0 < t \leq 10.0$ |

Table 4.4: The highest performing parameters for each clustering algorithm.

| Algorithm | Parameters Used |
|---|---|
| CB-SMoT | $\mathcal{Q}_{area} = 0.40$, $t_{min} = 20$ |
| GVE | $\alpha = 0.14$, $\beta = 160.0$, $n_{points} = 75$, $t_{max}(m) = 170$ |
| STA | $\mathcal{N}_{buf} = 4$, $\mathcal{D}_{thresh} = 1.43$ |

use feature selection to reduce the number of features as there is potential for overlap between features, and therefore redundant information. 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 are key indicators 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 to distinguish between Manoeuvre and Parked activities 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 chapter is to use sensors that are already on the vehicle and are common across multiple vehicles, a motivation guided 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 chapter.

### 4.3.2 Experimental Parameters

To optimise the parameters for each clustering algorithm, we perform simulated annealing using the Sørensen-Dice coefficient on the training set, as described in Section 4.2.1. A merge threshold of 0 seconds is used when optimising the parameters, to assess the performance of the clustering algorithm. This means that no clusters are merged together and the raw output of the clustering algorithms can be evaluated. Each clustering algorithm is run multiple times, with a single parameter being altered on each step. For each clustering algorithm, simulated annealing is performed 1000 times, evaluating over 3000 parameters in each iteration. We used the parameter space shown in Table 4.3, where the GVE and STA parameters correspond to those used by Thomason *et al.* [157]. Table 4.4 details the best performing parameters for each clustering algorithm, and these are the parameters used in this chapter.

To use the AVPE algorithm, we are required to instantiate the algorithm with a number of parameters, including the set of activities ($L$), a value for the merge threshold ($\lambda$), a classification algorithm, and a feature selection algorithm. The activities used are defined in Section 3.1.2, and we investigate a range of merge thresholds ($\lambda$), 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 [105, 113, 148, 168], and SVMs have previously been shown to be effective for activity prediction [80, 97, 137]. 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[1], since tuning the classification is not the focus of this chapter 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 [129]. Both the classifier and feature selection methods can be replaced with alternatives if required (such as Bayesian Inference [60] or Principle Component Analysis [4] respectively) since our approach is agnostic with respect to the methods used.

### 4.3.3 Data Collection

In order to evaluate AVPE, we defined a data collection methodology and collected two datasets, namely the LED and the POLD. The data collection methodology, and the specifics of each route in the LED are detailed in Chapter 3.

The LED is used to train the classifier and evaluate the performance of

[1]Weka implementations of the Random Forest and SVM classifiers were used [173].

Table 4.5: Summary of durations in the POLD per participant.

| Participant | Duration (s) | | | |
|---|---|---|---|---|
| | **Minimum** | **Maximum** | **Average** | **Total** |
| A | 487 | 2948 | 1621 | 45387 |
| B | 189 | 6005 | 1162 | 55772 |
| | **189** | **6005** | **1392** | **101159** |

Table 4.6: Summary of distances in the POLD per participant.

| Participant | Distance (m) | | | |
|---|---|---|---|---|
| | **Minimum** | **Maximum** | **Average** | **Total** |
| A | 1424 | 35930 | 22562 | 631735 |
| B | 446 | 95626 | 12169 | 584102 |
| | **446** | **95626** | **17366** | **1215837** |

AVPE. For our evaluation in this chapter, 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 that all of the journeys in the test set are unseen, meeting the out-of-sample principle.

The POLD comprises journeys and activities undertaken as though an individual was using their own personal vehicle. For the evaluation in this chapter, we take a subset of this dataset, specifically 4 weeks of data from 2 different participants (1 week for each participant and vehicle combination). We refer to this subset as the POLD for the remainder of this chapter, and describe the properties of this subset below (as they are different from the complete dataset described in Chapter 3). The POLD contains 76 journeys and 101,063 instances, totalling over 1,215 kilometres and around 28 hours of driving. 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 POLD is summarised in Tables 4.5 and 4.6. All of the journeys in the POLD are used for testing, since the classifier is trained on the LED, to demonstrate that AVPE is applicable to unscripted driving data.

## 4.4 Results

In this section, we first discuss the results of the state-of-the-art clustering algorithms, namely CB-SMoT [126], STA [18] and GVE [158] when applied to the LED. Following this, we present the results of the activity classification stage, evaluating the performance using accuracy and AUC. Finally, we analyse the performance of AVPE as a whole, looking at the trade-off between the

Table 4.7: Summary of the performance of the clustering algorithms.

| **Algorithm** | **Merge threshold, $\lambda$ (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 |

removal of true positives and false positives (in which the goal is to remove false positives). We define a percentage reduction metric to evaluate this.

### 4.4.1 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 [158] produces the highest number of instances, along with the most clusters. STA [18] gives the fewest clusters, and CB-SMoT [126] returns the fewest instances. When investigating these results further, we find some interesting properties, as discussed below.

GVE missed 28 clusters (or activities), of which 2 were pick-up activities and 26 parked. All of the missed clusters were less than 48 seconds in duration, with the highest durations being the pick-up activities. The parked activities that are not recorded are 2–13 seconds in duration, with the majority being located in multi-story 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 activities are 20–48 seconds in duration and occur directly after long parked activities. The slight increase in movement compared to that of the parked activities appears to prevent these pick-ups from being captured.

Similarly to GVE, 31 clusters are missed by STA, 29 of which are parked activities. The remaining missed clusters are 2 pick-up activities, which are the same ones discussed above for GVE. The missed parked activities are between 2–14 seconds in duration and a similar combination of cluster length and GPS inaccuracy causes them to be missed.

CB-SMoT misses 30 activities in total, of which 26 are parked clusters,

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.52 | 0.00 | 0.17 | 0.05 | 0.00 | 0.00 | 0.26 | 0.00 |
| 0.09 | 0.68 | 0.00 | 0.05 | 0.00 | 0.00 | 0.14 | 0.05 |
| 0.01 | 0.01 | 0.95 | 0.01 | 0.00 | 0.00 | 0.01 | 0.01 |
| 0.14 | 0.00 | 0.08 | 0.78 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.21 | 0.26 | 0.53 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.06 | 0.06 | 0.00 | 0.00 | 0.33 | 0.17 | 0.39 |
| 0.14 | 0.03 | 0.02 | 0.02 | 0.00 | 0.00 | 0.79 | 0.00 |
| 0.00 | 0.00 | 0.12 | 0.00 | 0.00 | 0.00 | 0.00 | 0.88 |

(a) CB-SMoT ($\lambda = 0$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.77 | 0.00 | 0.16 | 0.00 | 0.00 | 0.00 | 0.07 | 0.00 |
| 0.01 | 0.93 | 0.06 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.04 | 0.94 | 0.01 | 0.00 | 0.01 | 0.01 | 0.00 |
| 0.00 | 0.00 | 0.19 | 0.51 | 0.31 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.21 | 0.18 | 0.61 | 0.00 | 0.00 | 0.00 |
| 0.05 | 0.05 | 0.67 | 0.00 | 0.00 | 0.19 | 0.00 | 0.05 |
| 0.12 | 0.03 | 0.35 | 0.00 | 0.00 | 0.00 | 0.49 | 0.00 |
| 0.09 | 0.00 | 0.39 | 0.00 | 0.00 | 0.26 | 0.00 | 0.26 |

(b) GVE ($\lambda = 0$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.45 | 0.02 | 0.05 | 0.15 | 0.00 | 0.00 | 0.33 | 0.00 |
| 0.00 | 0.54 | 0.40 | 0.00 | 0.00 | 0.00 | 0.05 | 0.00 |
| 0.00 | 0.06 | 0.92 | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 |
| 0.07 | 0.00 | 0.12 | 0.38 | 0.43 | 0.00 | 0.00 | 0.00 |
| 0.03 | 0.01 | 0.18 | 0.32 | 0.45 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.04 | 0.54 | 0.00 | 0.00 | 0.00 | 0.17 | 0.25 |
| 0.15 | 0.02 | 0.04 | 0.03 | 0.00 | 0.00 | 0.76 | 0.00 |
| 0.53 | 0.00 | 0.37 | 0.00 | 0.00 | 0.05 | 0.05 | 0.00 |

(c) STA ($\lambda = 0$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.59 | 0.00 | 0.12 | 0.07 | 0.00 | 0.00 | 0.22 | 0.00 |
| 0.15 | 0.70 | 0.10 | 0.00 | 0.00 | 0.00 | 0.00 | 0.05 |
| 0.00 | 0.01 | 0.94 | 0.04 | 0.00 | 0.00 | 0.02 | 0.00 |
| 0.06 | 0.03 | 0.03 | 0.83 | 0.03 | 0.00 | 0.03 | 0.00 |
| 0.00 | 0.00 | 0.05 | 0.65 | 0.30 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.06 | 0.35 | 0.00 | 0.00 | 0.29 | 0.24 | 0.06 |
| 0.16 | 0.02 | 0.00 | 0.05 | 0.00 | 0.00 | 0.77 | 0.00 |
| 0.17 | 0.00 | 0.42 | 0.00 | 0.00 | 0.00 | 0.08 | 0.33 |

(d) CB-SMoT ($\lambda = 5$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.64 | 0.02 | 0.11 | 0.00 | 0.00 | 0.00 | 0.23 | 0.00 |
| 0.00 | 0.89 | 0.09 | 0.00 | 0.00 | 0.00 | 0.02 | 0.00 |
| 0.01 | 0.04 | 0.93 | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 |
| 0.00 | 0.02 | 0.34 | 0.29 | 0.33 | 0.00 | 0.02 | 0.00 |
| 0.03 | 0.00 | 0.41 | 0.18 | 0.38 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.14 | 0.05 | 0.21 | 0.00 | 0.00 | 0.00 | 0.58 | 0.03 |
| 0.00 | 0.00 | 0.29 | 0.00 | 0.00 | 0.21 | 0.43 | 0.07 |

(e) GVE ($\lambda = 5$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.53 | 0.04 | 0.05 | 0.07 | 0.00 | 0.00 | 0.31 | 0.00 |
| 0.00 | 0.77 | 0.15 | 0.00 | 0.00 | 0.00 | 0.05 | 0.02 |
| 0.00 | 0.02 | 0.96 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.02 | 0.00 | 0.10 | 0.55 | 0.32 | 0.00 | 0.02 | 0.00 |
| 0.03 | 0.00 | 0.16 | 0.41 | 0.39 | 0.00 | 0.01 | 0.00 |
| 0.00 | 0.00 | 0.65 | 0.00 | 0.00 | 0.17 | 0.00 | 0.17 |
| 0.03 | 0.05 | 0.00 | 0.03 | 0.00 | 0.00 | 0.86 | 0.02 |
| 0.38 | 0.00 | 0.12 | 0.00 | 0.00 | 0.12 | 0.31 | 0.06 |

(f) STA ($\lambda = 5$)

which varied in length between 4–15 seconds. Additionally, 2 drive-through activities, 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 behaviour is not beneficial to PoI extraction, since 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 activities. This caused the mean of the distances to be low, whilst keeping a relatively high standard deviation. Using the best performing parameters with this distribution caused a negative distance threshold to be generated, which resulted in no activities being extracted for the two journeys.

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.54 | 0.00 | 0.15 | 0.02 | 0.00 | 0.00 | 0.29 | 0.00 |
| 0.05 | 0.85 | 0.05 | 0.00 | 0.00 | 0.00 | 0.05 | 0.00 |
| 0.00 | 0.00 | 0.94 | 0.03 | 0.00 | 0.00 | 0.02 | 0.01 |
| 0.09 | 0.03 | 0.06 | 0.83 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.12 | 0.35 | 0.52 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.06 | 0.18 | 0.00 | 0.00 | 0.47 | 0.24 | 0.06 |
| 0.13 | 0.04 | 0.00 | 0.04 | 0.00 | 0.00 | 0.80 | 0.00 |
| 0.00 | 0.00 | 0.17 | 0.00 | 0.00 | 0.00 | 0.08 | 0.75 |

(g) CB-SMoT ($\lambda = 10$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.73 | 0.04 | 0.13 | 0.00 | 0.00 | 0.00 | 0.11 | 0.00 |
| 0.00 | 0.75 | 0.21 | 0.00 | 0.00 | 0.00 | 0.04 | 0.00 |
| 0.01 | 0.03 | 0.94 | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 |
| 0.00 | 0.00 | 0.33 | 0.33 | 0.30 | 0.00 | 0.04 | 0.00 |
| 0.04 | 0.00 | 0.36 | 0.11 | 0.49 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.79 | 0.00 | 0.00 | 0.21 | 0.00 | 0.00 |
| 0.20 | 0.09 | 0.16 | 0.00 | 0.00 | 0.00 | 0.53 | 0.02 |
| 0.00 | 0.00 | 0.25 | 0.00 | 0.00 | 0.17 | 0.50 | 0.08 |

(h) GVE ($\lambda = 10$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.48 | 0.02 | 0.08 | 0.08 | 0.00 | 0.00 | 0.33 | 0.02 |
| 0.01 | 0.71 | 0.20 | 0.00 | 0.00 | 0.00 | 0.05 | 0.03 |
| 0.00 | 0.02 | 0.96 | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 |
| 0.00 | 0.02 | 0.15 | 0.37 | 0.46 | 0.00 | 0.00 | 0.00 |
| 0.01 | 0.00 | 0.16 | 0.29 | 0.53 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.06 | 0.44 | 0.00 | 0.00 | 0.28 | 0.17 | 0.06 |
| 0.00 | 0.00 | 0.01 | 0.04 | 0.00 | 0.00 | 0.95 | 0.00 |
| 0.42 | 0.00 | 0.00 | 0.00 | 0.00 | 0.50 | 0.00 | 0.08 |

(i) STA ($\lambda = 10$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.51 | 0.03 | 0.15 | 0.03 | 0.00 | 0.00 | 0.28 | 0.00 |
| 0.05 | 0.60 | 0.10 | 0.00 | 0.00 | 0.00 | 0.20 | 0.05 |
| 0.00 | 0.00 | 0.95 | 0.04 | 0.00 | 0.00 | 0.01 | 0.00 |
| 0.00 | 0.03 | 0.03 | 0.94 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.08 | 0.65 | 0.27 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.06 | 0.25 | 0.00 | 0.00 | 0.38 | 0.25 | 0.06 |
| 0.10 | 0.08 | 0.00 | 0.04 | 0.00 | 0.00 | 0.78 | 0.00 |
| 0.00 | 0.00 | 0.33 | 0.00 | 0.00 | 0.08 | 0.33 | 0.25 |

(j) CB-SMoT ($\lambda = 20$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.67 | 0.02 | 0.04 | 0.00 | 0.00 | 0.00 | 0.26 | 0.00 |
| 0.01 | 0.84 | 0.08 | 0.00 | 0.00 | 0.00 | 0.07 | 0.00 |
| 0.00 | 0.04 | 0.93 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.03 | 0.00 | 0.19 | 0.41 | 0.38 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.33 | 0.17 | 0.50 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.09 | 0.27 | 0.00 | 0.00 | 0.36 | 0.18 | 0.09 |
| 0.09 | 0.07 | 0.02 | 0.00 | 0.00 | 0.00 | 0.81 | 0.00 |
| 0.00 | 0.08 | 0.08 | 0.00 | 0.00 | 0.58 | 0.08 | 0.17 |

(k) GVE ($\lambda = 20$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.69 | 0.02 | 0.05 | 0.00 | 0.00 | 0.00 | 0.21 | 0.02 |
| 0.00 | 0.45 | 0.42 | 0.02 | 0.00 | 0.00 | 0.05 | 0.07 |
| 0.01 | 0.02 | 0.94 | 0.01 | 0.00 | 0.00 | 0.01 | 0.00 |
| 0.00 | 0.00 | 0.24 | 0.37 | 0.39 | 0.00 | 0.00 | 0.00 |
| 0.04 | 0.02 | 0.40 | 0.19 | 0.35 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.72 | 0.00 | 0.00 | 0.00 | 0.17 | 0.11 |
| 0.11 | 0.02 | 0.00 | 0.00 | 0.00 | 0.00 | 0.86 | 0.02 |
| 0.58 | 0.08 | 0.00 | 0.00 | 0.00 | 0.00 | 0.33 | 0.00 |

(l) STA ($\lambda = 20$)

Figure 4.3: Confusion matrices for Random Forest classifiers trained on the CB-SMoT, GVE and STA clusters with different merge thresholds.

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 4.7 summarises 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 favoured algorithm of the three considered, since the priority in selecting a clustering algorithm is to minimise the number of false PoIs, 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.

### 4.4.2 Activity Classification & 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 4.2. Minimising 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 [141] as shown in Algorithm 3. The number of features selected varied between 11–30 for the Random Forest, and 6–34 for the SVM classifiers. The mean and standard deviation for the number of features was 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 which both required fewer features than CB-SMoT. In contrast, the SVM

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.62 | 0.02 | 0.19 | 0.02 | 0.02 | 0.00 | 0.12 | 0.00 |
| 0.09 | 0.77 | 0.00 | 0.05 | 0.00 | 0.00 | 0.09 | 0.00 |
| 0.00 | 0.01 | 0.96 | 0.00 | 0.01 | 0.00 | 0.01 | 0.01 |
| 0.00 | 0.03 | 0.11 | 0.83 | 0.03 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.05 | 0.81 | 0.14 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.11 | 0.11 | 0.00 | 0.00 | 0.17 | 0.17 | 0.44 |
| 0.17 | 0.02 | 0.02 | 0.02 | 0.00 | 0.00 | 0.78 | 0.00 |
| 0.00 | 0.00 | 0.12 | 0.00 | 0.00 | 0.00 | 0.00 | 0.88 |

(a) CB-SMoT ($\lambda = 0$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.71 | 0.04 | 0.04 | 0.03 | 0.00 | 0.00 | 0.17 | 0.00 |
| 0.01 | 0.96 | 0.01 | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 |
| 0.01 | 0.07 | 0.90 | 0.01 | 0.00 | 0.00 | 0.01 | 0.00 |
| 0.02 | 0.00 | 0.14 | 0.29 | 0.56 | 0.00 | 0.00 | 0.00 |
| 0.04 | 0.00 | 0.16 | 0.18 | 0.61 | 0.00 | 0.00 | 0.00 |
| 0.05 | 0.10 | 0.52 | 0.00 | 0.00 | 0.00 | 0.00 | 0.33 |
| 0.12 | 0.05 | 0.02 | 0.02 | 0.00 | 0.00 | 0.78 | 0.00 |
| 0.13 | 0.09 | 0.09 | 0.00 | 0.00 | 0.43 | 0.00 | 0.26 |

(b) GVE ($\lambda = 0$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.65 | 0.00 | 0.00 | 0.13 | 0.00 | 0.00 | 0.22 | 0.00 |
| 0.03 | 0.67 | 0.24 | 0.00 | 0.00 | 0.00 | 0.05 | 0.01 |
| 0.01 | 0.04 | 0.94 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.03 | 0.02 | 0.15 | 0.33 | 0.47 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.18 | 0.32 | 0.49 | 0.00 | 0.00 | 0.00 |
| 0.33 | 0.04 | 0.42 | 0.00 | 0.00 | 0.00 | 0.00 | 0.21 |
| 0.06 | 0.01 | 0.01 | 0.03 | 0.00 | 0.01 | 0.88 | 0.00 |
| 0.16 | 0.05 | 0.00 | 0.00 | 0.00 | 0.79 | 0.00 | 0.00 |

(c) STA ($\lambda = 0$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.66 | 0.02 | 0.07 | 0.05 | 0.00 | 0.00 | 0.20 | 0.00 |
| 0.05 | 0.85 | 0.00 | 0.00 | 0.00 | 0.00 | 0.05 | 0.05 |
| 0.00 | 0.01 | 0.93 | 0.04 | 0.00 | 0.01 | 0.02 | 0.00 |
| 0.06 | 0.03 | 0.09 | 0.83 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.12 | 0.22 | 0.65 | 0.00 | 0.00 | 0.00 |
| 0.06 | 0.06 | 0.18 | 0.00 | 0.00 | 0.47 | 0.00 | 0.24 |
| 0.25 | 0.04 | 0.00 | 0.00 | 0.04 | 0.00 | 0.68 | 0.00 |
| 0.17 | 0.00 | 0.33 | 0.00 | 0.00 | 0.00 | 0.08 | 0.42 |

(d) CB-SMoT ($\lambda = 5$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.86 | 0.08 | 0.02 | 0.00 | 0.00 | 0.00 | 0.05 | 0.00 |
| 0.02 | 0.92 | 0.03 | 0.00 | 0.00 | 0.00 | 0.02 | 0.01 |
| 0.00 | 0.04 | 0.95 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.02 | 0.03 | 0.74 | 0.21 | 0.00 | 0.00 | 0.00 |
| 0.03 | 0.03 | 0.18 | 0.08 | 0.68 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.53 | 0.00 | 0.00 | 0.32 | 0.00 | 0.16 |
| 0.12 | 0.04 | 0.01 | 0.00 | 0.00 | 0.00 | 0.83 | 0.00 |
| 0.07 | 0.07 | 0.00 | 0.00 | 0.00 | 0.50 | 0.00 | 0.36 |

(e) GVE ($\lambda = 5$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.69 | 0.05 | 0.02 | 0.04 | 0.00 | 0.00 | 0.20 | 0.00 |
| 0.02 | 0.89 | 0.02 | 0.00 | 0.00 | 0.00 | 0.04 | 0.02 |
| 0.00 | 0.03 | 0.96 | 0.01 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.05 | 0.03 | 0.13 | 0.42 | 0.37 | 0.00 | 0.00 | 0.00 |
| 0.03 | 0.00 | 0.14 | 0.39 | 0.44 | 0.00 | 0.00 | 0.00 |
| 0.13 | 0.04 | 0.39 | 0.00 | 0.00 | 0.17 | 0.00 | 0.26 |
| 0.07 | 0.02 | 0.00 | 0.02 | 0.00 | 0.00 | 0.87 | 0.01 |
| 0.25 | 0.00 | 0.00 | 0.00 | 0.00 | 0.31 | 0.00 | 0.44 |

(f) STA ($\lambda = 5$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.66 | 0.02 | 0.10 | 0.02 | 0.00 | 0.00 | 0.20 | 0.00 |
| 0.05 | 0.90 | 0.05 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.01 | 0.01 | 0.94 | 0.03 | 0.00 | 0.00 | 0.02 | 0.00 |
| 0.03 | 0.03 | 0.17 | 0.77 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.02 | 0.00 | 0.12 | 0.08 | 0.75 | 0.00 | 0.02 | 0.00 |
| 0.06 | 0.06 | 0.18 | 0.00 | 0.00 | 0.41 | 0.24 | 0.06 |
| 0.33 | 0.02 | 0.00 | 0.02 | 0.00 | 0.00 | 0.64 | 0.00 |
| 0.17 | 0.00 | 0.08 | 0.00 | 0.00 | 0.00 | 0.08 | 0.67 |

(g) CB-SMoT ($\lambda = 10$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.87 | 0.05 | 0.00 | 0.00 | 0.00 | 0.00 | 0.07 | 0.00 |
| 0.01 | 0.86 | 0.06 | 0.00 | 0.00 | 0.00 | 0.06 | 0.02 |
| 0.00 | 0.03 | 0.96 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.09 | 0.11 | 0.61 | 0.20 | 0.00 | 0.00 | 0.00 |
| 0.05 | 0.04 | 0.24 | 0.13 | 0.55 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.36 | 0.00 | 0.00 | 0.36 | 0.00 | 0.29 |
| 0.18 | 0.05 | 0.02 | 0.00 | 0.00 | 0.00 | 0.75 | 0.00 |
| 0.08 | 0.08 | 0.00 | 0.00 | 0.00 | 0.50 | 0.00 | 0.33 |

(h) GVE ($\lambda = 10$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.63 | 0.06 | 0.00 | 0.10 | 0.00 | 0.00 | 0.21 | 0.00 |
| 0.04 | 0.87 | 0.05 | 0.00 | 0.00 | 0.00 | 0.00 | 0.04 |
| 0.01 | 0.01 | 0.96 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.04 | 0.00 | 0.19 | 0.26 | 0.52 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.22 | 0.19 | 0.59 | 0.00 | 0.00 | 0.00 |
| 0.06 | 0.11 | 0.39 | 0.00 | 0.00 | 0.28 | 0.00 | 0.17 |
| 0.04 | 0.04 | 0.00 | 0.04 | 0.00 | 0.00 | 0.89 | 0.00 |
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.75 | 0.00 | 0.25 |

(i) STA ($\lambda = 10$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.56 | 0.03 | 0.13 | 0.03 | 0.00 | 0.00 | 0.26 | 0.00 |
| 0.05 | 0.85 | 0.00 | 0.00 | 0.00 | 0.05 | 0.05 | 0.00 |
| 0.00 | 0.01 | 0.96 | 0.03 | 0.00 | 0.00 | 0.01 | 0.00 |
| 0.00 | 0.06 | 0.15 | 0.79 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.00 | 0.14 | 0.84 | 0.03 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.06 | 0.19 | 0.00 | 0.00 | 0.44 | 0.25 | 0.06 |
| 0.14 | 0.02 | 0.10 | 0.04 | 0.00 | 0.00 | 0.71 | 0.00 |
| 0.00 | 0.00 | 0.33 | 0.00 | 0.00 | 0.00 | 0.25 | 0.42 |

(j) CB-SMoT ($\lambda = 20$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.80 | 0.09 | 0.00 | 0.00 | 0.00 | 0.00 | 0.11 | 0.00 |
| 0.02 | 0.89 | 0.05 | 0.00 | 0.00 | 0.00 | 0.01 | 0.03 |
| 0.01 | 0.04 | 0.93 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.08 | 0.08 | 0.32 | 0.49 | 0.03 | 0.00 | 0.00 |
| 0.00 | 0.12 | 0.19 | 0.14 | 0.55 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.09 | 0.00 | 0.00 | 0.00 | 0.55 | 0.09 | 0.27 |
| 0.12 | 0.02 | 0.00 | 0.00 | 0.00 | 0.00 | 0.86 | 0.00 |
| 0.00 | 0.17 | 0.00 | 0.00 | 0.00 | 0.83 | 0.00 | 0.00 |

(k) GVE ($\lambda = 20$)

| Park | Driv | Traf | Pick | Drop | Barr | Mano | DThr |
|---|---|---|---|---|---|---|---|
| 0.67 | 0.00 | 0.05 | 0.10 | 0.00 | 0.00 | 0.19 | 0.00 |
| 0.05 | 0.57 | 0.28 | 0.05 | 0.00 | 0.00 | 0.02 | 0.03 |
| 0.04 | 0.01 | 0.91 | 0.02 | 0.00 | 0.00 | 0.01 | 0.00 |
| 0.20 | 0.00 | 0.00 | 0.41 | 0.39 | 0.00 | 0.00 | 0.00 |
| 0.19 | 0.02 | 0.02 | 0.42 | 0.35 | 0.00 | 0.00 | 0.00 |
| 0.00 | 0.06 | 0.06 | 0.00 | 0.00 | 0.67 | 0.00 | 0.22 |
| 0.05 | 0.00 | 0.07 | 0.05 | 0.00 | 0.00 | 0.79 | 0.04 |
| 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.42 | 0.00 | 0.58 |

(l) STA ($\lambda = 20$)

Figure 4.4: Confusion matrices for SVM classifiers trained on the CB-SMoT, GVE and STA clusters with different merge thresholds.

classifier used the most features with CB-SMoT at $\lambda = 5$ and $\lambda = 10$. Similarly, a higher number of features are used for both GVE and STA when using 5 and 10 second merge thresholds.

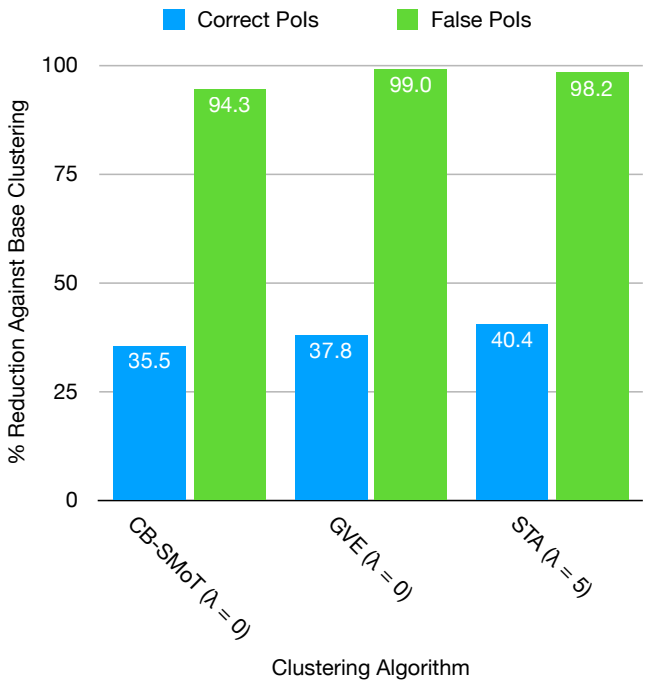
Once classifiers are trained with the selected number of features, we apply them to the test set. Table 4.8 details the accuracy and AUC achieved for each combination on the test set. We consider AUC as our main performance metric since, unlike accuracy, AUC is not biased towards class imbalance. The classification performance is of high importance when considering downstream applications such as destination prediction. Misclassification can result in PoIs being missed, in addition to false PoIs being used. Overall, both cases should be considered, however it is more important to remove false PoIs and therefore reduce the noise in the data.

The confusion matrices in Figures 4.3 and 4.4 show a breakdown of the per-activity 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 activities, 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 $\lambda$ values of 5 and 10. Additionally, a high misclassification rate between barrier and drive-through activities can be seen, however this is not unexpected given the similarity of these activities.

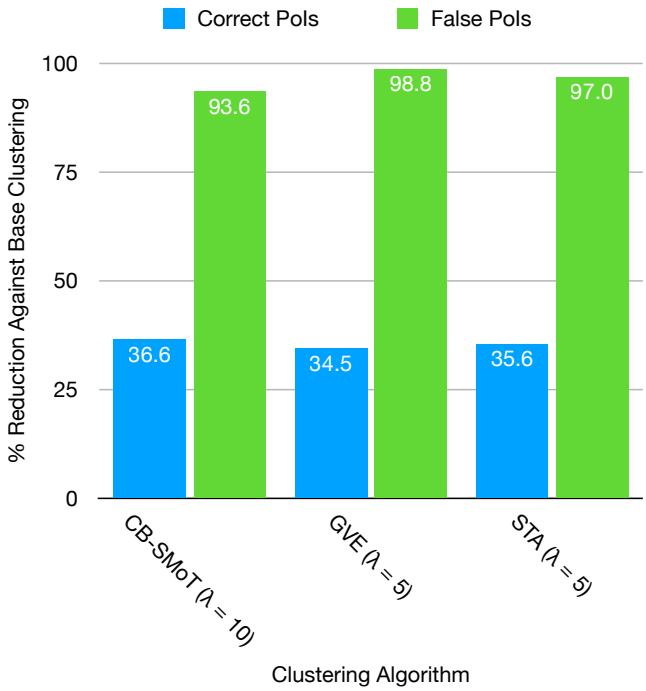
From the confusion matrices, it is also apparent that the Random Forest classifier incorrectly predicts the traffic activity at a much higher frequency than the SVM classifier. Although the AUC values when using the Random Forest classifier are generally higher than when using the SVM classifier, the confusion matrices show that the SVM classifier appears to be better over all activities.

Table 4.8 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.

Table 4.9 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




(a) Random Forest




(b) SVM

Figure 4.5: Reduction in PoIs when using AVPE compared to the base clustering methods.

Table 4.8: Classification results, where $*$ denotes the highest performing parameter combination, in terms of AUC, for each clustering algorithm and classifier combination.

| **Classifier** | **Algorithm** | **Merge threshold $\lambda$ (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 |

Table 4.9: 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 |
| 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 |

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 4.5 shows the improvement that AVPE gives over the three existing state-of-the-art algorithms alone. We define the percentage reduction as,

$$reduction_\alpha = 100 - \frac{\#P^*_\alpha}{\#P_\alpha},$$

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 $\alpha$ 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 4.5a), 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 4.5b shows the same metrics for the SVM classifier. We see similarities to using the Random Forest 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 in 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.

### 4.4.3 Applying AVPE to Pattern of Life Data

In this section, we apply AVPE to the POLD and compare the performance of AVPE on the two different vehicles used. The classifier was trained using the LED, with a SVM as the classification method (since in general it outperformed

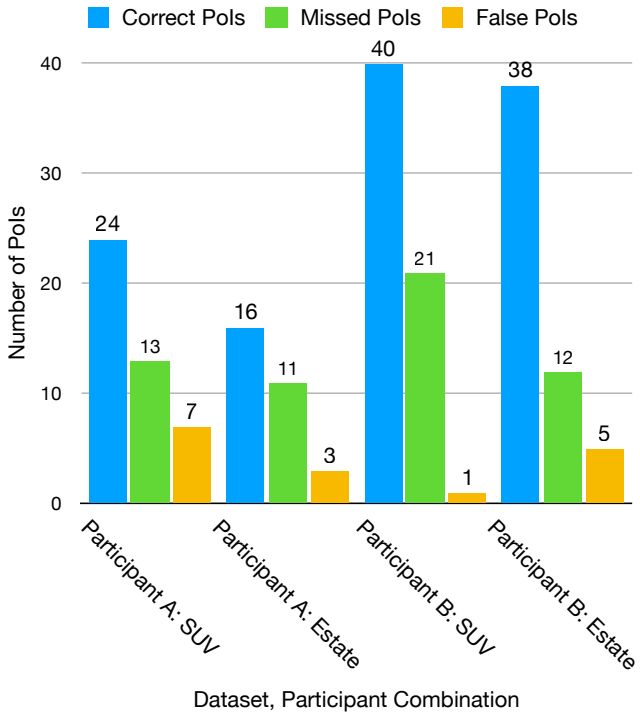
Random Forest on the LED) and mRMR feature selection, as detailed in Section 4.2. The input clusters were generated using GVE with $\lambda = 5$, and both vehicles are used in the training and testing sets.

Participant A had 37 ground truth PoIs that were obtained in the SUV. 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 activities, while the false PoIs were barrier, manoeuvre and traffic activities. When using the estate car, 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 activities. Once again, all missed PoIs were parked activities, of which 3 were due to missing data as a result of poor GPS signal.

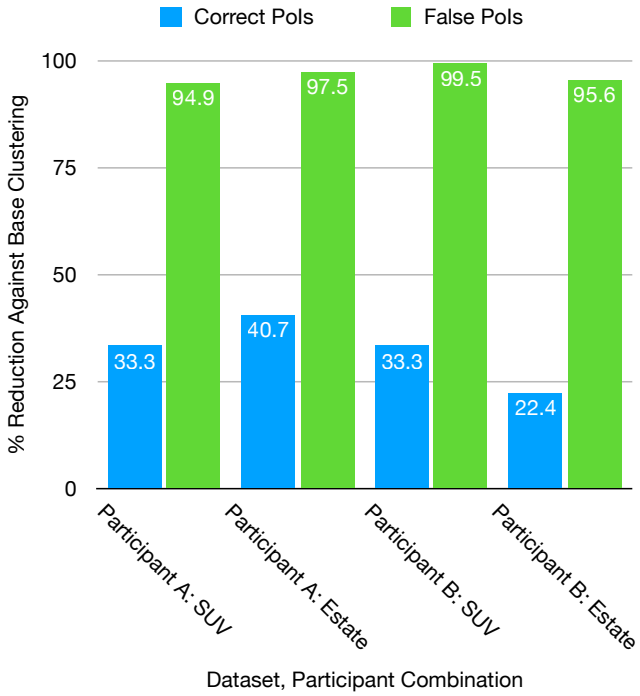
Based on the the SUV data for Participant B, 61 ground truth PoIs were identified. Using AVPE, there were 40 correct PoIs extracted. There was only a single false PoI, but 21 PoIs were missed in the process, 19 of which were parked activities along with a single drop-off and a single pick-up activity. In the case of the estate car, 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 activities with a single drop-off activity.

The results are summarised in Figure 4.6a, 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 activities. Overall, it is clear that AVPE continues to provide a significant reduction in false PoIs, which is the aim of the method. Figure 4.6b illustrates this, with over 94.9% of false PoIs removed in the POLD. 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 LED (see Figure 4.5), 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 activity. 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 activity, and a single cluster covers 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 activity. The length of the manoeuvre activity in the cluster is generally longer than the duration of the parked activity, hence the features (such as averages) are biased towards the qualities of a manoeuvre.




(a) Number of correct, missing and false PoIs output by AVPE (SVM and GVE, $\lambda = 5$).




(b) Reduction in PoIs when using AVPE (SVM and GVE, $\lambda = 5$) compared to the base clustering.

Figure 4.6: Results of the AVPE methodology applied to the pattern of life data, using 21 features.

Other variations include slow driving, drop-off and traffic activities 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.

### 4.4.4 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.

## 4.5 Summary

In this chapter, 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 clear trade-off between reduction in false PoIs against loss of

true PoIs. How to determine a suitable trade-off between these, whilst also dependant on the application, is outside of the scope of the thesis and is left as an open question. When applied on the POLD, AVPE saw comparable performance to that on the LED, 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 assist the development of applications such as destination prediction [41, 54, 79, 120] and identifying ride sharing opportunities [26, 55], that benefit from accurate PoI data.

This chapter proposed classifying the activity of each stay point using on-board sensor data, which has been used to filter activities that are not of interest (i.e., false PoIs). However, it is also important to consider all of the activities and how they make up each trajectory. Therefore, in Chapter 5 we adapt this approach, which considers a trajectory as a sequence of locations, where each location has an activity associated with it, and then evaluate how effective this can be at predicting future destinations.

# Chapter 5

# Extracting Activity-annotated Location Sequences from Trajectories for use in Destination Prediction

In the previous chapter, we proposed Activity-based Vehicle PoI Extraction (AVPE), an algorithm that improves current methods for stay point extraction by predicting the activity of a vehicle using data from on-board sensors. This approach is used to filter out stay points that are not PoIs, such as traffic, for use in applications such as destination prediction. In this chapter, we consider the case where these stay points are not filtered out, and instead use the activity of each stay point to help infer the final destination of the vehicle.

## 5.1 Introduction

Destination prediction is concerned with predicting the intended destination of a journey undertaken by an individual. Several applications can benefit from obtaining accurate predictions for a given journey, such as cooperative traffic management [6], intelligent parking recommendations [52, 78] and vehicle customisation [85, 100].

Location data is a fundamental input into existing destination prediction methods, since transforming raw GPS data into sequences of locations and building models on these sequences has been shown to be able to correctly predict destinations with reasonable success [3, 76, 98, 169]. Some existing methods combine location data with other data sources, such as temporal data and weather data [154]. Our proposed method, AVPE, which is presented in Chapter 4, has shown that vehicle data can be used to predict the activity of a vehicle and subsequently annotate stay points on a vehicular trajectory with activities. Using this approach, a sequence of locations can be transformed into an activity-annotated location sequence, and in this chapter we investigate whether such activity annotations can improve the performance of destination prediction. In this chapter, our approach is benchmarked against the performance of a unannotated sequence of locations, generated by the GVE algorithm


(a) Example of stay points having multiple different locations.

(b) Example of all stay points having a single location.

Figure 5.1: Example of stay points and locations.

[158]. Specifically, in this chapter, we (i) propose annotating locations in vehicle trajectories with activity labels for use in destination prediction models, (ii) evaluate the impact of introducing activity labels on predictive performance, and (iii) compare the performance of using different grouping methods for stay points, namely DBSCAN clustering and a grid-based approach.

This chapter is organised as follows. Section 5.2 presents our proposed algorithm, and in Section 5.3 we outline our experimental methodology. Section 5.4 presents the results of applying our algorithm to predict the destinations of vehicle trajectories in our dataset. Finally, Section 5.5 concludes the chapter.

## 5.2 Methodology

A trajectory, as defined in Section 2.4.1, is a vector of instances, in which an instance, $x_j$, at time $j$ is a tuple $x_j = \langle lat, long, V \rangle$. We define a stay point to be a period of low mobility within a trajectory. A stay point is a strictly ordered subsequence of instances, $[x_1, x_2, ..., x_n]$. A stay point can contain an activity annotation, that describes what the vehicle is doing during the duration of the stay point. Our methodology begins by running each trajectory through an algorithm to extract the stay points, followed by an activity annotation algorithm, such as AVPE (see Chapter 4), which will predict the activity performed at each stay point, to obtain a vector of annotated stay points for each trajectory.

We define a location to be a defined geographic area, within which multiple stay points can fall. A location can contain additional data such as the duration of time elapsed at the location and the activity performed at the location. To transform stay points into locations, we define a function that takes a

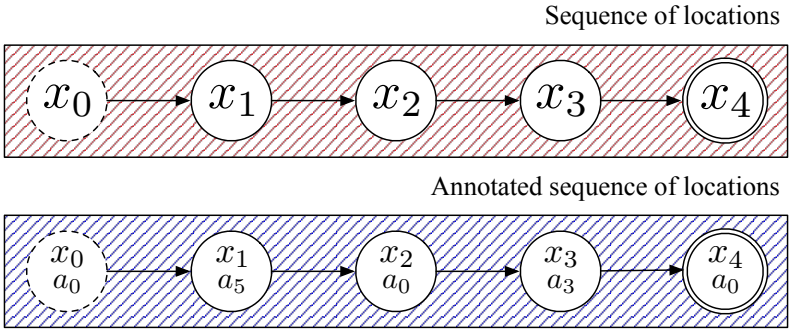



Figure 5.2: Example of a sequence of locations where $x_i$ is a location, $a_i$ is an annotated activity, and $x_0$ and $x_4$ are the source and destination respectively.

grouping method and a set of vectors of stay points, $\mathcal{S}$, and outputs a set of vectors of locations. Figures 5.1a and 5.1b show the same set of stay points and how they can be associated with different locations. Figure 5.1a shows an example of stay points that belong to different locations, and Figure 5.1b shows how those same stay points could instead belong to a single location. The manner in which stay points are grouped into locations depends on the grouping method used, an example of which could be a density-based clustering algorithm such as DBSCAN or an arbitrary mapping such as the OS National Grid reference system (which we refer to as OS Grid). The grouping method used is not defined in the algorithms proposed in this chapter, though can be chosen depending on the application and input data.

In this chapter, we propose the use of activity annotations within a sequence of locations, where each location is annotated with the vehicle's activity, that can provide additional information for that location in the sequence. Figure 5.2 illustrates a sequence of 5 locations, from a source location, $x_0$, to a destination location, $x_4$, that is traversed by a vehicle, both with and without activity data. A partial sequence of locations represents a journey that is in progress, and therefore the sequence will be missing the destination location and potentially multiple intermediate locations.

We will refer to a location that contains an activity and duration as an enriched location. Our method takes a sequence of enriched locations. These enriched locations will be used by our method to train a classifier, with the destination as the class label. Our method assumes that all trajectories have a stay point (i.e., a period of low mobility) at the start and end, so that locations for both the source and destination are present.

### 5.2.1 Pre-processing

In our proposed method, we begin with a pre-processing stage, as defined in Algorithm 5. This begins by taking each trajectory, and applying a function,

Table 5.1: Functions used in this chapter.

| Notation | Description |
|---|---|
| $lookup(s)$ | A function that returns the associated value for stay point $s$ from the given mapping |
| $getStayPointDuration(s)$ | A function that returns the duration of time elapsed for the stay point input |
| $combineAdjacentDuplicates(\Upsilon_t)$ | A function that returns a condensed sequence of enriched locations, with any consecutive duplicate location, activity values are combined, summing the duration together |
| $remapActivities(\Upsilon, m)$ | A function that returns a set of enriched locations, with the activities modified according to the mapping $m$ |
| $getCrossFolds(\mathcal{T}, k)$ | A function that returns an array of training and validation trajectories for a given number of folds, $k$ |
| $initialNPercentOfTragectory(\cdot)$ | A function that results the initial $n$-percent of the enriched locations, $\Upsilon_t$ |
| $haversine(destination(t), \phi)$ | A function that returns the Haversine distance (defined in Equation 5.1) between the predicted destination, $\phi$, and the actual destination |
| $destination(t)$ | A function that returns the labelled destination of trajectory $t$ |
| $areaUnderCurve(avgDistError)$ | A function that returns the area under the curve given a set of values |

that returns a vector of stay points, $S_i$. We define $cluster()$ to be a function that takes a single trajectory, $t_i$, and returns a vector of stay points. CB-SMoT [126], GVE [158] and STA [18] are three representative algorithms that could be used for the function. We define $\mathcal{S}$ to be a set of vectors of stay points. Formally, $\mathcal{S}$ is constructed by applying a stay point extraction function, $cluster()$, to every trajectory of raw data $t_i$ in the set of raw trajectories $\mathcal{T}$, i.e., $\mathcal{S} = \{S_i | S_i = cluster(t_i) \forall t_i \in \mathcal{T}\}$. In the evaluation later in this chapter, we use the Gradient-based Visit Extractor (GVE) algorithm [158] as a representative algorithm that would perform this, since our previous work in Chapter 4 found this algorithm to be effective.

Once the trajectories are transformed from sequences of instances to sequences of stay points, our method applies a function to the set vector of stay points, that transforms the stay points into locations. Formally, $groupLocations()$ takes a set of vectors of stay points, $\mathcal{S}$, and outputs a lookup table that will map each stay point, $s$, to a location, depending on the pre-determined location grouping algorithm used. Examples of the grouping algorithm can be a clustering algorithm such as DBSCAN [56], or a grid-based approach such as OS Grid [124]. This lookup table is stored since it is later used for pre-processing when predicting an unfolding trajectory.

Our method will then iterate through the vector of stay points for each trajectory, and will use the location lookup table to obtain the location that

**Algorithm 5:** *preprocess*$(\cdot)$ — Pre-processing stage.

**inputs :** $\mathcal{T} = \{t_1, ..., t_n\}$, the set of $n$ trajectories,
*cluster*(), the chosen stay point clustering algorithm, with pre-trained parameters
*annotate*(), the chosen activity annotation algorithm, with pre-trained parameters
*groupLocations*(), the chosen location grouping algorithm, with pre-trained parameters

**output :** $\Upsilon$, a set of enriched locations

**for** $t_i \in \mathcal{T}$ **do**
  $S_i = cluster(t_i)$
  $\mathcal{S} = \mathcal{S} \cup S_i$
**end**
`// group stay points to locations, store in lookup table`
$locationMap = groupLocations(\mathcal{S})$
**for** $t_i \in \mathcal{T}$ **do**
  **for** $S_i \in \mathcal{S}$ **do**
    `// sequence of enriched locations`
    `// (location, activity, duration)`
    $\Upsilon_t =$ $[(locationMap.lookup(s), annotate(s), getStayPointDuration(s))$ | for each $s \in S_i]$
    `// combine consecutive duplicates using duration`
    $\Upsilon_t = combineAdjacentDuplicates(\Upsilon_t)$
  **end**
**end**
**return** $\Upsilon$

corresponds to each stay point. Additionally, as we iterate through each stay point, we pass the current stay point into functions that will return the activity predicted for the stay point and its elapsed duration. We use the AVPE algorithm as our chosen activity annotation algorithm. This uses vehicle signals within the trajectory to predict the activity at each stay point.

With this data, each stay point is now represented as a enriched location, which consists of the location, activity and duration associated with the stay point. This is repeated for all stay points within a trajectory, and therefore a trajectory is now represented as a sequence of enriched locations. Within this sequence, the values of location and activity within each enriched location are examined. If any consecutive duplicates (where a duplicate is defined as both the location and activity being identical for different enriched locations) are present, the *combineAdjacentDuplicates*() function will combine these enriched locations into a single enriched location, where the duration is the sum of all durations for the individual enriched locations. An example of the

operation of this function is illustrated in Figure 5.3.

The pre-processing of a set of trajectories is now complete, with each trajectory now transformed into a sequence of enriched locations, consisting of location, activity and duration.

### 5.2.2 Training

The training stage, detailed in Algorithm 6, takes a set of training trajectories, $\mathcal{T}_{train}$, a stay point clustering algorithm, $cluster()$, an activity annotation algorithm, $annotate()$, an algorithm for grouping locations, $groupLocations()$, a classification method, $classificationMethod()$, and a set of activity mappings, $\mathcal{A}$.

The activity mappings provide a map from the original activity to a newly created activity. For example, if you have drop-off and pick-up activities, you may wish to combine these into a single activity, therefore the individual drop-off and pick-up activities would both be mapped to this composite activity. Thus, while our method takes all possible activities given by the annotation algorithm, this approach gives the opportunity to re-map these to examine different combinations and select the best performing one for the input data used. For each mapping provided to the training algorithm from the set of activity mappings, $\mathcal{A}$, the activities are remapped using current mapping. An example of three activity mappings ($a_1$, $a_2$, and $a_3$) are shown in Figure 5.4.

The training algorithm begins by using Algorithm 5 to pre-process the trajectories, outputting the enriched locations as detailed in Section 5.2.1. The training trajectories are then split using stratified k-fold cross validation, and a classifier, $\psi$, is trained on the completed sequences of enriched locations consisting of location, activity and duration (i.e. using the trajectories at 100% completion), using the chosen classification method. A single classifier is

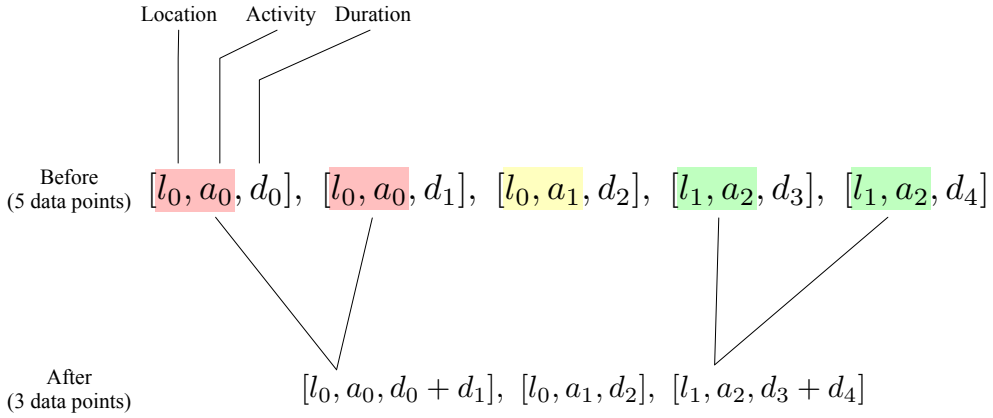



Figure 5.3: Example of combining consecutive duplicates, where $l_i$ is the location, $a_i$ is the annotated activity and $d_i$ is the duration of the stay point.

```
BARRIER         ->   BARRIER
DRIVETHROUGH    ->   DRIVETHROUGH
DRIVING         ->   DRIVING
DROPOFF         ->   DROPOFF
MANOEUVRE       ->   MANOEUVRE
PARKED          ->   PARKED
PICKUP          ->   PICKUP
TRAFFIC         ->   TRAFFIC
```

Example mapping (a)

```
BARRIER         ->   BARRIER
DRIVETHROUGH    ->   DRIVETHROUGH
DRIVING         ->   {DRIVING, PARKED}
DROPOFF         ->   {DROPOFF, PICKUP}
MANOEUVRE       ->   MANOEUVRE
PARKED          ->   {DRIVING, PARKED}
PICKUP          ->   {DROPOFF, PICKUP}
TRAFFIC         ->   TRAFFIC
```

Example mapping (b)

```
BARRIER         ->   BARRIER
DRIVETHROUGH    ->   DRIVETHROUGH
DRIVING         ->   {DRIVING, PARKED, TRAFFIC}
DROPOFF         ->   {DROPOFF, MANOEUVRE, PICKUP}
MANOEUVRE       ->   {DROPOFF, MANOEUVRE, PICKUP}
PARKED          ->   {DRIVING, PARKED, TRAFFIC}
PICKUP          ->   {DROPOFF, MANOEUVRE, PICKUP}
TRAFFIC         ->   {DRIVING, PARKED, TRAFFIC}
```

Example mapping (c)

Figure 5.4: Example of different activity mappings within the set of activity mappings.

**Algorithm 6:** $train(\cdot)$ — Training a destination prediction model with annotated vehicle trajectories.

**inputs :** $\mathcal{T}_{train} = \{t_1, ..., t_n\}$, the set of $n$ training trajectories
$cluster()$, the chosen stay point clustering algorithm
$annotate()$, the chosen activity annotation algorithm
$groupLocations()$, the chosen location grouping algorithm
$classificationMethod$, the chosen classification method
$\mathcal{A} = \{a_1, \ldots, a_r\}$, a set of activity mappings
$k$, the number of folds to use
$inc$, the incremental granularity for the percentage of the trajectory

**output :** $\psi^*$, a trained classifier
$\omega^*$, the activity mapping used in the classifier $\psi^*$

$\psi^*, \omega^* = null$
$AUC = -\infty$
`// pre-process the training trajectories`
$\Upsilon = preprocess(\mathcal{T}_{train}, cluster(), annotate(), groupLocations())$
**for** $a \in \mathcal{A}$ **do**
  $\Upsilon^* = remapActivities(\Upsilon, a)$
  **for** $(Train, Test) \in getCrossFolds(\mathcal{T}, k)$ **do**
    $\psi = trainClassifer(classificationMethod, \{\Upsilon_t \mid t \in Train\})$
    **for** $t \in Test$ **do**
      **for** $percComp \in range(0, 100, inc)$ **do**
        `// predict destination from partial sequence of locations`
        $\phi = predict(initialNPercentOfTragectory(\Upsilon_t, N = percComp), \psi)$
        `// calculate distance from predicted destination`
        $distError[t][percComp] = haversine(destination(t), \phi)$
      **end**
    **end**
  **end**
  **for** $percComp \in range(0, 100, inc)$ **do**
    $avgDistError[percComp] = avg_{t \in \mathcal{T}}(distError[t][percComp])$
  **end**
  **if** $areaUnderCurve(avgDistError) > AUC$ **then**
    $\psi^* = trainClassifer(classificationMethod, \{\Upsilon_t \mid t \in \mathcal{T}\})$
    $\omega^* = a$
    $AUC = areaUnderCurve(avgDistError)$
  **end**
**end**
**return** $\psi^*, \omega^*$

output, as our method proposed uses multi-class classification, in which each destination is a class label. For each processed trajectory in the validation set, and in increments of $inc$%, the trajectory is trimmed (to only contain the first $percComp$% of the journey) and input into the trained classifier to predict the destination, $\phi$. The distance error is calculated as the Haversine distance between the prediction $\phi$ and the actual destination, $destination(\upsilon)$. The Haversine distance is the spherical distance between two latitude, longitude coordinates on the surface of a sphere [81], defined as,

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\psi_2 - \psi_1}{2}\right) + \cos\psi_1 \cdot \cos\psi_2 \cdot \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right), \quad (5.1)$$

where $r$ is the radius of the sphere, $\psi_1, \psi_2$ are the latitude of point 1 and point 2, and $\lambda_1, \lambda_2$ are the longitude of point 1 and point 2. The area under the

curve (AUC) is calculated using values for distance error, and should this be the lowest AUC seen thus far in training, the classifier and respective activity mapping are stored. This is repeated for every mapping in the set of activity mappings, $\mathcal{A}$, and the classifier, $\psi^*$ and activity mapping, $\omega^*$ with the lowest AUC are returned by the training algorithm.

### 5.2.3 Prediction

The prediction stage is detailed in Algorithm 7, and takes an unfolding trajectory, $t$, a stay point clustering algorithm, $cluster()$, an activity annotation algorithm, $annotate()$, an algorithm for grouping locations, $groupLocations()$, the pre-trained classifier, $\psi$, and the activity mapping, $\omega$, used in the classifier. The pre-processing stage is almost identical to the training stage, with the exception of the algorithm for grouping locations, $groupLocations()$. This algorithm needs to match the locations to those defined in the training set. For algorithms such as OS Grid [124], this is trivial, since the mapping is uniform, and we can discard previously unseen locations. However, when DBSCAN [56] is used, a more complex approach is required, such as finding the nearest location to each stay point. Given that our method is agnostic to the location grouping algorithm, our method will allow for both of the above, finding the nearest location to each stay point, or, in the case of a reference system such as OS Grid mapping, either matching or discarding the locations. Once the unfolding trajectory has been processed, the activities are remapped using the specified activity mapping, $\omega$, and the pre-trained classifier, $\psi$, is used to output a predicted destination, $\phi$.

## 5.3 Experimental Methodology

To demonstrate and evaluate our method, we have selected a set of activity labels, grouping methods and classifiers to be used. In this section, we discuss the experimental parameters and grouping methods used in our implementation of the algorithms proposed in this chapter. We discuss the pre-processing carried out on the POLD for our evaluation, and its resulting properties.

### 5.3.1 Experimental Parameters

In this chapter, we use a 70-30 split for the training and test data, since given the small size of our dataset, we opt for a larger proportion to be used for training. Within the training data, to evaluate the model, we use $k = 5$ for the stratified k-folds.

The stay point extraction method used is the Gradient-based Visit Extractor (GVE) [158] along with AVPE, as defined in the previous chapter, for the

**Algorithm 7:** *predict*$(\cdot)$ — Predicting the destination of a unseen trajectory.

**inputs :** $t$, an unfolding trajectory
*cluster*(), the chosen stay point clustering algorithm, with pre-trained parameters
*annotate*(), the chosen activity annotation algorithm, with pre-trained parameters
$\psi$, the pre-trained classifier
$\omega$, the activity mapping used in the classifier $\psi$
**output :** $\phi$, the predicted destination of trajectory $t$

```
// pre-process the trajectory
// here closestLocation() returns the closest location from
   the set of locations found in the training phase
```
$\Upsilon = preprocess(t, cluster(), annotate(), closestLocation())$

```
// convert the activities using the mapping
```
$\Upsilon^* = remapActivities(\Upsilon, \omega)$

```
// predict destination from partial sequence of locations
```
$\phi = predict(\Upsilon^*_t, \psi)$
**return** $\phi$

activity annotation algorithm. When investigating the location grouping algorithm, we consider both DBSCAN [56] and OS Grid [124].

Our method is agnostic to the activities used, however we use the activities defined in Chapter 3 for our evaluation. These activities are barrier, drive through, driving, drop-off, manoeuvre, parked, pick-up and traffic.

Each classifier is trained using the locations observed over the complete trajectory (i.e., when at 100% trajectory completion). We evaluate the performance of the classifier by examining the distance error, for which we use the Haversine distance [81] between the predicted and actual destination location, as defined in Equation 5.1. The distance error is calculated at 5% increments of trajectory completion, and is averaged over all trajectories in the validation set.

In this chapter, we compare a decision tree model (using a bag-of-words approach, where each distinct enriched location is a word) with that of a Hidden Markov Model (HMM) [134], both with and without activity annotations included in the enriched locations. We have chosen these approaches as both decision tree learning [42, 44, 121] and HMMs [16, 45, 73] have been previously used for destination prediction. However, our method is agnostic to the classification method used.

Our method takes in an activity mapping as input, which enables the activity labels to be aggregated into groups. For example, activities such as drop-off and pick-up can be combined into a single composite activity. We

| Mapping | Composite Activities |
|---|---|
| A.1 | None |
| B.1 | {Driving, Traffic} |
| B.2 | {Driving, Manoeuvre, Traffic} |
| C.1 | {Drop-off, Pick-up} |
| C.2 | {Drop-off, Parked, Pick-up} |
| D.1 | {Driving, Manoeuvre, Traffic}<br>{Drop-off, Parked, Pick-up} |
| E.1 | {Barrier, Driving, Manoeuvre, Traffic}<br>{Drive through, Drop-off, Parked, Pick-up} |

Table 5.2: Activity mappings considered in our evaluation.

explore both multi-class and binary annotations within the enriched locations. In our evaluation, we consider 7 activity mappings, as shown in Table 5.2. The table details the activities that are combined into composite activities. For example, the B.2 mapping combines the driving, manoeuvre and traffic activities into a single composite activity, and the E.1 mapping combines 2 groups of 4 activities into 2 composite activities. The A.1 mapping does not create any composite activities, and is equivalent to using the original 8 activities individually.

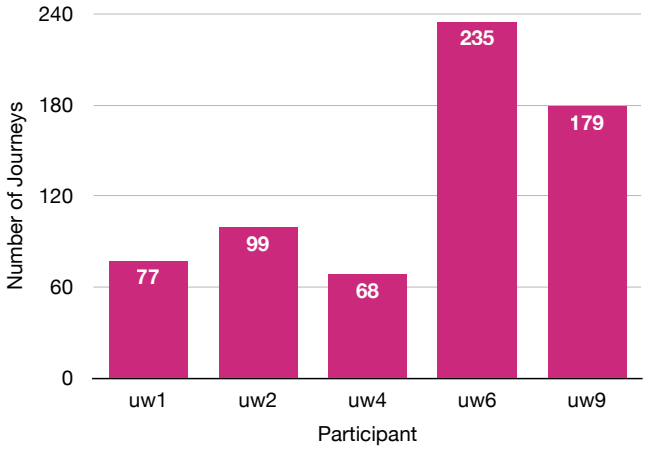
### 5.3.2 Dataset

For our evaluation, we use the POLD, as introduced in Chapter 3, that contains 5 participants. Prior to our evaluation, we pre-process the trajectories in the dataset. We assume that the set of trajectories input into the training and prediction algorithms contain no trajectories in which the destination is only visited once. Any of these trajectories that are found in the dataset are removed, and therefore is a subset of the POLD (for brevity, we refer to this subset as the POLD).

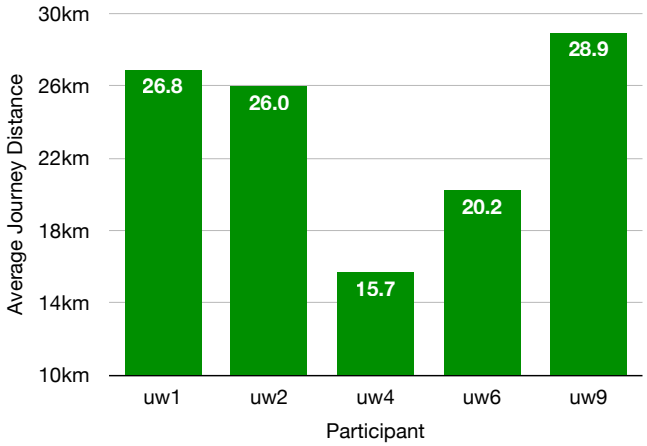
Illustrated in Figure 5.5a, each participant made between 68–235 journeys during the period of data collection, after pre-processing. Journey lengths varied from under a kilometre to over 300 kilometres, but on average were in the range of 20–30 kilometres, with the exception of one participant. The average journey lengths for each participant are shown in Figure 5.5b.

### 5.3.3 Missing Stay Points

In our dataset, some trajectories were missing a stay point at the start of the journey. This is due to a delay in getting a GPS location from the power on state, and the vehicle already moving from the start location before a GPS fix can be obtained. To resolve this issue, we added a stay point to trajectories




(a) Total number of journeys per participant.




(b) Average journey distance per participant.

Figure 5.5: Statistics from the POLD.

where this was missing. This synthetic stay point was one second in duration, and its latitude and longitude were approximated from both the experiment logbook and manual inspection of the previous location of the vehicle. In a real-world situation, the source can be inferred from the destination of the previous journey, which mitigates this issue in practice.

### 5.3.4 Grouping Method

Nearby stay points are grouped together into a location using either DBSCAN [56], or the OS grid mapping [124]. As stay points consist of multiple instances, where each instance can be a different set of coordinates, we represent the coordinates of a stay point as the mean of the longitude and latitude of all instances. DBSCAN is a density-based clustering algorithm [56] that uses two parameters to determine the neighbourhood of an instance, and the minimum number of instances required to form a cluster. Unlike other clustering algorithms DBSCAN does not require the number of clusters to be specified in advance and is therefore suited to the application of grouping stay points into locations. For DBSCAN, we provide an epsilon value, $\epsilon$, which defines the distance threshold in metres. Altering this value will affect the number of locations determined from stay points. For example, a higher value of $\epsilon$ may combine points from multiple sides of a roundabout into a single location. The DBSCAN algorithm uses a threshold (epsilon) to specify how close points should be to each other to be considered part of the same cluster. While the Euclidean distance is often used with DBSCAN, since we are considering geospatial data, we use the Haversine distance between points, as defined in Equation 5.1.

As an alternative to DBSCAN clustering, we consider the use of OS Grid mapping to group the stay points into locations. OS Grid is a grid reference system used in Great Britain [124]. Using 2 grid-letters, the country is divided into 25 100km by 100km grid squares, with further pairs of digits used to vary the resolution of the grid square. In our approach, the coordinates of each stay point are converted into a grid reference, to determine its respective grid square.

Both approaches present their own challenges. Using the OS Grid mapping approach provides a consistent location area, however stay points whose coordinates reside on the boundary of multiple grid squares could be assigned a different grid square given even the smallest of changes in coordinates. The grid boundaries are arbitrary and therefore the locations are not tailored to the input data, unlike in DBSCAN.

Using DBSCAN provides flexibility in the shape and size of the location, however DBSCAN requires a specified distance threshold which is not easy

to estimate for large sets of trajectory data. This is a similar problem faced when using OS Grid mapping, as determining the grid size to use is not obvious. There is a trade-off between the threshold or grid size used, which could also cause a suboptimal grouping of locations. Choosing a grid size which is too large for OS Grid mapping or a threshold that is too small for DBSCAN may group too many locations together, and reduce the precision of the system, but may improve the overall classification performance (for example, any misclassifications between two nearby locations would be avoided as they are now considered a single location, however this will artificially inflate the performance in the case that the two locations are really different). Choosing a suitable threshold or grid size is a non-trivial task, and for the purpose of this evaluation we use 100 metre grid squares, and a threshold of 100 metres (the choice of 100 metres was due to this being the nearest OS Grid reference size to a car park).

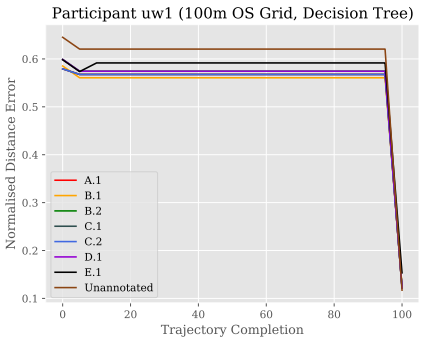
## 5.4 Results

As described in Section 5.3, we consider DBSCAN and OS Grid mapping as methods for obtaining the locations from visits output by AVPE. In addition to the method used to gather locations, our results compare the performance of a simple decision tree classifier to a more sophisticated HMM. The performance of unannotated and annotated vehicle trajectories are compared, highlighting the benefits and drawbacks of the activity annotations. In the ideal case, the plots in this section would illustrate a steady downward trend, where the prediction was perfect at 100% completion. Normalised distance error is used for all the results, to allow for a consistent comparison between both journeys and participants. The normalised distance error for a trajectory is calculated as the distance from the predicted to the actual destination locations, divided by the total distance travelled in the trajectory. This will ensure that the same distance error is more significant on shorter trajectories.

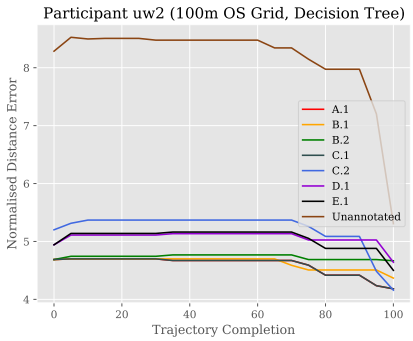
The first set of results is the validation set from the training phase, where we examine the distance error to select the activity mapping that provides the best performance. In this section, we refer to the mappings as defined in Table 5.2.

### 5.4.1 OS Grid mapping

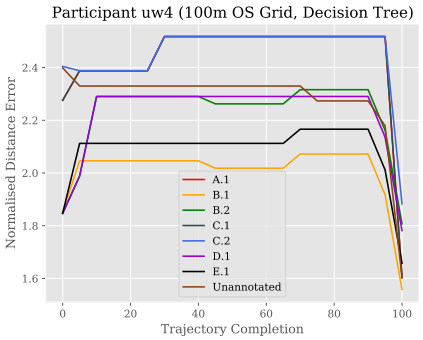
First, we consider the performance for the OS Grid mapping method. It can be seen from Figures 5.6 and 5.7 that the models produced from the decision tree classifier fluctuate in performance less so than those using the HMM, having mostly a downward trend as the journey unfolds. The distance errors



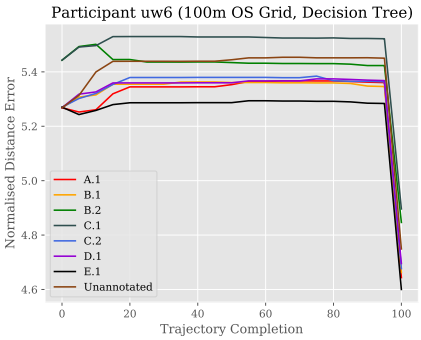

(a) Participant uw1



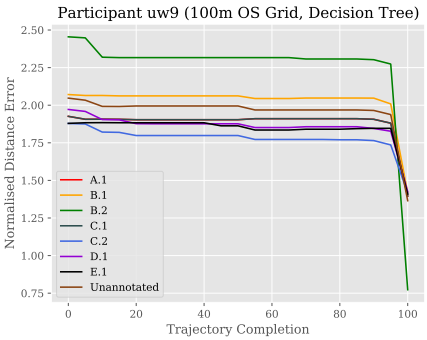

(b) Participant uw2




(c) Participant uw4

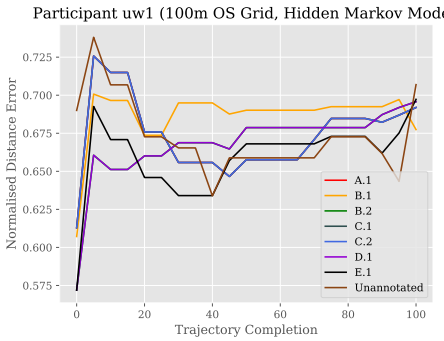



(d) Participant uw6

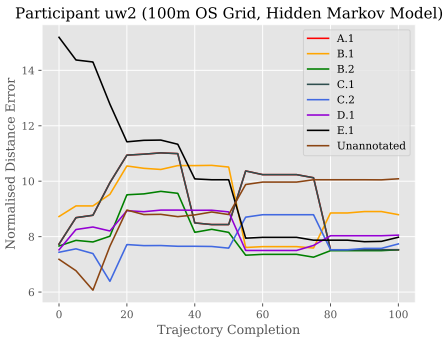



(e) Participant uw9

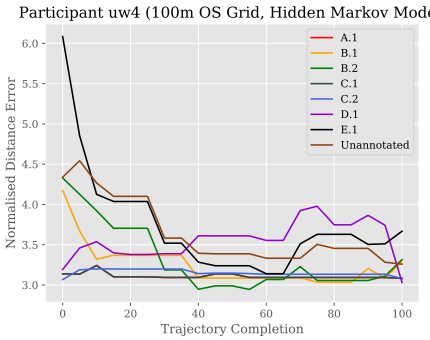
Figure 5.6: Distance error when using $100m$ OS Grid squares and a decision tree classifier, for all participants.



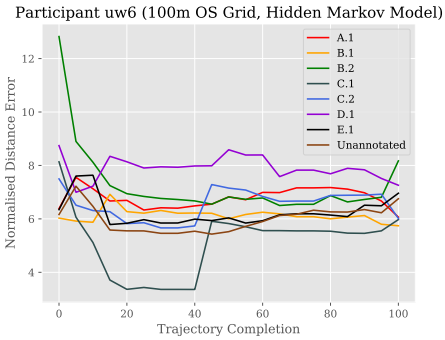

(a) Participant uw1



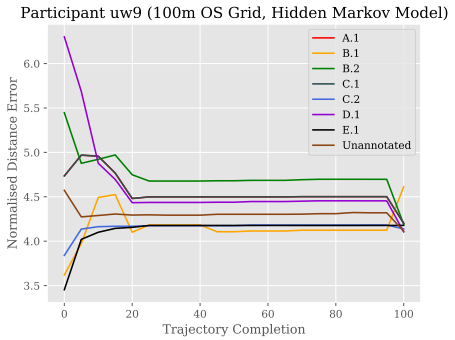

(b) Participant uw2




(c) Participant uw4




(d) Participant uw6




(e) Participant uw9

Figure 5.7: Distance error when using $100m$ OS Grid squares and a HMM classifier, for all participants.

| Participant | Classifier | Mapping(s) | Average Dist. Error |
|---|---|---|---|
| uw1 | Decision tree | B.1 | 0.541 |
| | HMM | E.1 | 0.660 |
| uw2 | Decision tree | A.1, C.1 | 4.596 |
| | HMM | C.2 | 7.818 |
| uw4 | Decision tree | B.1 | 2.007 |
| | HMM | A.1, C.1 | 3.111 |
| uw6 | Decision tree | E.1 | 5.251 |
| | HMM | C.1 | 5.122 |
| uw9 | Decision tree | C.2 | 1.776 |
| | HMM | E.1 | 4.129 |

Table 5.3: Best performing mappings when using OS Grid mapping on the validation set (refer to Table 5.2 for the mappings).

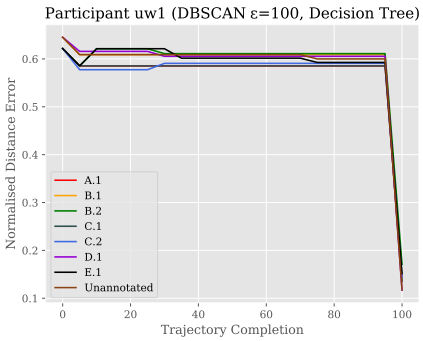
produced from the HMMs vary as the journey unfolds, where it seems that in some cases more data results in a worse prediction from the model. When we examine the different participants, we notice no obvious pattern between them, where some show that using the full set of annotations give improved performance (such as Figure 5.6a) and in other cases binary annotation gives better performance (such as Figure 5.7a). These results demonstrate that the choice of classification model used can greatly affect the performance, for example in the case of uw2 (see Figures 5.6b and 5.7b). Using the decision tree classifier produces a smaller distance error for all participants, compared to the HMM.

If we consider the different activity mappings, we observe that the ranking of each mapping changes frequently when using the HMM, and is generally more consistent for the decision tree classifier, with most of the changes in the final 10% of the trajectory.

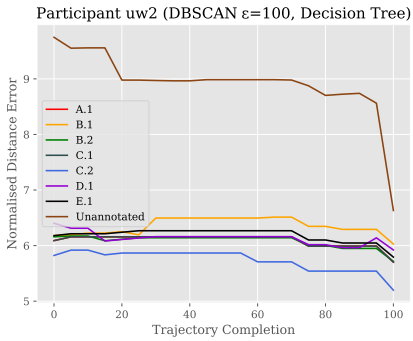
To examine the mappings that perform the best, we consider the average distance error, as shown in Table 5.3. For all participants, a mapping with activity labelling outperforms the original that does not have activity labelling. Out of the 7 mappings with activities, we can see a mixture that provide the best performance, with the E.1 and C.1 mappings being the most common. Conversely, the B.2 and D.1 mappings are never the best performing.

### 5.4.2 DBSCAN

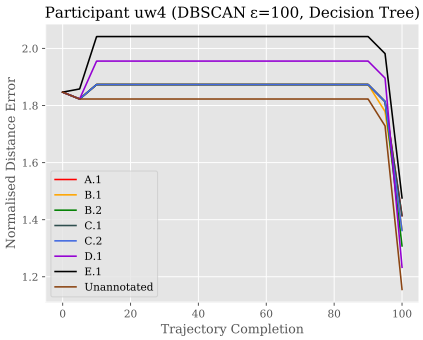
When considering the performance with DBSCAN used as the grouping method, we see that the general trends exhibited have the same properties as those observed when using OS Grid mapping, however there are some notable



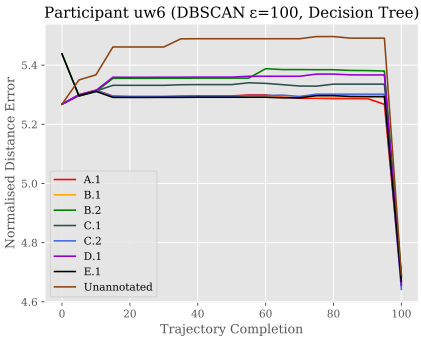

(a) Participant uw1



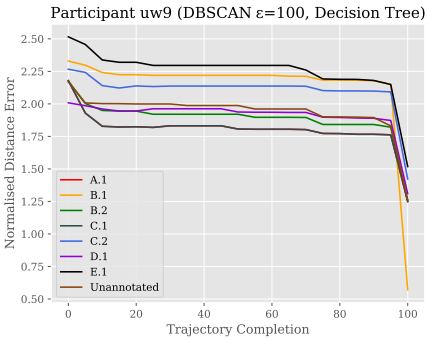

(b) Participant uw2




(c) Participant uw4

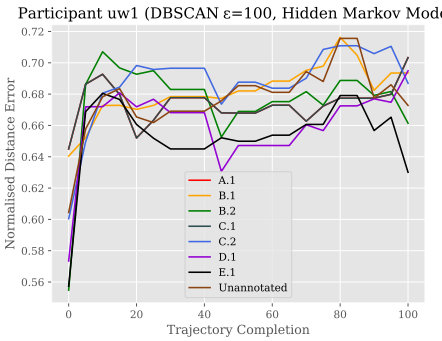



(d) Participant uw6

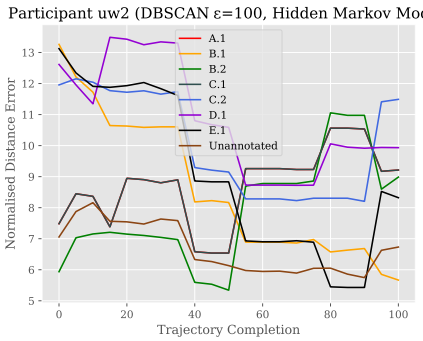



(e) Participant uw9

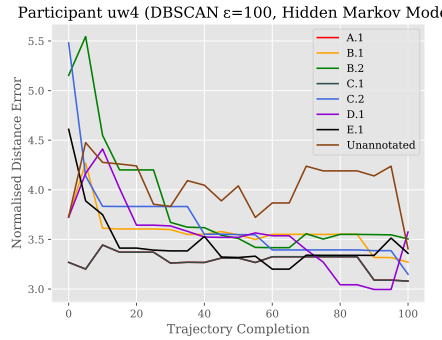
Figure 5.8: Distance error when using DBSCAN with a threshold of 100m and a decision tree classifier, for all participants.



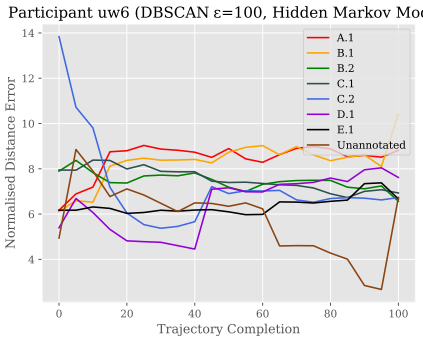

(a) Participant uw1



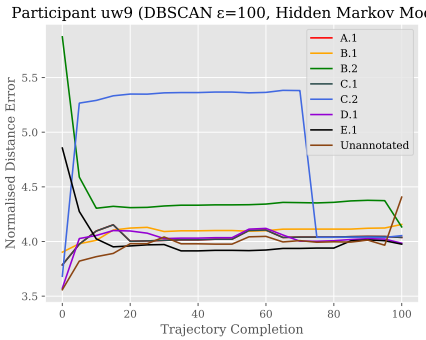

(b) Participant uw2




(c) Participant uw4




(d) Participant uw6




(e) Participant uw9

Figure 5.9: Distance error when using DBSCAN with a threshold of 100m and a HMM classifier, for all participants.

| Participant | Classifier | Mapping(s) | Average Dist. Error |
|---|---|---|---|
| uw1 | Decision tree | A.1, C.1 | 0.566 |
| | HMM | E.1 | 0.653 |
| uw2 | Decision tree | C.2 | 5.736 |
| | HMM | Unannotated | 6.687 |
| uw4 | Decision tree | Unannotated | 1.788 |
| | HMM | A.1, C.1 | 3.282 |
| uw6 | Decision tree | A.1 | 5.262 |
| | HMM | Unannotated | 5.780 |
| uw9 | Decision tree | A.1, C.1 | 1.802 |
| | HMM | Unannotated | 3.976 |

Table 5.4: Best performing mappings when using DBSCAN on the validation set (refer to Table 5.2 for the mappings).

differences.

For example, when using a decision tree classifier and DBSCAN as the grouping method, we observe degraded performance for participant uw2 (see Figure 5.8b) compared to when using OS Grid mapping (see Figure 5.6b). However, there is a larger difference in performance between different activity mappings, with the C.2 mapping providing the best performance, by at least 0.239 over any other mapping.

Conversely, when using a decision tree classifier for participant uw4, the DBSCAN grouping method (see Figure 5.8c) gives better performance. However, the unannotated variant outperforms all other mappings, with the performance only matched in the initial 5% of the trajectory.

Table 5.4 details the best performing mappings when using DBSCAN grouping on the validation set. Unlike OS Grid mapping, these results show that the unannotated variant sometimes outperform the annotated sequences for a given participant/classifier combination. Similar to when using OS Grid mapping, B.2 and D.2 mappings are never the best performing, along with the B.1 mapping. We observe that the A.1 and C.1 mappings are the most common, underneath the unannotated variant.

### 5.4.3 Applying the models to unseen data

After comparing the performance of the mappings on the validation set, we now apply the best performing mappings to our unseen test data, along with the unannotated variant for comparison. In the instances where there were multiple best performing mappings, we will apply all of them to the test set, and where the unannotated variant was best, it will be compared to the best

| Participant | Grouping Method | Classifier | Mappings |
|---|---|---|---|
| uw1 | OS Grid mapping | Decision tree | B.1 |
| uw2 | OS Grid mapping | Decision tree | A.1, C.1 |
| uw4 | DBSCAN | Decision tree | B.2* |
| uw6 | OS Grid mapping | HMM | C.1 |
| uw9 | OS Grid mapping | Decision tree | C.2 |

Table 5.5: Best performing mappings (refer to Table 5.2 for the mappings) on the validation set, where * denotes that the unannotated variant was the best.

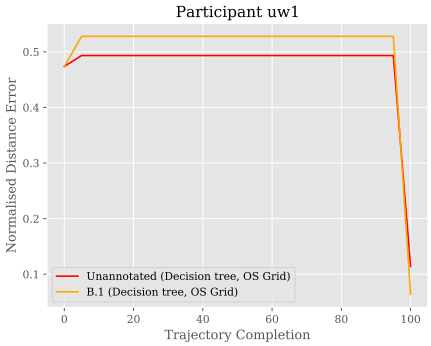
annotated mapping. Given the above, we will now compare the performance on the test set for each participant, using the highest performing combinations shown in Table 5.5.

As shown in Figure 5.10, when applied to the test set, the unannotated sequence of locations results in a smaller normalised distance error for 4 out of the 5 participants. There are a few exceptions to this, specifically the final part of the trajectories for participant uw1, the entire length of trajectories for uw2, and the start of the trajectories for uw4 and uw9. In the case of uw6, no activity mapping outperforms the unannotated variant at any point. For participant uw2, there were 2 identical best performing combinations from the validation stage, and this is mirrored when applied to the test set, where the A.1 and C.1 mappings give the same distance error.

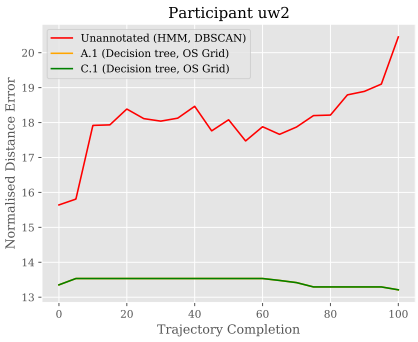
In terms of the difference between the distance error for unannotated and annotated sequences, we see a wide variation, depending on the participant. For participant uw1 (see Figure 5.10a), the difference is small, with the distance error for the unannotated sequence being on average 0.03 smaller. For participants uw4 and uw9 (Figures 5.10c and 5.10e respectively), the difference is on average under 1.0. However, for participant uw6 (Figure 5.10d), we see a huge difference of over 31 at the beginning of the journey, that gradually decreases to the just over 3. Finally, for participant uw2 (Figure 5.10b), in which the annotated sequence outperforms the unannotated sequence, we see a difference in the average distance error that starts at 2.29 and gradually increases throughout to 7.24.

### 5.4.4 Discussion

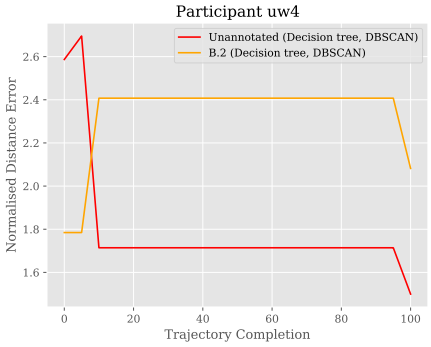
These results show that despite an activity mapping outperforming the unannotated variant on that validation sets (for participants uw1, uw6, and uw9), the performance is not reflected when the models are applied on the unseen test data. This indicates that the models built on the training data do not transfer well to unseen trajectories for the same participant. There could be a number of reasons for this, the most likely being that there are a lack of



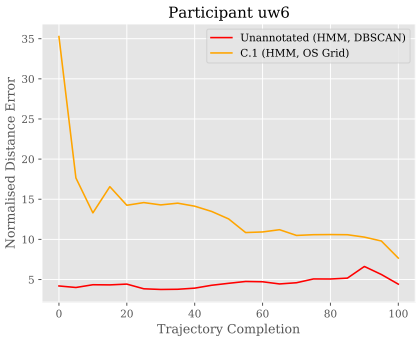

(a) Participant uw1



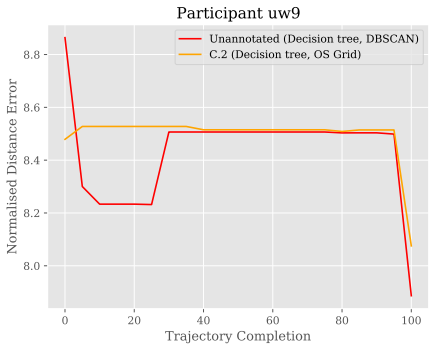

(b) Participant uw2




(c) Participant uw4




(d) Participant uw6




(e) Participant uw9

Figure 5.10: Distance error when applying the highest performing combinations to the test set, for all participants.

common locations between trajectories, given the small size of our dataset. There is a large difference between the performance of participants uw2 and uw6, in which participant uw6 has a significantly higher number of unique destinations than uw2.

The data for each participant exhibits different properties, where the purpose of each journey, and the commonality of the type of journey were varied. The journeys from participant uw1 were more consistent, with a low number of unique destinations, and several trips to the majority of those destinations. In contrast, participants uw6 and uw9 presented with a much higher number of unique destinations but the frequency of some destinations was low, with a significant number of destinations only having a few occurrences in the data. In several participants, the most common journey was between their home and work locations, however it was not uncommon for this to be split by an intermediate location, such as going to a shop or a petrol station. The type of activities varied per participant, with some (such as participant uw1) not performing any of a specific activity e.g. a drive through. Further statistics are summarised in Section 3.2

When we consider the best performing activity mappings used on the test data, we observe that each participant has a different activity mapping to another (with the exception of participant uw2). This indicated that different participants benefit from training a model on different activity mappings. Given this observation, the activity mapping used in each classifier should be individualised to the participant.

Finally, our evaluation demonstrated that the best performance obtained for each participant was on a variety of different classifier and grouping method combinations. However, the most common combination was using a decision tree classifier with OS Grid mapping, making it difficult to have a one-size-fits-all approach.

## 5.5 Summary

In this chapter, we have proposed a methodology for representing a journey as a sequence of activity-annotated locations, and investigated the impact of several different activity mappings. We compared the performance using two location grouping methods, namely DBSCAN [56] and OS Grid mapping [124], in addition to two classification algorithms, namely a decision tree classifier and a HMM [134]. We found that when using the decision tree classifier the distance error from the ground truth throughout the unfolding trajectory was significantly more stable than when using a HMM, which exhibited large fluctuations in performance where the distance error would repeatedly decrease and increase.

For the pattern of life data that we collected, we observed mixed results. For one participant, we saw a significant improvement over the performance of the unannotated sequences. However, for the remaining participants, the unannotated sequences outperformed the annotated ones, in which most were near equal in performance, with a single participant being greatly disadvantaged by the addition of annotations. From our investigation, we can see that there is a great discrepancy between participants and their data, where the distribution of activities can have an impact on performance.

# Chapter 6

# Destination Prediction by Trajectory Sub-clustering

In Chapters 4 and 5, we focused on periods of low mobility within a trajectory, specifically to predict the activities within a trajectory and the final destination of a trajectory. In this chapter, we consider the trajectory as a whole, in contrast to the individual periods of low mobility, comparing the similarity of an unfolding trajectory to previous trajectories. Our overall aim of the proposed method in this chapter is to predict the final destination of a vehicle trajectory. Existing destination prediction approaches make use of the spatial information contained within trajectories, but this can be insufficient to achieve an accurate prediction at the start of an unfolding trajectory, since several destinations may share a common starting portion. To reduce the prediction error in the early stages of a given journey, we propose the Destination Prediction by Trajectory Sub-clustering method (DPTS) for iteratively clustering similar trajectories into groups using additional information contained within trajectories, such as temporal data. We show in our evaluation that DPTS is able to reduce the mean distance error in the first 40-60% of journeys. The implication of reducing the distance error early in a journey is that location-aware applications could provide more accurate functionality earlier in a journey.

## 6.1 Introduction

Intelligent transportation systems can assist drivers in making their daily journeys, by performing tasks such as assessing the current traffic levels [21, 178] and minimising congestion [13, 35, 142], both of which could benefit from an accurate prediction of the destination in advance. Besse *et al.* proposed a method for destination prediction [23], which we refer to as BDP (denoting Besse *et al.*'s Destination Prediction method). Their method uses trajectory similarity classification, a technique that tries to predict which group of trajectories an unfolding trajectory is most likely to match. To calculate the trajectory groupings, Besse *et al.* use hierarchical agglomerative clustering with the Symmetrized Segment-Path-Distance (SSPD), an instance-based trajectory

distance metric [22]. Their method uses either a simple unweighted score based on the GMM likelihood, or a weighted score that uses auxiliary variables (such as the hour-of-day and the day-of-week) and weighting functions to modify the score of each cluster. In this chapter, we opt for using the unweighted BDP method as a baseline, so that the GMM likelihood score can be used directly without needing to define a weighting function for each auxiliary variable. The unweighted BDP method suffers from poor performance at the start of a journey, where limited spatial information is available. When only a small proportion of a journey has been completed, there is only spatial information for the completed section, and since multiple journeys may originate at a single location and share an initial route, this makes it difficult to distinguish the destination early on. Other destination prediction methods exist in the literature, but either use external information from outside the vehicle, such as ground cover data or road type information, to improve predictive performance [99, 187], require knowledge of the identity of the driver [15, 149], or use a complex representation of the road network [15, 127, 149]. In this chapter, we assume that such information is not available and that the identity of the driver is unknown.

In this chapter, we (i) propose the Destination Prediction by Trajectory Sub-clustering (DPTS) method, which extends BDP [23], by using additional data and an iterative sub-clustering approach to combine trajectory clusters into more specific groupings, and (ii) we evaluate DPTS against the baseline performance of BDP (with the unweighted score). A key difference of DPTS from BDP is the use of iterative sub-clustering that can take multiple metrics and their respective parameters, performing iterations of clustering. In contrast, BDP uses a single iteration of clustering with the SSPD metric alone. DPTS can be easily extended by adding additional metrics and iterations to the clustering process, and by varying the order of iterations. This chapter is organised as follows. Section 6.2 presents the DPTS method, and Section 6.3 introduces our experimental methodology and the datasets used for evaluation. Section 6.4 presents the results of applying DPTS to vehicle trajectories, and compares the performance to that achieved by the baseline unweighted BDP. Finally, Section 6.5 concludes the chapter.

## 6.2 Destination Prediction by Trajectory Sub-Clustering (DPTS)

The motivation behind Destination Prediction by Trajectory Sub-clustering (DPTS) is to reduce the distance error in destination prediction using vehicle data, specifically when making predictions in the early stages of a journey. We define the distance error as the Haversine (or spherical) distance [81] between the actual and predicted destination. Reducing the distance error

improves confidence in the correctness of the predicted destination, which can in turn improve location-aware applications, such as recommendations for which routes to avoid [35, 142], locations of electric vehicle charging points [50, 59], and so on. Defined in Section 2.4.1, a trajectory, $t$, is as a strictly ordered sequence of instances $[x_1, ..., x_{|t|}]$, in which an instance $x_j$ at time $j$ is a tuple $x_j = \langle lat, long, V_j \rangle$ containing a latitude, $lat$, longitude, $long$, and a vector of vehicle signal values, $V_j$. We hypothesize that the distance error in prediction can be reduced by using the additional data that is contained within the trajectories, such as temporal data or vehicle sensor data, including the vehicle speed and status of doors. Using this additional data, we can group trajectories into more specific clusters than those of BDP, enabling us to (i) lower the average distance between the trajectories within a cluster (and since destination prediction uses cluster centroids, a lower average distance has the potential to reduce the average error), and (ii) improve the prediction of which cluster an unfolding trajectory belongs to. In this chapter, we focus on using temporal data from within the trajectories. However, our method is data agnostic, and can be used with different input data depending on the application and the data available. For example, the number of passengers in a vehicle (obtained from seatbelt status data) could be used as an input to help separate and predict trajectory clusters. We evaluate DPTS using the temporal properties of trajectories to group the clusters, in addition to the spatial information, using data that is implicitly available in the time signal associated with each instance in a trajectory. The evaluation in this chapter assumes that the raw time signal in a trajectory can be translated into a suitable format, e.g., from a unix timestamp to a date and time.

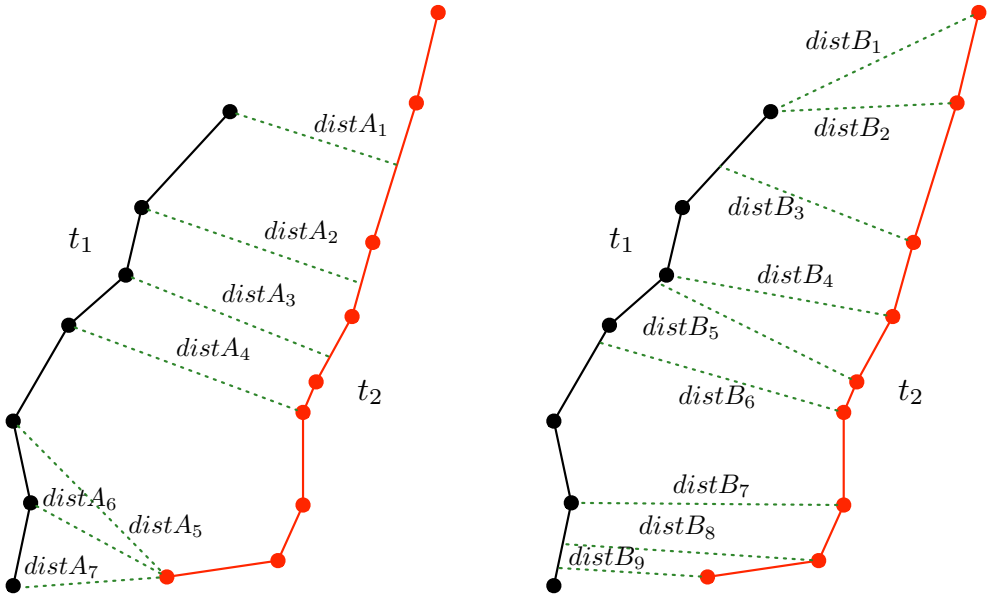
### 6.2.1 Overview & Definitions

DPTS begins by performing an initial clustering of the trajectories, akin to that in BDP. The trajectories are clustered using hierarchical agglomerative clustering, using pairwise dissimilarity matrices. For spatial dissimilarity, we adopt the approach taken by BDP, which uses the Symmetrized Segment-Path Distance (SSPD) to generate the dissimilarity matrices [22]. SSPD uses the Segment-Path distance, which is calculated as the mean of the distances from each of the points in trajectory $t_1$ to the closest segment in trajectory $t_2$ [22]. SSPD is calculated as the mean of the sum of the Segment-Path distance from $t_1$ to $t_2$ and the Segment-Path distance from $t_2$ to $t_1$. Figure 6.1 illustrates an example of the Segment-Path distances between two trajectories $t_1$ and $t_2$. Formally, the SSPD for trajectories $t_1$ and $t_2$ in Figure 6.1 is defined as,

$$\frac{1}{2}\left(\frac{\sum_{i=1}^{n} distA_i}{n} + \frac{\sum_{i=1}^{m} distB_i}{m}\right),$$

Table 6.1: Functions used when defining DPTS.

| **Function** | **Description** |
|---|---|
| $generateDissimilarityMatrix(\mathcal{T}, d)$ | Computes a pairwise dissimilarity matrix for all trajectories, $\mathcal{T}$, according to the distances calculated by the distance function, $d$ |
| $hierarchical(\mathcal{T}, D, \alpha)$ | Performs hierarchical agglomerative clustering on trajectories, $\mathcal{T}$, using the dissimilarity matrix, $D$, according to the clustering criteria, $\alpha$ |
| $getTrajectoriesFromCluster(c_j)$ | Returns the set of trajectories within cluster $c_j$ |
| $likelihood(x_i, \chi_j)$ | Calculates the log-likelihood for instance, $x_i$, to fit the GMM, $\chi_j$ |
| $softmax(likelihood)$ | Performs the softmax function on the log-likelihood, *likelihood* |
| $clusterCentroid(cluster)$ | Returns the centroid of the cluster, *cluster* |
| $BIC(\chi_j)$ | Obtains the Bayesian Information Criterion for GMM, $\chi_j$ |
| $trainGMM(comp, instances)$ | Trains a GMM using *comp* components with the given *instances* |
| $extractInstances(c_j, features)$ | Extracts all instances, containing the selected $features$ from cluster $c_j$ |
| $chooseRandom(instances, n)$ | Selects $n$ random instances from *instances* |
| $getDistanceFunction(m_i)$ | Returns the distance function for the non-null entry in row $m_i$ of the matrix of clustering parameters, $M$ |
| $getClusteringParameter(m_i)$ | Returns the clustering parameter for the non-null entry in row $m_i$ of the matrix of clustering parameters, $M$ |
| $isEmpty(m_i)$ | Returns true if there are only null entries in row $m_i$ of the matrix of clustering parameters, $M$ |
| $encodeDay(t)$ | Returns the encoded day-of-week, [0..6], on which the trajectory $t$ started, s.t. 0 is a Monday and 6 is a Sunday |
| $encodeHour(t)$ | Returns the encoded hour-of-day, [0..23], on which the trajectory $t$ started, s.t. 0 is 12am and 23 is 11pm |




(a) Segment-path distance from $t_1$ to $t_2$

(b) Segment-path distance from $t_2$ to $t_1$

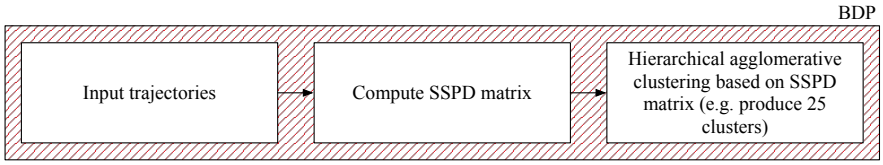
Figure 6.1: Example of the segment-path distances between two trajectories, $t_1$ and $t_2$ [22].

where $n$ is the number of points in $t_1$, $distA_i$ is the minimum distance for point $i$ in $t_1$ to $t_1$, $m$ is the number of points in $t_2$, $distB_i$ is the minimum distance for point $i$ in $t_2$ to $t_1$.

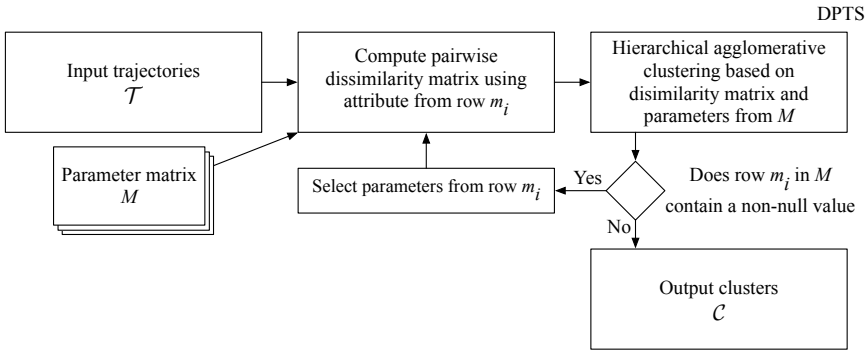
For temporal similarity, we focus on two properties, the day-of-week and the hour-of-day. These temporal properties are only considered for the first instance in a trajectory, unlike the spatial similarity which considers each instance within a trajectory. This is done to minimise the required computation, since the start instance is a key temporal indicator. We define two functions, $encodeDay(t)$ and $encodeHour(t)$, which when given an input trajectory, $t$, convert the time of the first instance into encoded values for the day-of-week and hour-of-day respectively. Since our approach uses hierarchical agglomerative clustering, dissimilarity matrices are required for both the day-of-week and hour-of-day. To create these dissimilarity matrices, we use the definitions of how the differences in the day-of-week and hour-of-day are calculated.

**Definition 6.1.** *The difference in day-of-week between trajectories $t_1$ and $t_2$ is defined as:*

$$diff_{day}(t_1, t_2) = min(abs(encodeDay(t_1) - encodeDay(t_2)), \\ 7 - abs(encodeDay(t_1) - encodeDay(t_2)))$$




(a) An overview of the BDP algorithm on which DPTS is based.




(b) Overview of the DPTS methodology for generating sub-clusters.

Figure 6.2: Overview of the BDP algorithm and the DPTS methodology.

**Definition 6.2.** *The difference in hour-of-day between trajectories* $t_1$ *and* $t_2$ *is defined as*

$$diff_{hour}(t_1, t_2) = min(abs(encodeHour(t_1) - encodeHour(t_2)), \\ 24 - abs(encodeHour(t_1) - encodeHour(t_2)))$$

Figure 6.2 gives a high-level overview of the proposed DPTS methodology, showing how clusters are generated, and highlighting the differences between DPTS and BDP (using the unweighted score). In particular, BDP clusters the input trajectories on spatial distance using SSPD, whereas in DPTS there is an iterative process, in which clustering occurs according to the rows of a parameter matrix, $M$. DPTS is iterative as opposed to clustering in multiple dimensions simultaneously since with an iterative process, an implicit hierarchy is formed, where the user can determine attributes with a higher importance and perform clustering on these first.

**Definition 6.3.** *A DPTS parameter matrix,* $M$*, is a sparse matrix in which each row corresponds to a single clustering iteration, each column corresponds*

*to a clustering attribute, and each entry to the parameter used.*

$$M = \begin{array}{c} \\ iteration_1 \\ iteration_2 \\ \vdots \\ iteration_{i-1} \\ iteration_i \end{array} \begin{array}{c} \begin{array}{cccc} attribute_1 & attribute_2 & ... & attribute_j \end{array} \\ \begin{bmatrix} & m_{1,2} & \dots & \\ m_{2,1} & & \dots & \\ \vdots & \vdots & \ddots & \vdots \\ & & \dots & m_{i-1,j} \\ & & \dots & \end{bmatrix} \end{array}$$

We define a DPTS parameter matrix, $M$, as a sparse matrix in which the column headings correspond to the available attributes on which to cluster, and the rows implicitly indicate which attribute is used for clustering in a given iteration and the parameter value to be used (see Definition 6.3). An attribute is comprised of two parts, namely the signal that is used, such as SSPD, and the measure to be used, such as the maximum cluster criterion. Each row corresponds to a single iteration of the hierarchical clustering process (ordered $1 \dots i$), such that a row contains at most a single non-null entry, $m_{i,j}$, denoting the parameter value, $m$, to be used in iteration $i$ of the hierarchical clustering using the attribute corresponding to column $j$. The number of columns correspond to the number of clustering attributes considered, and the number of rows corresponds to the number of iterations, plus one null row. The final row only contains null entries, which is interpreted as being the termination criteria for clustering. We define two functions to access entries in a parameter matrix, namely, $getDistanceFunction(m_i)$ and $getClusteringParameter(m_i)$. Both functions take as input a single row of the matrix, $m_i$, and identify a non-null column, such that $getDistanceFunction(m_i)$ returns the distance function corresponding to this non-null column and $getClusteringParameter(m_i)$ returns the entry in the column. Both of these functions are undefined for a row containing only null entries.

In our evaluation, discussed later in Section 6.4, we consider three different signals for clustering, namely SSPD, the difference in day-of-week and the difference in hour-of-day. We use the maximum cluster criterion as the measure for the SSPD signal (adopted from BDP [23]), and the distance criterion as the measure for the difference in day-of-week and the difference in hour-of-day. These attributes are denoted *msspd*, *ddow* and *dhod* respectively. In this chapter, our evaluation uses each attribute a maximum of once, meaning that a maximum of 3 clustering iterations are performed. An example parameter matrix is shown in Example 6.1, which will cause DPTS to perform 3 iterations of clustering. The first iteration will use SSPD with a clustering parameter of 25, followed by the hour-of-day with a parameter value of 6 and the final iteration will use the day-of-week, with a parameter value of 2. An illustration

of representing the clustering performed in BDP using a DPTS parameter matrix is shown in Example 6.2. Since BDP only uses SSPD for clustering with a single clustering iteration, the parameter matrix, $M_{BDP}$, only has a single non-null entry in the top-left cell.

**Example 6.1.** *An example parameter matrix, $M_{DPTS}$, for DPTS.*

$$M_{DPTS} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccc} \textit{msspd} & \textit{ddow} & \textit{dhod} \end{array} \\ \begin{bmatrix} 25 & & \\ & & 6 \\ & 2 & \\ & & \end{bmatrix} \end{array}$$

**Example 6.2.** *A representation of example BDP algorithm parameters in the form of a DPTS parameter matrix, $M_{BDP}$.*

$$M_{BDP} = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} \textit{msspd} & \textit{ddow} & \textit{dhod} \end{array} \\ \begin{bmatrix} 25 & & \\ & & \end{bmatrix} \end{array}$$

### 6.2.2 The training stage of DPTS

Algorithm 8 details the approach used to generate the clusters. Given a set of input trajectories, $\mathcal{T}$, and a parameter matrix, $M$, the algorithm starts by selecting the distance function, $d$, and hierarchical clustering parameter, $\alpha$, from the first row, $m_1$, in $M$. The distance function, $d$, is then used to compute a dissimilarity matrix, $D$, over the trajectories, $\mathcal{T}$. Hierarchical agglomerative clustering is then performed using the dissimilarity matrix, $D$, and the clustering parameter, $\alpha$, to generate a set of clusters. For example, using the parameter matrix from Example 6.1, the initial dissimilarity matrix would be computed using the SSPD distance function, and the subsequent hierarchical agglomerative clustering would generate up to 25 clusters. For each further iteration of clustering, represented by the rows, $m_i$, in $M$, dissimilarity matrices are computed over the trajectories, $\mathcal{T}_{c_j}$, in each cluster, $c_j$, in the current set of clusters, i.e., $c_j \in \mathcal{C}$. These dissimilarity matrices are used to generate a further set of clusters, $\mathcal{C}_{m_i}$. Each new set of clusters, $\mathcal{C}_{m_i}$, generated over the current clusters, is appended to $\mathcal{C}'$, for use in the following iteration. This process is repeated for each of the clustering iterations specified in the parameter matrix, $M$, updating the current set of clusters $\mathcal{C}$ at the end of each iteration with the newly calculated clusters $\mathcal{C}'$. Once the current row of the parameter matrix contains only null entries, the algorithm terminates and returns the clusters resulting from the final iteration of clustering.

A set of Gaussian Mixture Models (GMMs), $X$, are trained on the resulting

**Algorithm 8:** $performCluster(\mathcal{T}, M)$

**inputs** : $\mathcal{T}$, a set of $n$ trajectories, $\{t_1, ..., t_n\}$
$M$, a $i \times j$ parameter matrix
**outputs**: $\mathcal{C}$, the set of clusters extracted from all trajectories in $\mathcal{T}$

$\mathcal{C} = \emptyset$
// for each row $m_i$ in M
**for** $m_i \in M$ **do**
  **if** $isEmpty(m_i)$ **then**
    **break**
  **end**
  $d = getDistanceFunction(m_i)$
  $\alpha = getClusteringParameter(m_i)$
  **if** $\mathcal{C} == \emptyset$ **then**
    $D = generateDissimilarityMatrix(\mathcal{T}, d)$
    $\mathcal{C} = hierarchical(\mathcal{T}, D, \alpha)$
  **else**
    $\mathcal{C}' = \emptyset$
    **for** $c_j \in \mathcal{C}$ **do**
      // for each initial cluster, generate a set of sub-clusters
      $\mathcal{T}_{c_j} = getTrajectoriesFromCluster(c_j)$
      $D = generateDissimilarityMatrix(\mathcal{T}_{c_j}, d)$
      $\mathcal{C}_{m_i} = hierarchical(\mathcal{T}_{c_j}, D, \alpha)$
      // add each sub-cluster to the final output
      $\mathcal{C}' = \mathcal{C}' \cup \mathcal{C}_{m_i}$
    **end**
    $\mathcal{C} = \mathcal{C}'$
  **end**
**end**
// return the set of clusters
**return** $\mathcal{C}$

clusters, as specified in Algorithm 9. In DPTS, a feature vector is used to define the features for training the GMMs. In this chapter, we consider the latitude, longitude, encoded day-of-week and encoded hour-of-day. In BDP, the latitude and longitude of an instance are the only features used in the GMM. When using a weighted score, Besse *et al.* use additional variables, such as encoded day-of-week and encoded hour-of-day, and weighting functions to modify the likelihood score, but these variables are not used to sub-divide clusters of trajectories. Our approach can also be extended to include additional data, for example the vehicle signals that are included in each instance, $x_i$. For each cluster, $c_j$, all instances from the trajectories within $c_j$ are extracted, containing the features in the provided feature vector. A sample of these instances is selected uniformly at random and without replacement according to the parameter, $\mu$, which controls the maximum number of instances to select.

**Algorithm 9:** $trainClassifiers(\mathcal{C}, \kappa, \mu, features)$

**inputs :** $\mathcal{C}$, a set of clusters containing trajectories
$\kappa$, the maximum number of components to use for each GMM
$\mu$, the maximum number of instances to select
$features$, the features to train the GMM with

**outputs :** $X$, a set containing the trained GMMs for each cluster in $\mathcal{C}$

```
for j ∈ [1, |C|] do
    ν, χ*_j = ∞, null
    // get all features vectors for all trajectories within cluster
    // e.g., the vector for sspd is lat,long
    instances = extractInstances(c_j, features)
    // pick random sample of μ instances from trajectories
    instances = chooseRandom(instances, μ)
    // for number of components in 1 to κ
    for comp ∈ [1, min(κ, |instances|)] do
        χ_j = trainGMM(comp, instances)
        // use Bayesian Information Criterion, ν, to select model
        if BIC(χ_j) < ν then
            // update best gmm, χ*_j for cluster c_j if better
            χ*_j = χ_j
            ν = BIC(χ_j)
        end
    end
    X = X ∪ χ*_j
end
// return trained GMMs
return X
```

If the number of instances in $c_j$ is less than $\mu$, then all instances are selected. GMMs are built starting with a single component, up to the minimum of either the $\kappa$ parameter or the number of instances, in increments of 1. Each GMM with an increased number of components, $\chi_j$, trained on the selected instances of $c_j$ is evaluated using the Bayesian Information Criterion (BIC) [147], and if it has a lower BIC than the best BIC observed so far, then the best GMM, $\chi_j^*$, and its BIC, $\nu$, are updated with the current values. This is repeated for every cluster, $c_j$, in the set of clusters output from the clustering stage, $\mathcal{C}$, and the trained GMMs are returned in a set, $X$.

The overall training stage of DPTS is detailed in Algorithm 10, in which the GMMs are trained. This method takes six parameters: (i) a set of training trajectories, $\mathcal{T}$, (ii) a parameter matrix, $M_{DPTS}$, (iii) a parameter matrix for BDP, $M_{BDP}$, (iv) the maximum number of components to consider for each

**Algorithm 10:** Training stage of DPTS.

**inputs :** $\mathcal{T}$, a set of $n$ trajectories, $\{t_1, ..., t_n\}$
$M_{DPTS}$, a $i \times j$ parameter matrix for DPTS
$M_{BDP}$, a $i \times j$ parameter matrix representing BDP
$\kappa$, the maximum number of components to use for each GMM
$\mu$, the maximum number of instances to select
$features$, the features to train the GMM with

**outputs :** $X_{BDP}$, a set containing the trained GMMs for each cluster in $\mathcal{C}_{BDP}$
$X_{DPTS}$, a set containing the trained GMMs for each cluster in $\mathcal{C}_{DPTS}$

1 $\mathcal{C}_{BDP} = performCluster(\mathcal{T}, M_{BDP})$
2 $X_{BDP} = trainClassifiers(\mathcal{C}_{BDP}, \kappa, \mu, \langle lat, long \rangle)$
3 $\mathcal{C}_{DPTS} = performCluster(\mathcal{T}, M_{DPTS})$
4 $X_{DPTS} = trainClassifiers(\mathcal{C}_{DPTS}, \kappa, \mu, features)$
`// return matrices of trained GMMs`
5 **return** $X_{BDP}, X_{DPTS}$

GMM, $\kappa$, (v) the maximum number of instances to select when training a GMM, $\mu$, and, (vi) the set of features to use to train the GMMs. The training stage returns two sets of GMMs, $X_{BDP}$ and $X_{DPTS}$, containing the trained GMMs for each cluster in $\mathcal{C}_{BDP}$ and $\mathcal{C}_{DPTS}$ respectively.

The training stage first clusters all trajectories in $\mathcal{T}$, using the parameters defined in $M_{BDP}$, and trains a set of GMMs, $X_{BDP}$, for each cluster in $\mathcal{C}_{BDP}$ using only the latitude and longitude, $\langle lat, long \rangle$, in the feature vector (see Algorithm 9). This is equivalent to performing BDP (with the unweighted score) on the input trajectories. We perform this step to allow DPTS to revert to the prediction made by BDP should its expected performance be better. DPTS then generates a set of clusters, $\mathcal{C}_{DPTS}$, for all trajectories in $\mathcal{T}$ using the parameter matrix, $M_{DPTS}$, (see Algorithm 8). The GMMs in $X_{DPTS}$ are trained with Algorithm 9, using the feature vector input to the algorithm, such as $\langle lat, long, day, hour \rangle$. Once the GMMs have been trained, the training stage of DPTS is complete, which returns two sets of GMMs, namely $X_{BDP}$ trained using the parameters in $M_{BDP}$, and $X_{DPTS}$ using the parameters in $M_{DPTS}$.

### 6.2.3 Trajectory Prediction

Algorithm 11 defines the process of predicting the cluster in which an unfolding trajectory belongs. The log-likelihood for each GMM in $X$ is calculated for each instance, $x_i$, within the trajectory, $t$, and is used to score the GMMs. This algorithm can be run with $X = X_{BDP}$ and $X = X_{DPTS}$, to obtain the respective predictions. The log-likelihood is then translated into a probability

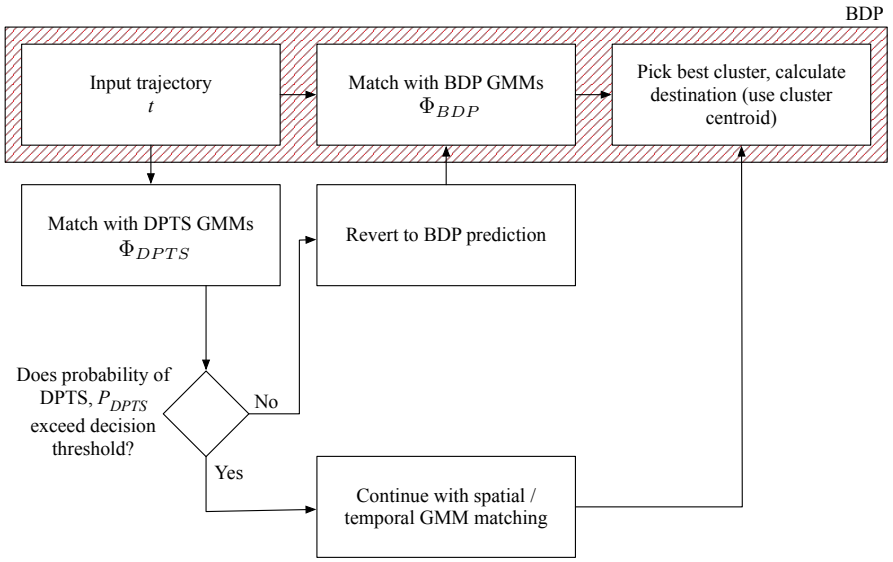



Figure 6.3: Overview of DPTS methodology for using GMMs on trajectory sub-clusterings for destination prediction.

using the softmax function. The prediction algorithm iterates through each instance, $x_i$, in the trajectory, maintaining a sum of the likelihood and probability over all instances. DPTS predicts the cluster for the final instance in the trajectory, where the probability is averaged. As the algorithm iterates through each GMM, $\chi_j \in X$, the total likelihood is compared to the best seen so far, updating the predicted cluster and its respective probability if it exceeds the previous best. Since the GMMs are not trained on whole trajectories, but rather random instances selected from clusters of trajectories (as detailed earlier in the section), our proposed method can provide reasonable predictions on an unfolding trajectory. The method returns the predicted cluster and its probability.

In DPTS, we introduce the notion of a decision threshold, which is the value to be exceeded by the probability of the DPTS prediction in order to use the DPTS prediction. Failing to exceed the decision threshold will cause DPTS to revert to the prediction made by BDP. In DPTS, we consider two modes of decision threshold, namely a static and dynamic mode, controlled by a boolean flag, $\theta_{dynamic}$. The static mode, $\theta_{dynamic} = False$, is where the prediction probability of DPTS, $P_{DPTS}$, is compared to a fixed predefined decision threshold, $\theta$. In the dynamic mode, $\theta_{dynamic} = True$, the DPTS prediction probability, $P_{DPTS}$, is compared to the prediction probability of BDP, $P_{BPD}$, multiplied by the decision threshold parameter value, $\theta$. The decision threshold parameter value, $\theta$, is therefore used to scale $P_{BPD}$ to increase or decrease the likelihood of the exceeding the decision threshold. Such scaling is needed since the BDP prediction probability, $P_{BPD}$, may be

**Algorithm 11:** *predict*$(t, X)$, prediction stage of DPTS.

**inputs :** $t$, a unfolding trajectory to predict
$X$, a set containing the trained GMMs for each cluster in $\mathcal{C}$

**output :** $predictedCluster, bestProb$, the predicted cluster for the last instance in $t$, along with the probability for the prediction

```
1  predictedCluster = null
2  bestLikelihood, bestProb = −∞, −∞
3  for j ∈ [1, |C|] do
4      totalLikelihood = 0
5      totalProbability = 0
6      for x_i ∈ t do
           // calculate likelihood for instance x_i in GMM χ_j
7          likelihood = likelihood(x_i, χ_j)
           // convert likelihood into probability using softmax
              function
8          prob = softmax(likelihood)
           // increment total likelihood and probability
9          totalLikelihood = totalLikelihood + likelihood
10         totalProbability = totalProbability + likelihood
11     end
       // calculate average probability
12     averageProbability = totalProbability/|t|
       // only predict for the latest instance in the
          unfolding trajectory
13     if totalLikelihood > bestLikelihood then
14         predictedCluster = j
15         bestLikelihood = totalLikelihood
16         bestProb = averageProbability
17     end
18 end
19 return predictedCluster, bestProb
```

consistently higher than that of DPTS, $P_{DPTS}$, since the BDP clusters are less specific. Algorithm 12 defines the method to check whether the decision threshold is exceeded or not. The algorithm returns true if the DPTS prediction has exceeded the decision threshold, and therefore will be used for prediction.

The deployment stage of DPTS is illustrated in Figure 6.3 and detailed in Algorithm 13. This method takes five parameters: (i) the unfolding trajectory to predict, $t$, (ii) a set of trained GMMs using BDP, $X_{BDP}$, (iii) a set of trained GMMs using DPTS, $X_{DPTS}$, (iv) a boolean flag that indicates whether to use the dynamic or static mode for the decision threshold, $\theta_{dynamic}$, and (v) the value to use within the decision threshold calculation, $\theta$. The algorithm begins with a given an input trajectory, $t$, for which cluster predictions and their corresponding probabilities, for both BDP and DPTS, are computed. Based on these probabilities, the decision threshold, $\theta$, is evaluated, and if it is exceeded

**Algorithm 12:** *exceedThreshold*$(P_{DPTS}, P_{BDP}, \theta_{dynamic}, \theta)$, decision threshold check in DPTS.

**inputs :** $P_{DPTS}$, average probability of the best GMM for the DPTS clusters, $\mathcal{C}_{DPTS}$
$P_{BDP}$, average probability of the best GMM for the BDP clusters, $\mathcal{C}_{BDP}$
$\theta_{dynamic}$, a boolean flag indicating whether to use dynamic or static mode
$\theta$, the parameter value to use in the decision threshold

**output :** *thresholdExceeded*, whether the decision threshold to use $X_{DPTS}$ was exceeded

```
if θ_dynamic then
    return P_DPTS > (θ * P_BDP)
else
    return P_DPTS > θ
end
```

then the DPTS prediction is used, otherwise the algorithm reverts to using the prediction made by BDP. The predicted destination itself is obtained by taking the cluster centroid of the predicted cluster.

## 6.3 Experimental Methodology

In this chapter, we use three separate datasets to evaluate DPTS, two of which are those used by Besse *et al.* to evaluate BDP [23], on which DPTS is based. The first of these, the Caltrain dataset, contains 4,127 taxi trajectories originating from Caltrain Station, San Francisco [130]. The second, the Porto dataset, contains 19,423 taxi trajectories commencing from Sao Bento station, located in the centre of Porto [1]. The third dataset, is a subset of our POLD (as introduced in Chapter 3, which consists of trajectories for a single participant collected over a number of non-consecutive weeks. We refer to this subset as the POLD for the remainder of this chapter. Unlike the Caltrain and Porto datasets, the POLD does not have a single starting location for all trajectories, and so allows us to evaluate the performance of DPTS when trajectories do not have a common starting location. Since there will be much more spatial dissimilarity between trajectories at the start of a journey, this has the potential to increase the difficulty of generating meaningful clusters. Detailed descriptions and statistics for these datasets are presented in Chapter 3.

For all stages of the evaluation, unless explicitly stated, we explore in detail the effect of the parameters on the Caltrain dataset, and state the best results for the Porto dataset. Due to the different nature of the POLD, we evaluate DPTS on the POLD separately in Section 6.4.5. Unless explicitly stated, our

**Algorithm 13:** Deployment stage of DPTS.

**inputs :** $t$, a unfolding trajectory to predict
$X_{BDP}$, a set containing the trained GMMs for each cluster in $\mathcal{C}_{BDP}$
$X_{DPTS}$, a set containing the trained GMMs for each cluster in $\mathcal{C}_{DPTS}$
$\theta_{dynamic}$, a boolean flag indicating whether to use dynamic or static mode
$\theta$, the parameter value to use in the decision threshold

**output :** $predictedDestination$, the predicted destination for trajectory, $t$

$prediction_{BDP}, P_{BDP} = predict(t, X_{BDP})$
$prediction_{DPTS}, P_{DPTS} = predict(t, X_{DPTS})$
**if** $exceedThreshold(P_{BDP}, P_{DPTS}, \theta_{dynamic}, \theta)$ **then**
  $predictedDestination = clusterCentroid(prediction_{DPTS})$
**else**
  $predictedDestination = clusterCentroid(prediction_{BDP})$
**end**
**return** $predictedDestination$

Table 6.2: Set of parameters, $\alpha$, used for our initial evaluation.

| **Experiment #** | **Order of Clustering** | **Parameters ($\alpha$) Considered** |
|---|---|---|
| 1–4 | SSPD → Day-of-week | 25 → [0,3] |
| 5–13 | SSPD → Hour-of-day | 25 → [0,8] |
| 14–17 | Day-of-week → SSPD | [0,3] → 25 |
| 18–53 | Day-of-week → Hour-of-day | [0,3] → [0,8] |
| 54–62 | Hour-of-day → SSPD | [0,8] → 25 |
| 63–98 | Hour-of-day → Day-of-week | [0,8] → [0,3] |

comparison against the baseline BDP method uses the unweighted score, rather than relying on auxiliary variables and weighting functions to modify the score since, as noted in Chapter 2, our focus is on identifying suitable clusters from which to make predictions. We comment on the effectiveness of our method on these datasets, noting the differences. In this chapter, we use a value of 10000 for $\mu$ and 20 for $\kappa$, since these parameters are not the focus of our investigation and these values were used in the original evaluation of BDP, allowing for a direct comparison [23].

The first stage of our evaluation of DPTS investigates the order of clustering and the parameters for the day-of-week and hour-of-day clustering, to find the best performing values for each. We perform all combinations of clustering with SSPD, day-of-week and hour-of-day using two iterations. Within this parameter search, we use a decision threshold of 0 in the static mode, meaning that the

DPTS prediction will always be used. Table 6.2 shows the set of parameters used in this evaluation. For the SSPD clustering, we use the parameter values from the work of Besse *et al.*, which are 25 and 45 for the Caltrain and Porto datasets respectively. We train the GMMs with 4 different feature vectors, namely $\langle lat, long \rangle$, $\langle lat, long, day \rangle$, $\langle lat, long, hour \rangle$, and $\langle lat, long, day, hour \rangle$, resulting in 392 sets of results for each dataset. Evaluating the mean distance error for each parameter combination against the baseline performance of BDP, we discard those that are significantly outperformed by the baseline from further evaluation.

After the parameter search has been completed, the next stage evaluates the effect of our proposed decision thresholds on performance. We analyse the decision threshold in both static and dynamic modes, and compare these results to both the baseline performance of BDP and the performance of DPTS where the decision threshold is set to 0. For the static and dynamic modes of the decision threshold, we explore the parameter value, $\theta$ in the range [0,1], in increments of 0.05 and 0.1 respectively.

The third stage of our evaluation explores the impact of the clustering parameter for SSPD. In our initial analysis, we use the best performing parameter for each dataset, as reported by Besse *et al.* [23], and so we also investigate a range of values for the SSPD clustering parameter, in increments of 5. Our stopping criteria is where the supplied parameter value causes an error due to the number of clusters being too large, and therefore not giving significant data to properly train the GMMs.

In the next stage of our evaluation, we add a third iteration of clustering to DPTS, considering SSPD, day-of-week and hour-of-day simultaneously. The ordering of clustering iterations is evaluated, and the performance of three iterations is compared to that of using two iterations, using the mean distance error.

For the final stage of our evaluation we consider destination clustering, specifically on the POLD, which looks at the spatial dissimilarity between destinations. The evaluation of the POLD is notable since, unlike the previous datasets, the POLD does not contain a single starting location for all journeys. To explore this aspect, we propose adding a fourth clustering approach which groups trajectories based on the trajectory destinations, using the Haversine distance between each destination to generate the dissimilarity matrix, $D$. The main motivation for this is to reduce the number of clusters generated, since multiple starting locations will increase the overall spatial dissimilarity between trajectories.

## 6.4 Results

In this section, we discuss the results of applying DPTS to the Caltrain [130] and Porto [1] datasets, in addition to the POLD. We evaluate the effect of the clustering parameters, and analyse the impact of introducing a decision threshold, using the evaluation approach outlined in the previous section. Unless stated, the results presented in this section are based on the Caltrain dataset [130]. Due to its distinct nature, the POLD is evaluated separately in Section 6.4.5. Note that for simplicity figures that have trajectory completion on the x-axis have an origin of 0%, however the data points start from the first instance of the trajectory.

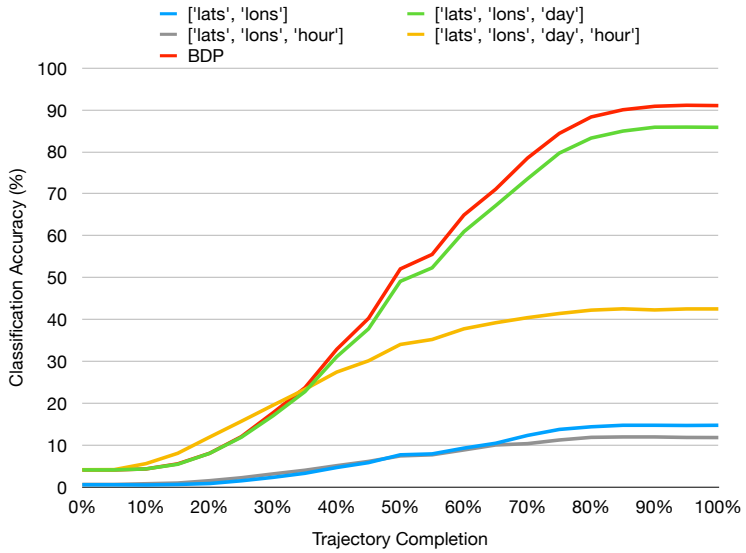
### 6.4.1 Clustering Parameter Search

This section evaluates our novel iterative clustering approach, and the impact of altering the parameters within the parameter matrix, $M_{DPTS}$. In this analysis, we discuss in detail the effect of altering the parameters on the Caltrain dataset, and simply report the best performing parameters on the Porto dataset.

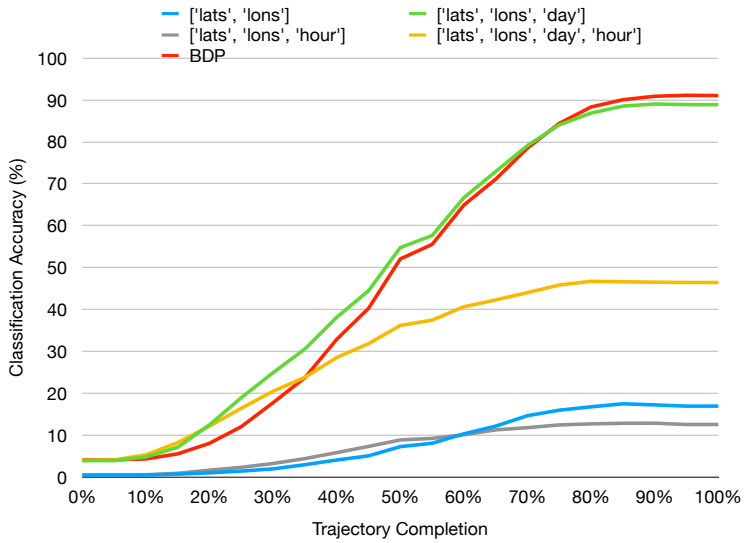
We first give an overview the classification performance for each of the 6 parameter combinations outlined in Table 6.2. Note that there is a strong correlation between the features used in the GMM and the clustering criteria. For example, if the hour-of-day is used to cluster the trajectories but is not present in the feature vector provided to the GMM, then the performance is severely degraded. The exception to this is that the $\langle lat, long \rangle$ features are always needed in the feature vector to achieve a reasonable performance, even if SSPD was not included in the clustering stage. The classification performance for the top performing parameters for each combination are shown in Figures 6.4 and 6.5, in addition to the baseline performance.

Clustering with SSPD followed by the day-of-week achieves a peak performance of 85.90% at 95% trajectory completion. This was obtained by setting clustering parameter for the day-of-week to $\alpha = 2$, and $\langle lat, long, day \rangle$ as our feature vector. If the day-of-week is omitted from the feature vector, then the performance falls to a maximum of 14.73%. Interestingly, if the hour-of-day is also included, i.e., $\langle lat, long, day, hour \rangle$, the performance sees a notable drop, with a maximum of 42.52% at 85% trajectory completion. These results are shown in Figure 6.4a.

Conversely, if we cluster using the day-of-week followed by SSPD (see Figure 6.4b), then the clustering parameter, $\alpha$, does not make any difference to the performance. Slightly decreased performance is observed in the first 10% of trajectory completion, but after this the performance exceeds that of having SSPD followed by the day-of-week. The peak performance is 89.02%,

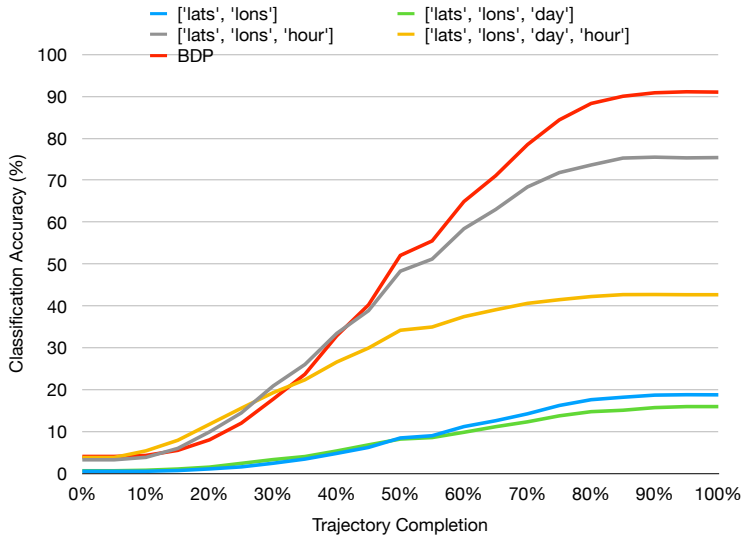



(a) SSPD → Day-of-week

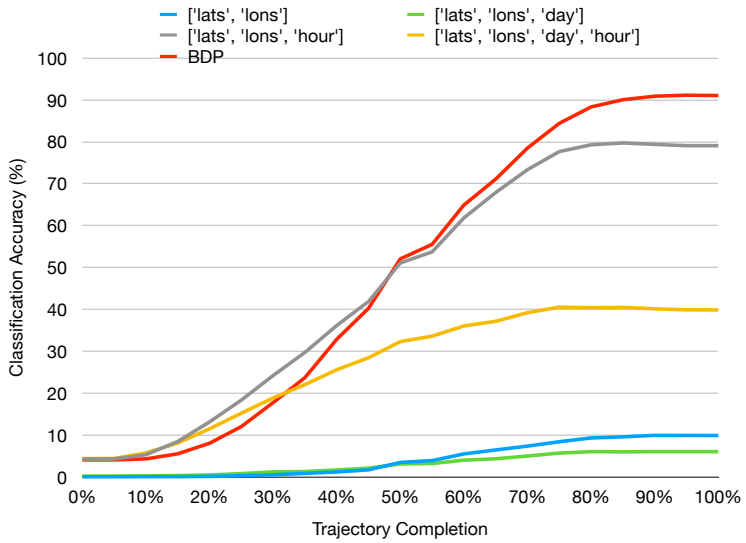



(b) Day-of-week → SSPD

Figure 6.4: Classification performance when using day-of-week for DPTS on the Caltrain dataset.




(a) SSPD → Hour-of-day




(b) Hour-of-day → SSPD

Figure 6.5: Classification performance when using hour-of-day for DPTS on the Caltrain dataset.

Table 6.3: Performance comparison between each two-iteration clustering combination.

| **Clustering Order** | **Accuracy (%)** | **Avg. Cluster Distance (m)** |
|---|---|---|
| SSPD, $\alpha = 25 \rightarrow$ Day, $\alpha = 2$ | 85.85 | 544 |
| Day, $\alpha = 0 \rightarrow$ SSPD, $\alpha = 25$ | 88.85 | 511 |
| Hour, $\alpha = 8 \rightarrow$ SSPD, $\alpha = 25$ | 79.06 | 405 |
| SSPD, $\alpha = 25 \rightarrow$ Hour, $\alpha = 1$ | 75.41 | 448 |
| Day, $\alpha = 0 \rightarrow$ Hour, $\alpha = 6$ | 99.18 | 1439* |
| Hour, $\alpha = 8 \rightarrow$ Day, $\alpha = 2$ | 99.03 | 1424* |

achieved at 90% trajectory completion. Similar to SSPD followed by the day-of-week, omitting day-of-week from the feature vector causes a noticeable drop in performance, as does the addition of the hour-of-day.

When clustering by the hour-of-day followed by SSPD, we observe a peak performance of 79.21% at 85% trajectory completion, as illustrated in Figure 6.5b. There is a noticeable degradation in performance when not using the hour-of-day in the feature vector, as seen in the previous results. If we reverse the order of clustering to have SSPD followed by hour-of-day, a peak performance of 75.58% is achieved at 90% trajectory completion (see Figure 6.5a). From these results, we can see that higher performance is achieved when the temporal component (day-of-week or hour-of-day) is clustered prior to the spatial component, SSPD.

If we consider both temporal components, the day-of-week and the hour-of-day, the classification performance is misleading. The day-of-week and the hour-of-day are taken from the start of the trajectory, and therefore their respective values are constant throughout. These combinations are unsuitable due to the little information they provide.

We take the best performing parameters from each of the 6 clustering combinations, using the classification percentage at 100% trajectory completion. The parameters, and the feature vector used for each of the top combinations is shown in Table 6.3. The temporal-only combinations are included for reference, but show a misleading classification accuracy as noted above. Figure 6.6 illustrates each of the top combinations from Table 6.3 against the baseline performance, BDP. Most of the performance gain can be seen in the initial 40% of the unfolding trajectories, after which BDP starts to outperform the DPTS combinations. Due to the misleading performance, the temporal-only combinations are omitted from Figure 6.6.

If we consider the predicted clusters and calculate the distance error from the prediction to the ground truth, we obtain the results shown in Figure 6.7. The first point to note is the two straight lines, which show the prediction error

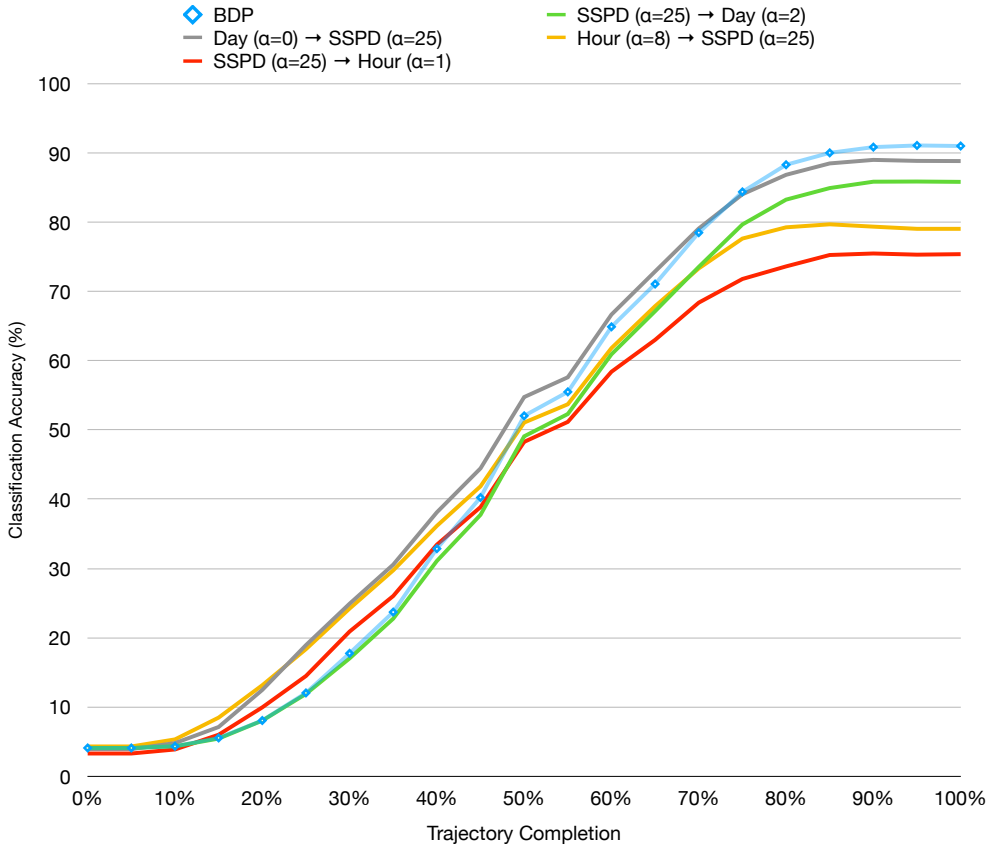



Figure 6.6: Comparison of the BDP performance against each of the top DPTS combinations on the Caltrain dataset.

of both temporal-only combinations. This is expected, since the temporal values provided to the GMM do not change as the trajectory progresses, but it may not be immediately apparent as to why such high classification performance translates to a large prediction error. If we refer back to Table 6.3, we note the large average cluster distances for the temporal combinations. This explains the high distance error, because even though the classification performance is good, the clusters are noticeably larger, and therefore the centroid that is used for prediction is on average further from the actual destination. When clustering with SSPD and then the day-of-week, we see no improvement over the baseline. The other combinations, day-of-week to SSPD, hour-of-day to SSPD and SSPD to hour-of-day, all show reductions in distance error over the baseline from 20% to 60% of trajectory completion. After 70% of trajectory competition, the baseline performance is unbeaten. Given that we saw no improvement when clustering from SSPD to day-of-week, and that the temporal combinations have such large cluster distances, we omit these combinations from further evaluation.

When applying DPTS to the Porto dataset, the hour-of-day ($\alpha = 0$) $\rightarrow$ SSPD ($\alpha = 45$) combination, gives the highest performance. Even though

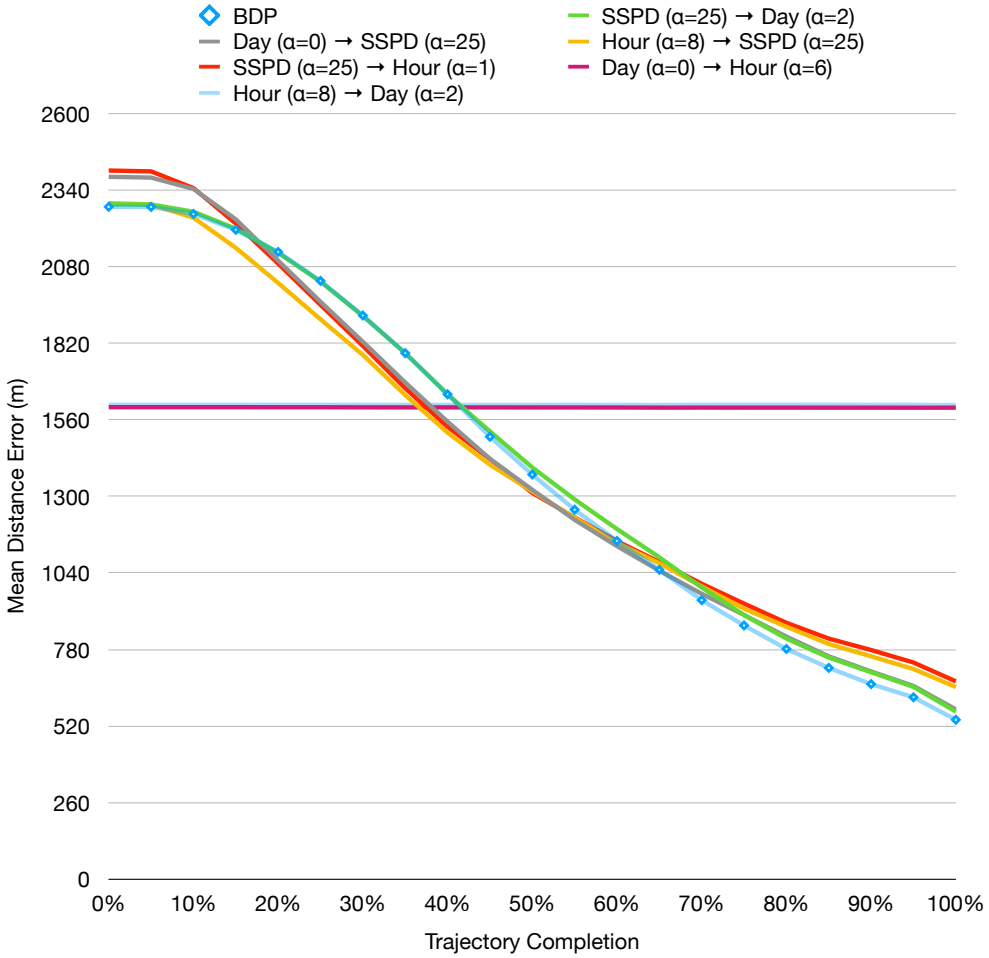



Figure 6.7: Comparison of the prediction error from BDP against each of the top parameter combinations for DPTS on the Caltrain dataset.

there is a slight improvement in the middle of the trajectories, the overall performance is lower than that of BDP, due to degraded performance at the start and end of the trajectories. This follows the trend seen with the Caltrain dataset. Overall, DPTS is outperformed by BDP on the Porto dataset, by an average of 7 metres.

### 6.4.2 Evaluation of the Decision Threshold

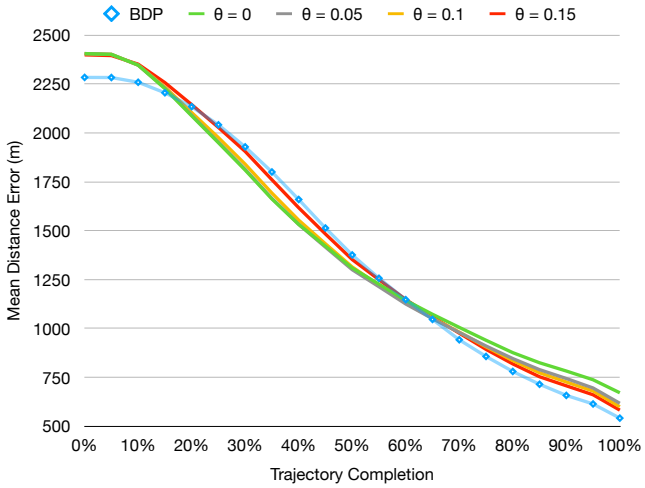
Considering the results discussed in Section 6.4.1, we see that the baseline performance exceeds that of DPTS in the final portion of the journey. To address this, we propose using a decision threshold, that combines our novel method, DPTS, and the existing method, BDP, within a single wrapper. The decision threshold selects a prediction to use at different stages of the unfolding trajectories, according to the prediction probability of DPTS, $P_{DPTS}$, and BDP, $P_{BDP}$. As described in Section 6.2, we consider two modes for the decision threshold, namely a static mode ($\theta_{dynamic} = False$) and a dynamic

mode ($\theta_{dynamic} = True$).

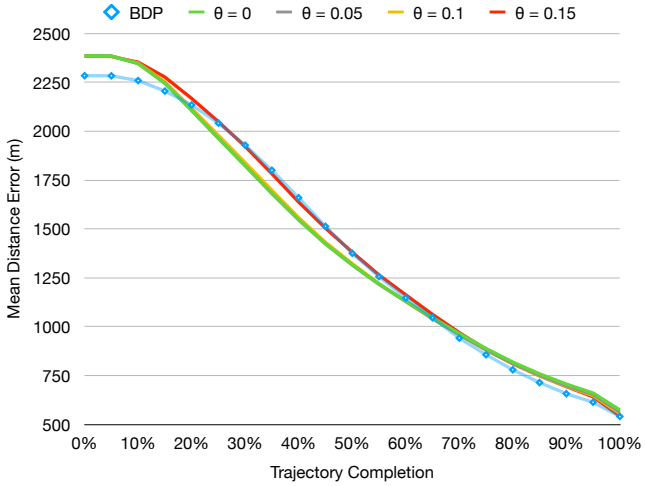
First we evaluate the effect of a decision threshold in the static mode, by considering values in the range [0,1] with increments of 0.05. The effect of the decision threshold, $\theta$, on SSPD $\rightarrow$ hour-of-day is shown in Figure 6.8a. A decision threshold of 0 in the static mode is essentially removing consideration of BDP, since all probabilities greater than 0 will pass, and therefore the result will be identical to our original results. Conversely, a decision threshold of 1 will always revert to the baseline results of BDP. We can see that setting a threshold of 0.05 improves the performance past 50% trajectory completion, with no apparent loss of performance below 50% completion. If we increase the decision threshold to 0.1, we notice a loss of performance (compared to a decision threshold of 0) from 15–50% of trajectory completion, after which, the performance improves. Further increasing the decision threshold to 0.15 leads to a more significant degradation in performance from 10–65% of trajectory completion, after which a small improvement is made for the remainder of the journey. At this decision threshold, we also see a slight improvement in performance in the first 5% of trajectory completion compared to our original results. Any further increase in the decision threshold has the effect of improving the first part of the journey (0–15% of trajectory completion), degrading the middle of the journey (15–65% of trajectory completion) and improving the final part of the journey (65–100% of trajectory completion). Overall, in static mode, a decision threshold of 0.05 gives the best trade-off, resulting in the highest average performance for SSPD $\rightarrow$ hour-of-day.

Figure 6.8b illustrates the effect of the decision threshold in static mode on day-of-week $\rightarrow$ SSPD. We observe a similar trend to SSPD $\rightarrow$ hour-of-day, but note that the original result (with a decision threshold of 0) performs nearer to the baseline result in the final stage of the trajectories (65–100% of trajectory completion). Therefore, it seems that adding a decision threshold will have a smaller positive impact on this combination. Decision thresholds of 0.05 and 0.1 provide a good trade-off between performance in the middle and final parts of the journey. We note that a decision threshold of 0.15 gives a greater loss of performance in the middle of the journey, similar to that reported in the analysis of SSPD $\rightarrow$ hour-of-day. In the static mode, a decision threshold of 0.05 also gives the best trade-off for day-of-week $\rightarrow$ SSPD performance.

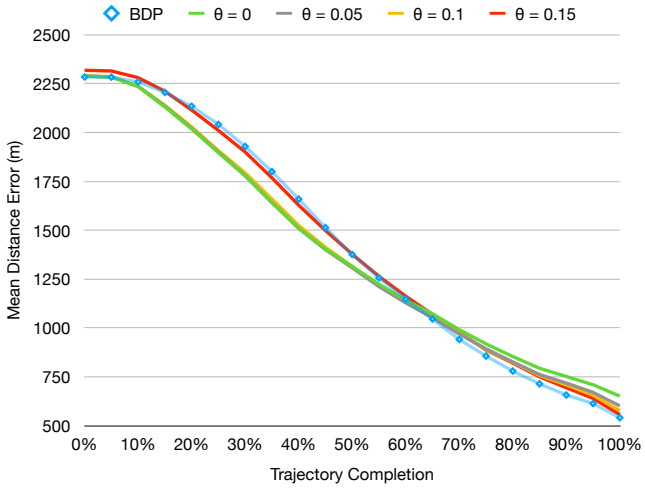
The combination of hour-of-day $\rightarrow$ SSPD, as shown in Figure 6.8c, appears to give the best results of the three alternatives. Most notably, the early part of the journey (0–10% of trajectory completion), is nearer the baseline performance than the other two combinations. As with the other results, we see that a decision threshold of 0.05 gives the optimum performance trade-off, with more apparent losses seen for decision threshold values of 0.15 and above. All three sets of results appear to provide the best overall performance when a




(a) SSPD → Hour-of-day




(b) Day-of-week → SSPD




(c) Hour-of-day → SSPD

Figure 6.8: Comparison of destination prediction performance for given decision threshold values in static mode.

decision threshold of 0.05 is used, with a more significant loss of performance with a decision threshold of 0.15.

We will now consider the decision threshold in dynamic mode ($\theta_{dynamic} = True$), to investigate whether this outperforms the static mode ($\theta_{dynamic} = False$). Figure 6.9 shows the performance when a decision threshold is used in dynamic mode. The decision threshold in dynamic mode ($\theta_{dynamic} = True$), as explained in Section 6.2, is where the probability of the DPTS prediction, $P_{DPTS}$, is compared directly to the probability of the baseline BDP prediction, $P_{BDP}$. The decision threshold parameter value, $\theta$, is used to scale the probability of the BDP prediction, $P_{BPD}$. For our evaluation we explored parameter values in the range [0,1] in increments of 0.1.

Overall, we found that a decision threshold value of 0.7 for SSPD $\rightarrow$ hour-of-day and 0.4 for day-of-week $\rightarrow$ SSPD and hour-of-day $\rightarrow$ SSPD gave the best prediction performance. On average, using the decision threshold in dynamic mode causes a slight improvement in performance compared to the static mode. This gain, however, is minimal in terms of metres, and appears to be of little effect, but could be influenced by properties of the input dataset.

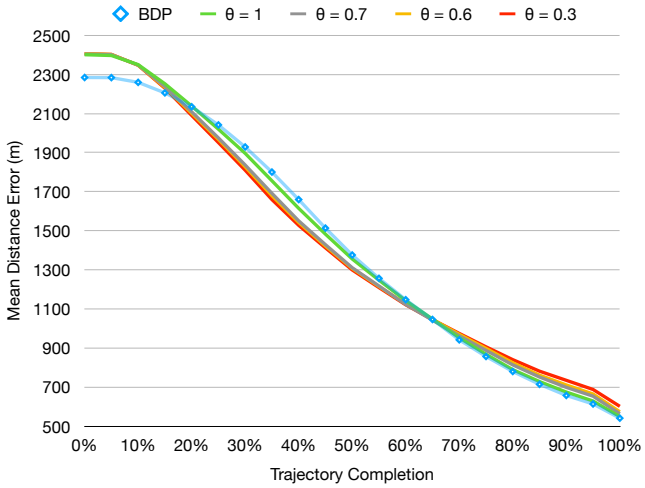
When evaluating the decision threshold on the Porto dataset, a slight improvement over the performance of BDP is seen. A decision threshold in the dynamic mode with a parameter value of $\theta = 0.9$, was used on the hour-of-day ($\alpha = 0$) $\rightarrow$ SSPD ($\alpha = 45$) combination, giving an average distance error of 10 metres lower than BDP. Figure 6.10 illustrates the performance comparison between BDP, DPTS ($\theta = 0$) and DPTS ($\theta = 0.9$).

The use of a decision threshold, where our method reverts back to the original prediction made by BDP, does not improve the distance error across the whole trajectory, but instead provides higher performance towards the end of a journey for the trade-off of reduced performance in the mid-journey. The trade-off between mid to end of journey performance can be evaluated based on the application.

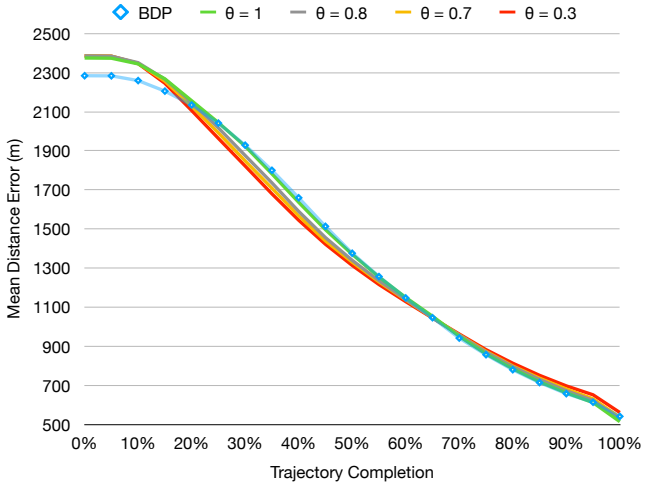
### 6.4.3 Altering the SSPD parameter values

We investigated changing the clustering parameter, $\alpha$, on the highest performing combinations. In the results discussed above, this was fixed at the values used by Besse *et al.* in their investigation [23]. Intuitively, lowering the parameter value in BDP, should increase the trajectory classification but also increase the destination prediction error, since the destinations in these larger clusters will be more spread out. However, since DPTS performs iterative clustering, there may be benefits to lowering the $\alpha$ value for SSPD.

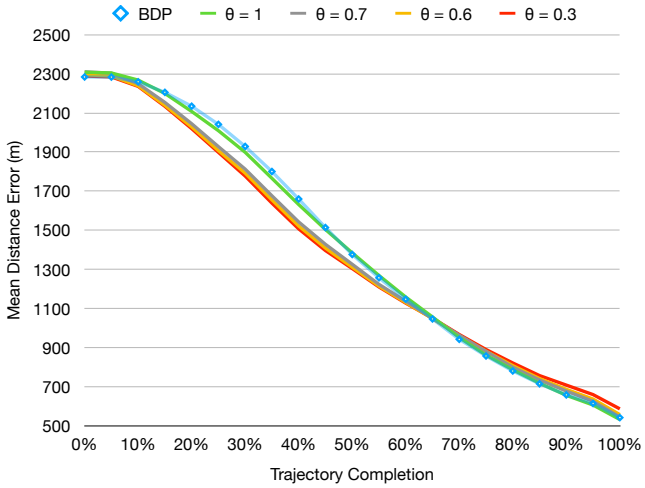
Figure 6.11 illustrates the comparison between BDP, DPTS (hour-of-day, $\alpha = 8 \rightarrow$ SSPD, $\alpha = 25$), DPTS (SSPD, $\alpha = 10 \rightarrow$ hour-of-day, $\alpha = 6$) with




(a) SSPD → Hour-of-day




(b) Day-of-week → SSPD




(c) Hour-of-day → SSPD

Figure 6.9: Comparison of destination prediction performance for given decision threshold values in dynamic mode.

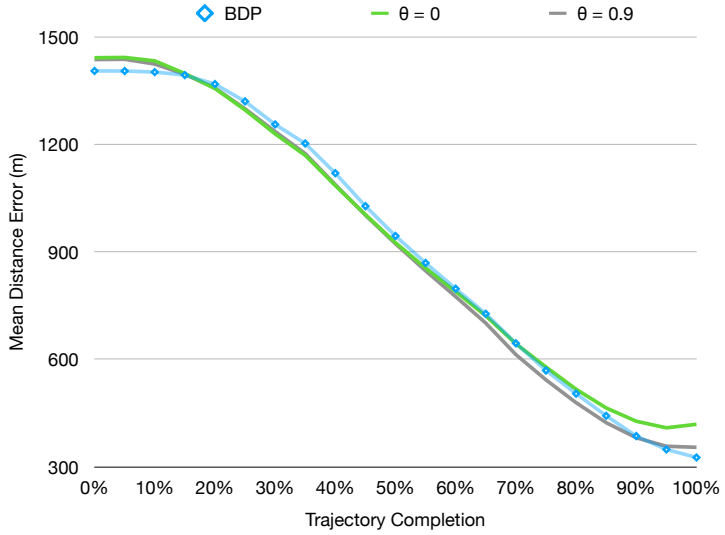



Figure 6.10: Comparison of the prediction error for the Porto dataset.

$\theta = 0.0$, and DPTS (SSPD, $\alpha = 10 \rightarrow$ hour-of-day, $\alpha = 6$) with $\theta = 1.0$. It is immediately apparent that the reducing $\alpha$ provides a significant reduction in destination error in the first portion of the journey. After 60% of the trajectory is complete the performance degrades below the performance of BDP. When introducing a decision threshold greater than 0, the performance gains are comparable at the start of the journey, and the performance degradation is slightly reduced past 65% trajectory completion. Overall, the variant with $\alpha = 10$ and a decision threshold of $\theta = 1.0$ provides the best performance on average over the entire duration of the journey, with significant gains in the first 30–40% of the trajectory.

If we compare DPTS (SSPD, $\alpha = 10 \rightarrow$ hour-of-day, $\alpha = 6$) with $\theta = 1.0$ with the weighted version of BDP, we observe similar performance at the start of the journey. As the trajectory unfolds, there is a larger performance gap between DPTS and the weighted version of BDP, with BDP seeing a maximum of 366 metres lower prediction error at some points.

When applied to the Porto dataset, no gains in performance were observed, and the original clustering parameter for SSPD, $\alpha = 45$, produced the highest performance.

### 6.4.4 Adding a third clustering iteration

We now evaluate the performance of DPTS using three iterations of clustering, and compare the performance with using two iterations. The motivation behind including an additional iteration is that we can further group the clusters (whilst trying to maintain a high accuracy for the trajectory classification). The drawback of adding a third iteration is that it can generate large numbers of clusters, each containing only a few trajectories. If the number of clusters

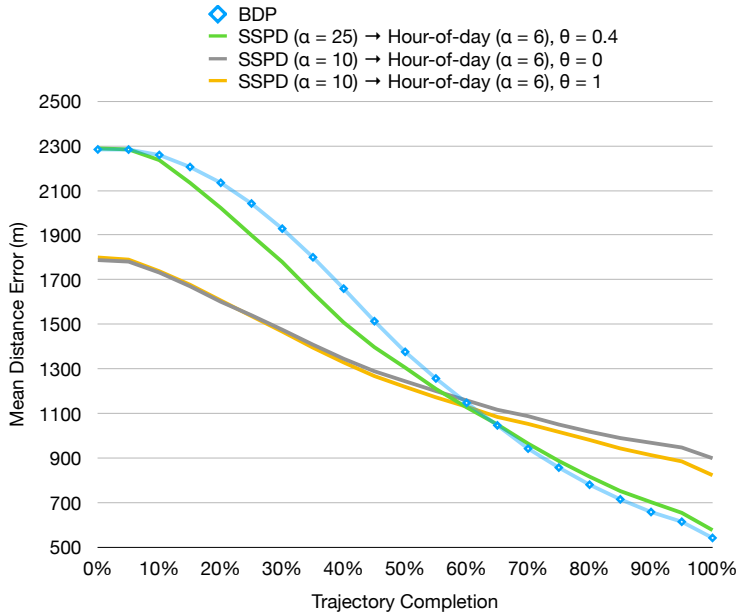



Figure 6.11: Comparison of the prediction error for the Caltrain dataset, when altering the clustering parameter value, $\alpha$, for SSPD.

increases too much, a situation where some clusters only contain a single trajectory may occur, and therefore the cluster has no training data and is not useful for prediction.

When using three iterations of clustering, we find the best combination to be SSPD → hour-of-day → day-of-week. However, Figure 6.12 shows that the performance of this combination is not as high as that of two iterations with a reduced $\alpha$ for SSPD (as discussed above). When we add a decision threshold in dynamic mode, the performance is degraded in the initial 40% of the trajectories, but sees improved performance from 60% completion onwards, nearer to that of the BDP. Taking into consideration the average distance error throughout the trajectory, the extra computation required for the additional layer, and the increased number of GMMs required, we take the previous combination with two clustering iterations to be the better variant.

### 6.4.5 Evaluating DPTS on the POLD

Applying DPTS to the POLD provides an insight into a more general application of the algorithm, since unlike the other datasets, the POLD contains trajectories with multiple starting locations. When applying BDP to the POLD, we notice an increase in distance error at around 50–80% trajectory completion. This is because, unlike the taxi datasets, we do not have a single starting location, and therefore we can not assume a fixed direction of travel away from the source. To address this issue, we add another iteration of clustering, in which we generate a dissimilarity matrix of trajectories based on the destination

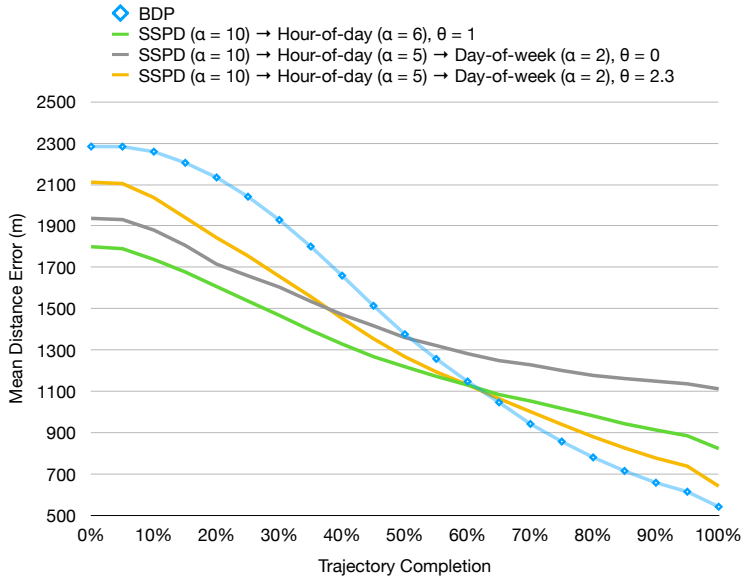



Figure 6.12: Comparison of the prediction error for the Caltrain dataset between two and three iterations of clustering in DPTS and BDP.

location to be used as input to the hierarchical agglomerative clustering. For our evaluation we use 2500 metres as the clustering parameter, $\alpha$, for this iteration of clustering. Further exploration of this value is outside the scope of this chapter, and could be investigated in future work.

Figure 6.13 shows the results of applying DPTS to the POLD, with a comparison to the performance of BDP. When we apply DPTS, using four iterations of clustering (hour-of-day, $\alpha = 0 \rightarrow$ day-of-week, $\alpha = 0 \rightarrow$ destination, $\alpha = 2500 \rightarrow$ SSPD, $\alpha = 30$), we see a significant improvement over BDP. This combination outperforms BDP over the entire trajectory, with an average reduction in error of over 1.2km. A small decrease in error can be seen as the trajectory unfolds, unlike the sudden rise in error as seen with BDP.

## 6.5 Summary

In this chapter, we propose DPTS, an extension to the existing BDP method. DPTS uses an iterative clustering stage and a decision threshold to improve the destination prediction performance on vehicle trajectories. DPTS harnesses the additional properties of the trajectories, attempting to further group them into more meaningful clusters, rather than using these temporal properties in combination with weighting functions to modify the likelihood. For our evaluation, we use the temporal properties of day-of-week and hour-of-day.

When applying DPTS to the Caltrain dataset, we see an improvement in overall performance, where our decision threshold allows the prediction to revert back to that of BDP towards the end of the trajectories, as the

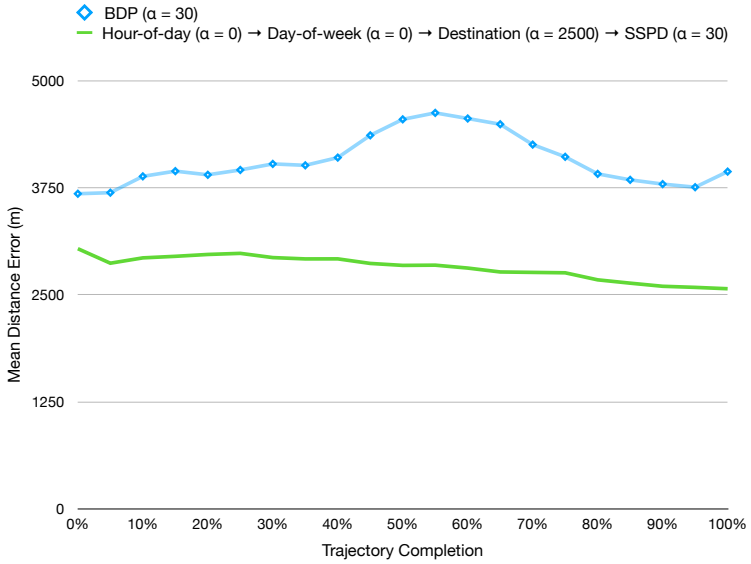



Figure 6.13: Comparison of the prediction error between BDP and DPTS when applied to the POLD.

performance of BDP improves. If two iterations of clustering are used, with smaller parameter values for SSPD, we see a reduction in error for the first half of the journey compared to BDP, however this is at a cost of lower performance in the final 40% of trajectories. We see severely reduced effectiveness from DPTS when used on the Porto dataset, barely matching the performance of BDP. The Caltrain and Porto datasets have different characteristics, such as the size of the area, the number of journeys, and the frequency of sampling. The difference in performance could imply that the capability of our method is impacted by these different characteristics. However, when applied to the POLD, which has multiple starting locations, we see promising results. BDP struggles to accurately predict destinations, with an increase in error in the middle of the trajectories. When applying DPTS with multiple clustering iterations, we see notable gains in prediction performance over BDP, that are consistent throughout the unfolding trajectories. This implies that selecting the best approach in practice is highly dependant on the application setting and the nature of the data available. In practise, we recommend adding clustering iterations for attributes that help differentiate clusters when using DPTS for applications. We also recommend the consideration of both static and dynamic thresholds, adjusting parameter values to maximise performance and balance the trade-off between performance in the early stages of a journey against that in the latter stages.

Reducing the prediction error can provide benefits to location-aware applications, such as on-route traffic updates, intelligent parking suggestions and amenity recommendations at the destination. Without an accurate location,

these applications will suffer from reduced effectiveness and ultimately poor user trust. Having a more accurate prediction earlier in the trajectory can enable these applications to provide their location-based functionality in a more timely manner.

# Chapter 7

# Conclusions and Future Work

In this thesis, we have explored human mobility patterns from vehicle usage, considering location-aware applications that can benefit from knowledge of a vehicle's trajectory. We have proposed two novel methods namely, Activity-based PoI Extraction (AVPE) and Destination Prediction by Trajectory Sub-Clustering (DPTS), which tackle point of interest (PoI) extraction and destination prediction respectively. With AVPE and DPTS, we have achieved improvements over current state-of-the-art approaches by increasing accuracy, enabling more accurate location data to be provided to future applications. Additionally, we proposed a method that utilised the activities predicted by AVPE, and constructed sequences of activity annotated locations that are used for destination prediction.

We have made contributions to data mining on trajectory data, exploring new avenues to tackle existing problems, building on state-of-the-art approaches, and ensuring that it is applicable to vehicle trajectory data. To support the research into both PoI extraction and destination prediction, we have collected two datasets containing vehicle sensor data alongside GPS data, which are used in our investigations. These datasets, namely the Location Extraction Dataset (LED) and the Pattern of Life Dataset (POLD), contain a full ground truth labelling for activities and destinations respectively. The LED has been made available for other researchers to use in future research[1].

## 7.1 Contributions

This section summarises each contribution in this thesis, discussing the overall findings and limitations.

1. **Using on-board vehicle data to classify the activity of a vehicle with the aim of improving point of interest detection within vehicle trajectories**

   In Chapter 4 we proposed Activity-based Vehicle PoI Extraction (AVPE) which uses activity classification to filter out false PoIs. Using existing

[1] For privacy reasons we are not able to publish the POLD.

state-of-the-art clustering algorithms, we showed that a high amount of false PoIs are extracted from vehicle data, and that AVPE was able to remove most of these (up to 99.0% in some cases) when applied to the LED. When we applied AVPE on the POLD, we observed performance comparable to that shown when using the LED. A limitation of AVPE is that it requires on-board vehicle signals to be available, and that PoIs can only be extracted retrospectively, after the journeys have been completed. Similarly, there is a high misclassification rate between drop-off and pick-up activities, which despite both being classified as a PoI for the task of filtering out false PoIs, may benefit from higher accuracy if used in other applications. Finally, AVPE suffers from a high rate of true PoIs being discarded. AVPE aims for correctness over completeness, however it is desirable to lower the number of true PoIs that are lost.

2. **Applying destination prediction with activity annotations on stay points in vehicle trajectories**

   In Chapter 5 we investigated the use of activity annotations locations in predicting the next destination. As a baseline for comparison, we compared sequences of locations without activity annotations to annotated sequences. Overall, we observed poor performance when using the annotated sequences of locations, in which the annotated sequences only outperformed the unannotated sequences of locations on a single participant's journeys. The evaluation of this method suffers from a small amount of data available which limits our findings. Additionally, our method produces a significantly large distance error between the predicted location and the ground truth. Our hypothesis that the chosen activity labels would provide meaningful insights was wrong, since the frequency of activity labels other than traffic or parked was relatively low. Consequently, there may not be enough variation in activities on a typical trajectory for this approach to work.

3. **Improving destination prediction by sub-clustering on different properties of vehicle trajectories**

   Destination Prediction by Trajectory Sub-Clustering (DPTS) was proposed in Chapter 6, as an extension to Besse *et al.*'s Destination Prediction method (BDP). DPTS aims to reduce the distance error between the actual and predicted destinations, particularly at the beginning of the journey. Additionally, we introduce a decision threshold to be able to alternate between the predictions provided by DPTS and BDP throughout an unfolding trajectory, depending on the estimated performance. In our evaluation, we applied DPTS to three different datasets. The

first dataset, the Caltrain dataset saw DPTS give a significant reduction in distance error compared to BDP over the first half of the journey when two iterations of clustering are used. However, when applied to the Porto dataset we observed minimal improvement using DPTS compared to BDP. Finally, when applied to the POLD, which, unlike the other datasets, does not have a single starting location, DPTS yielded a notable reduction in distance error compared to BDP, throughout the entire journey. The main limitation of DPTS is that a single set of parameter values is not suitable across all datasets, especially with respect to the decision threshold. DPTS requires re-training and parameter values to be determined to suit the input data for a given application.

4. **Collecting two datasets containing vehicle trajectories with on-board vehicle data for point of interest and pattern of life experiments**

   In Chapter 3 we introduced two vehicle trajectory datasets, namely the Location Extraction Dataset (LED) and the Pattern of Life Dataset (POLD). The LED was created with multiple scripted scenarios, each containing a range of activities. Its purpose was to investigate PoI extraction, providing 22 vehicle signals, alongside GPS coordinates and a timestamp. The LED has been made publicly available, with the aim that it can be used for future research into vehicle PoI extraction. The POLD provides data for 5 drivers over selected periods, and serves as a basis for evaluation of PoI extraction in Chapter 4 and destination prediction in Chapters 5 and 6. There are no existing public datasets of this type due to the lack of access to vehicle signals and for privacy concerns. The limitations of the POLD are that the dataset has only a small number of participants, who represent a limited sample in terms of demographics, and there are a limited set of trajectories per participant.

## 7.2 Future Work

In this section, we present possibilities for future work in each of our proposed methods.

1. AVPE could be extended to make use of external data, such as data on nearby amenities, to provide the classifier more context on the current surroundings of a vehicle. For example, if a vehicle is consistently near a drive-through service for the duration of a PoI, this could be used to inform the prediction of the current activity.

2. Our method of using sequences of activity annotated locations could be

investigated on a larger dataset, to establish whether the small amount of data in our evaluation is an issue, in addition to considering an approach that uses both annotated and unannotated sequences in parallel. Additionally, the impact of the chosen grid size and threshold on the performance could be further evaluated. The chosen activity labels could also be reviewed, as the majority of activities within our trajectories were of the traffic type.

3. With respect to DPTS, future work could consider the decision threshold and investigate whether the parameter values can be removed, in order to make DPTS more generic across datasets. Additionally, further information from the time signal can be extracted to analyse the effect of seasonality and trends in user mobility, to see if these can aid performance. Different signals could be investigated, to see if other data can provide meaningful information for prediction, to use in another iteration of clustering.

## 7.3 Final Remarks

In this thesis, we have studied both PoI extraction and destination prediction from trajectories. Specifically, we have focused on the vehicular domain, and how data from on-board a vehicle can be utilised. In Chapter 4 we proposed our method for classifying the activity of a vehicle during periods of low mobility and using that classification to filter out false PoIs. In Chapter 5 we used this proposed activity classification method and applied it to the task of destination prediction. Finally, in Chapter 6, we showed that by using additional properties of a vehicle trajectory, we could cluster these trajectories into smaller groupings to reduce the error when performing destination prediction. We created two new vehicular datasets to help evaluate our work on PoI extraction and destination prediction, in addition to evaluating DPTS on two existing taxi datasets.

We have shown that using on-board vehicle data can accurately predict the activity of the vehicle, providing extra context for applications such as destination prediction. Although the performance of using annotated activities to prediction destinations was poor, clustering trajectories on different properties of a trajectory yielded promising results, and gives a method to discover other useful attributes contained within trajectories.

# Appendix A

# Activity Labelling Transitions

In this section, we present the transition tables from a specific activity to the following activity. The defined transitions are composed of both qualitative and quantitative that formalise the start and end instances for each activity, as introduced in Chapter 3. Where activities are missing from the next activity column, this implies that there is no defined transition e.g., in Table A.2 there is no transition from a drive through to a parked activity, as this would not occur.

Table A.1: Transition table from the Barrier activity to the next activity.

| **Next activity** | **Criteria** |
|---|---|
| Driving | When any part of the vehicle has past the barrier and the vehicle speed first increases to greater than 5km/h. |
| Traffic | When any part of the vehicle has past the barrier, immediately encountering congestion or road infrastructure that causes 5km/h not to be reached within 5 seconds. |

Table A.2: Transition table from the Drive Through activity to the next activity.

| **Next activity** | **Criteria** |
|---|---|
| Driving | When the car first moves away from the final booth (or collection point) and the vehicle speed first increases to greater than 5km/h. |
| Traffic | When the car first moves away from the final booth (or collection point), immediately encountering congestion or road infrastructure that causes 5km/h not to be reached within 5 seconds. |

Table A.3: 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 1km/h. |
| Drive-through | When the first booth (or order point) is reached and the vehicle speed first falls below 1km/h. |
| Drop-off | When the vehicle speed first falls below 1km/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 activity to be true. |
| Manoeuvre | When the vehicle speed first falls below 1km/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 1km/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 1km/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 activity to be true. |
| Traffic | When the vehicle speed first falls below 5km/h as a consequence of encountering congestion or road infrastructure that causes either 5km/h not to be reached within 10 seconds or the vehicle speed to fall below 1km/h within 10 seconds. |

Table A.4: Transition table from the Drop-off activity to the next activity.

| **Next activity** | **Criteria** |
|---|---|
| Driving | When the vehicle speed first increases to greater than 5km/h, after the vehicle doors have been closed. |
| Manoeuvre | When the car first moves away from the drop-off location (after which no doors are subsequently opened) or reverse gear is selected, whichever occurs first after manual inspection of a manoeuvre has confirmed the activity. |
| Traffic | When the car first moves away from the drop-off location (after which no doors are subsequently opened), immediately encountering congestion or road infrastructure that causes 5km/h not to be reached within 5 seconds. |

Table A.5: Transition table from the Manoeuvre 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 1km/h (without a further increase in speed). |
| Driving | When the vehicle speed first increases to greater than 5km/h after manual verification that the manoeuvre is finished. |
| Drop-off | When the vehicle speed first falls below 1km/h (without a further increase in speed) preceding a door to the vehicle being opened. Manual verification that a passenger is exiting the vehicle is required (from dash cam or seatbelt signals). The vehicle cannot be turned off for this activity to be true. |
| Parked | When either the vehicle speed first falls below 1km/h and subsequently stops or (if the vehicle is in reverse gear and below 1km/h) the first point when not in reverse. 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. If you are in parked this transition is true (this supersedes all other transitions if true). |
| Pick-up | When the vehicle speed first falls below 1km/h (without a further increase in speed) preceding a door to the vehicle being opened. Manual verification that a passenger is exiting the vehicle is required (from dash cam or seatbelt signals). The vehicle cannot be turned off for this activity to be true. |
| Traffic | When the vehicle speed first increases to greater than 1km/h, after manual verification that the manoeuvre has finished, immediately encountering congestion or road infrastructure that causes 5km/h not to be reached within 5 seconds. |

Table A.6: Transition table from the Parked activity to the next activity.

| **Next activity** | **Criteria** |
|---|---|
| Driving | When the vehicle speed first increases to greater than 1km/h. |
| Manoeuvre | When the vehicle speed first increases greater than 1km/h or reverse gear is selected, whichever occurs first after manual inspection of a manoeuvre has confirmed the activity. This must include a stop (or use reverse). |
| Traffic | When the vehicle speed first increases to greater than 1km/h, immediately encountering congestion or road infrastructure that causes 5km/h not to be reached within 5 seconds. |

Table A.7: Transition table from the Pick-up activity to the next activity.

| **Next activity** | **Criteria** |
|---|---|
| Driving | When the vehicle speed first increases to greater than 5km/h, after the vehicle doors have been closed. |
| Manoeuvre | When the car first moves away from the pick-up location (after which no doors are subsequently opened) or reverse gear is selected, whichever occurs first after manual inspection of a manoeuvre has confirmed the activity. |
| Traffic | When the car first moves away from the pick-up location (after which no doors are subsequently opened), immediately encountering congestion or road infrastructure that causes 5km/h not to be reached within 5 seconds. |

Table A.8: Transition table from the Traffic activity to the next activity.

| **Next activity** | **Criteria** |
|---|---|
| Barrier | When the vehicle first reaches the barrier, without any vehicle between itself and the barrier, where the vehicle speed first falls below 1km/h. |
| Drive-through | When the first booth (or order point) is reached and the vehicle speed first falls below 1km/h resulting in a stop. |
| Driving | When the vehicle speed first increases to greater than 15km/h. If the road infrastructure prevents 15km/h from being reached in normal driving (within 10 seconds of normal driving), transition when the vehicle speed first increases to greater than 10km/h. |
| Drop-off | When the vehicle speed first falls below 1km/h (without a further increase in speed) preceding a door to the vehicle being opened. Manual verification that a passenger is exiting the vehicle is required (from dash cam or seatbelt signals). The vehicle cannot be turned off for this activity to be true. |
| Manoeuvre | When reverse gear is selected or the steering angle exceeds a difference 10 degrees from the previous reading, whichever occurs first after manual inspection of a manoeuvre has confirmed the activity. |
| Parked | When the vehicle speed first falls below 1km/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 1km/h (without a further increase in speed) preceding a door to the vehicle being opened. Manual verification that a passenger is entering the vehicle is required (from dash cam or seatbelt signals). The vehicle cannot be turned off for this activity to be true. |

# Appendix B

# Location Extraction Dataset Route Maps

In this section, we present the maps for each route that was part of the Location Extraction Dataset (LED). Descriptions of these routes, along with the distribution of activities are detailed in Chapter 3.


Figure B.1: Route 1.

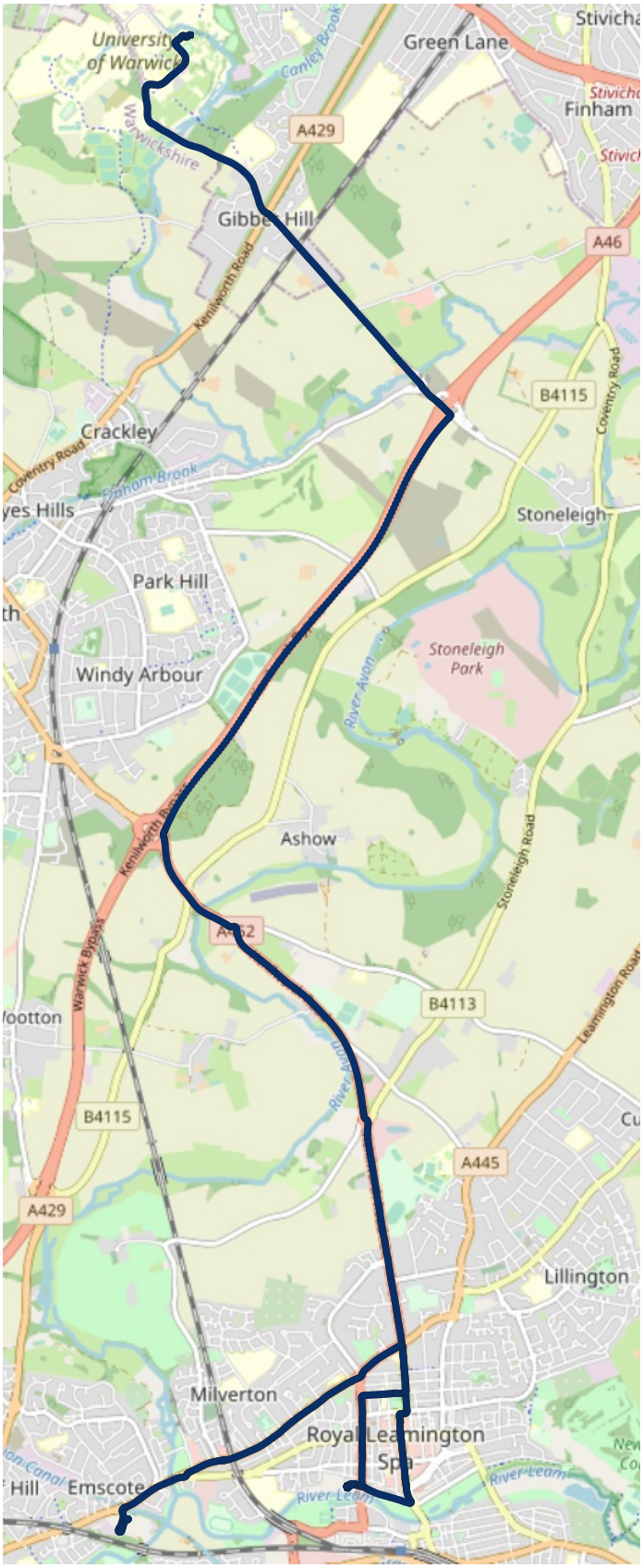



Figure B.2: Route 1 (Estate variant).

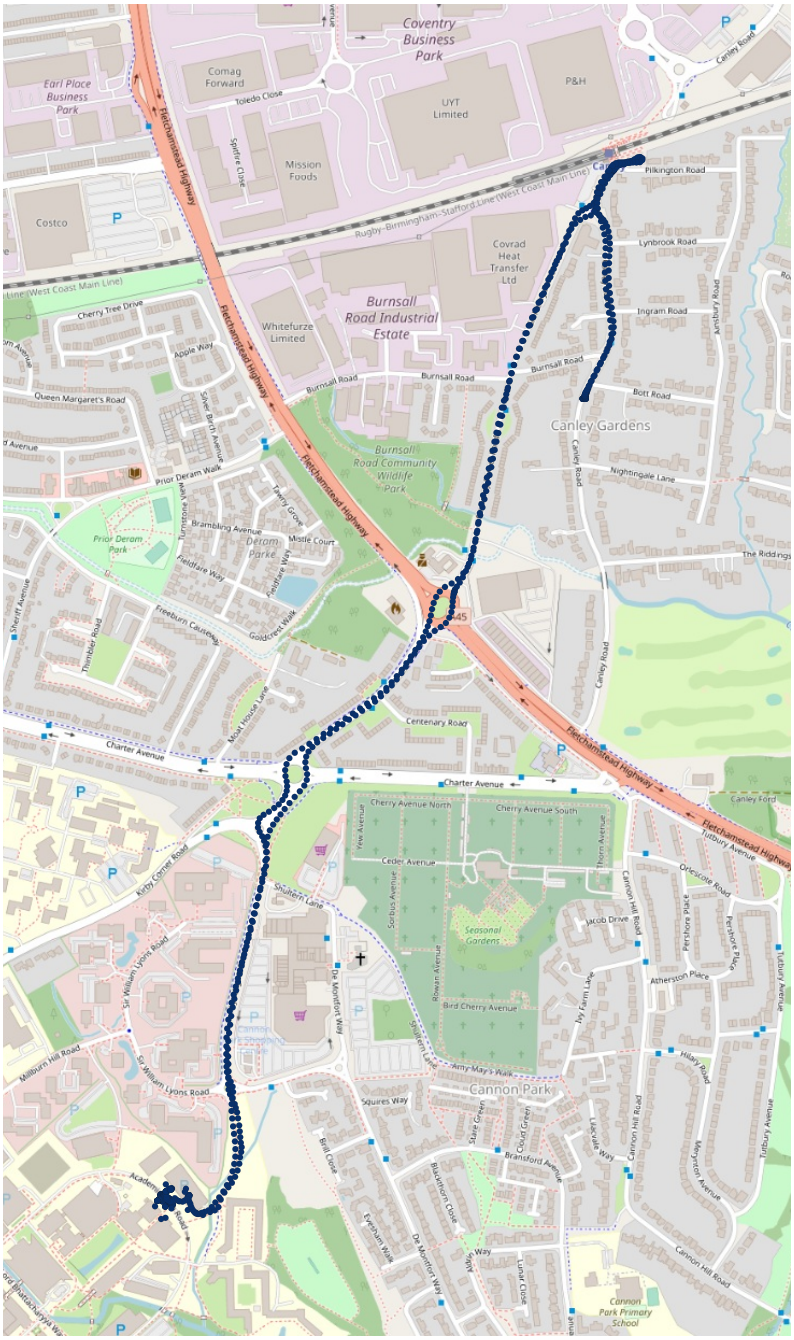



Figure B.3: Route 2.

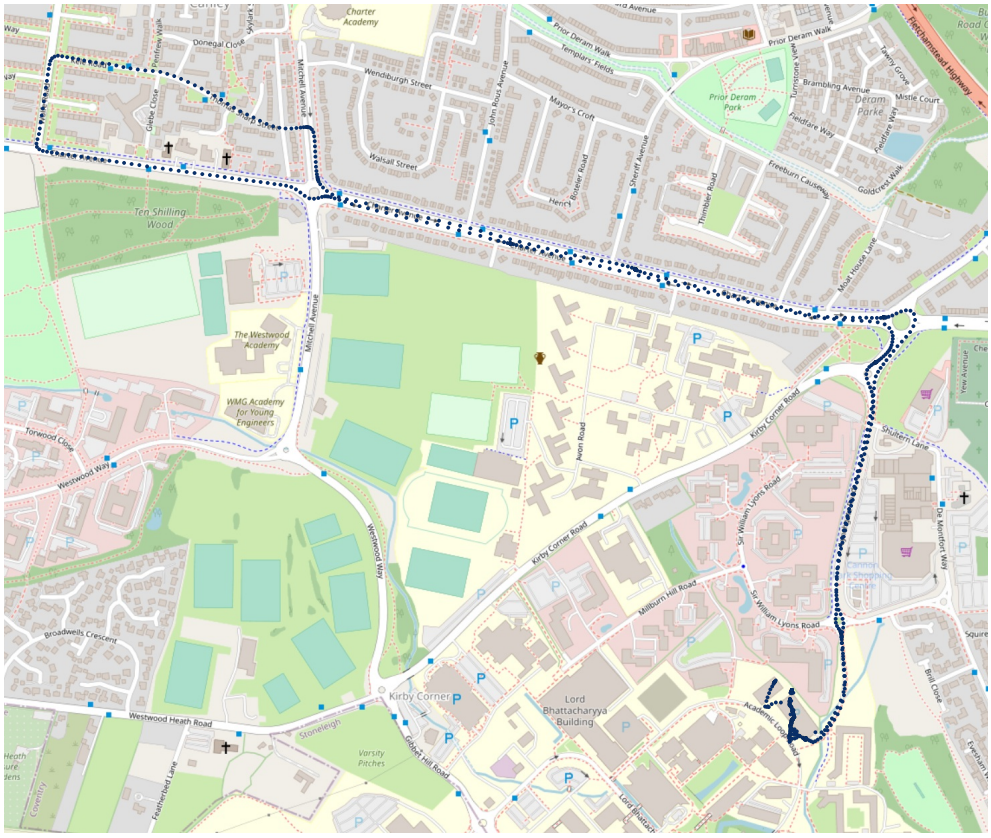



Figure B.4: Route 3.

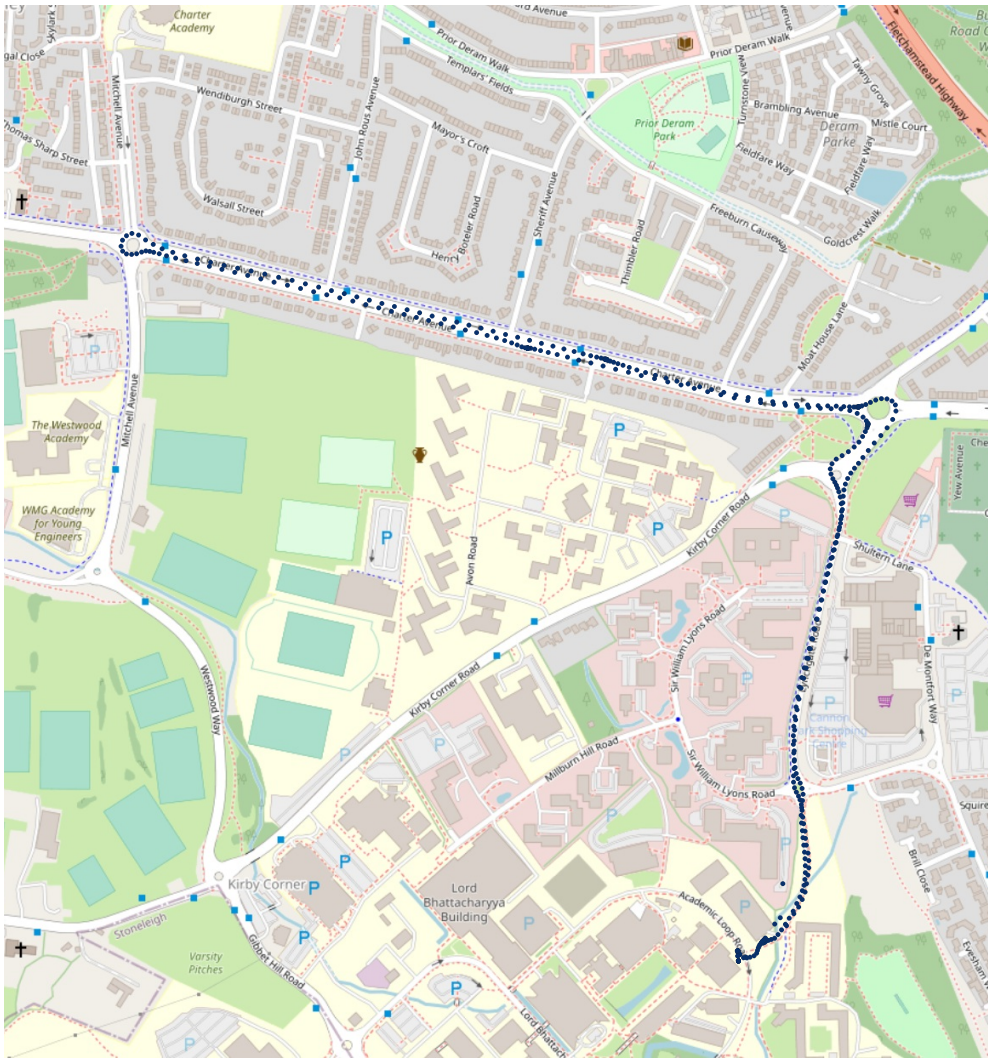



Figure B.5: Route 3 (Estate variant).

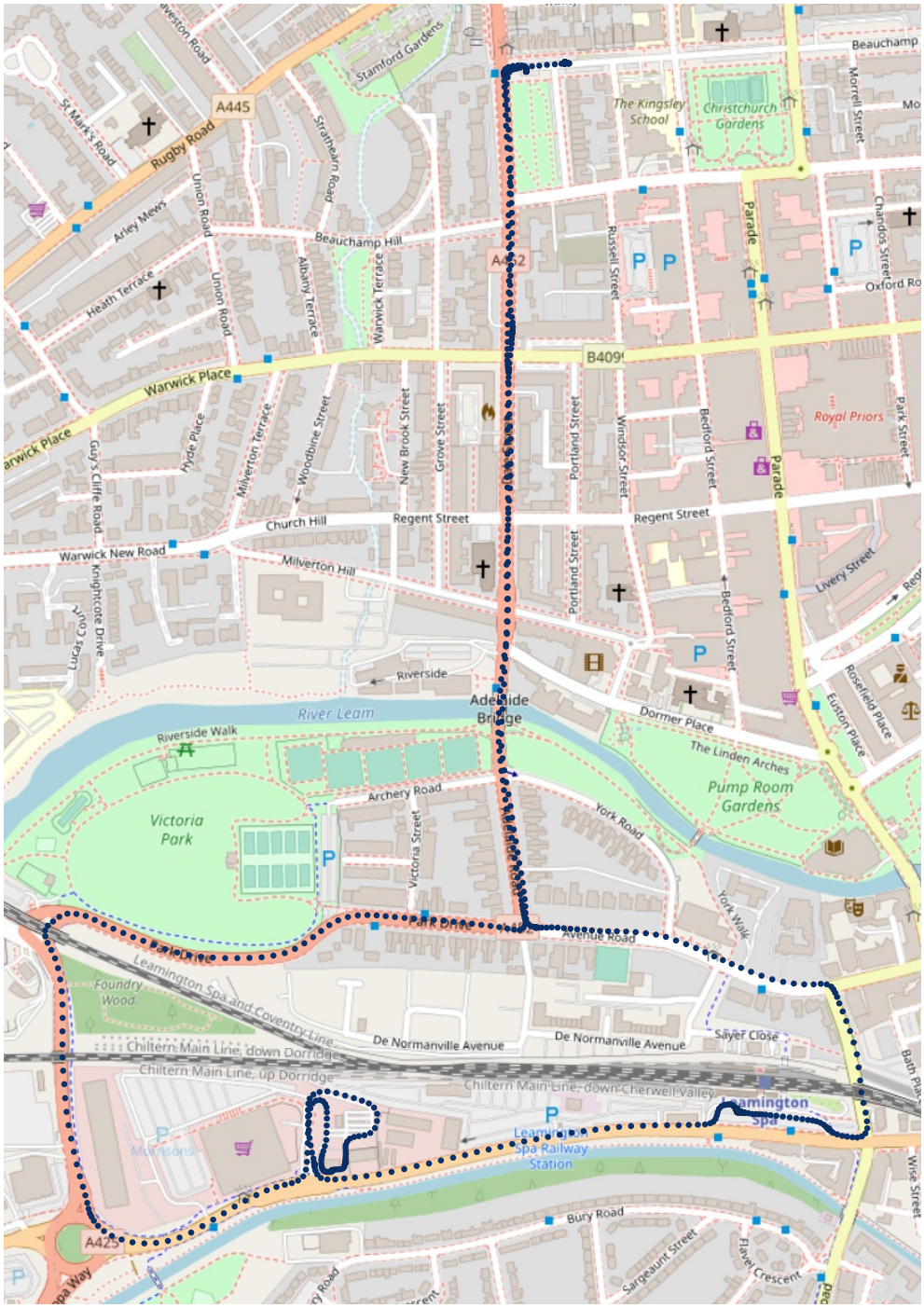



Figure B.6: Route 4.


Figure B.7: Route 5.


Figure B.8: Route 5 (Estate variant).

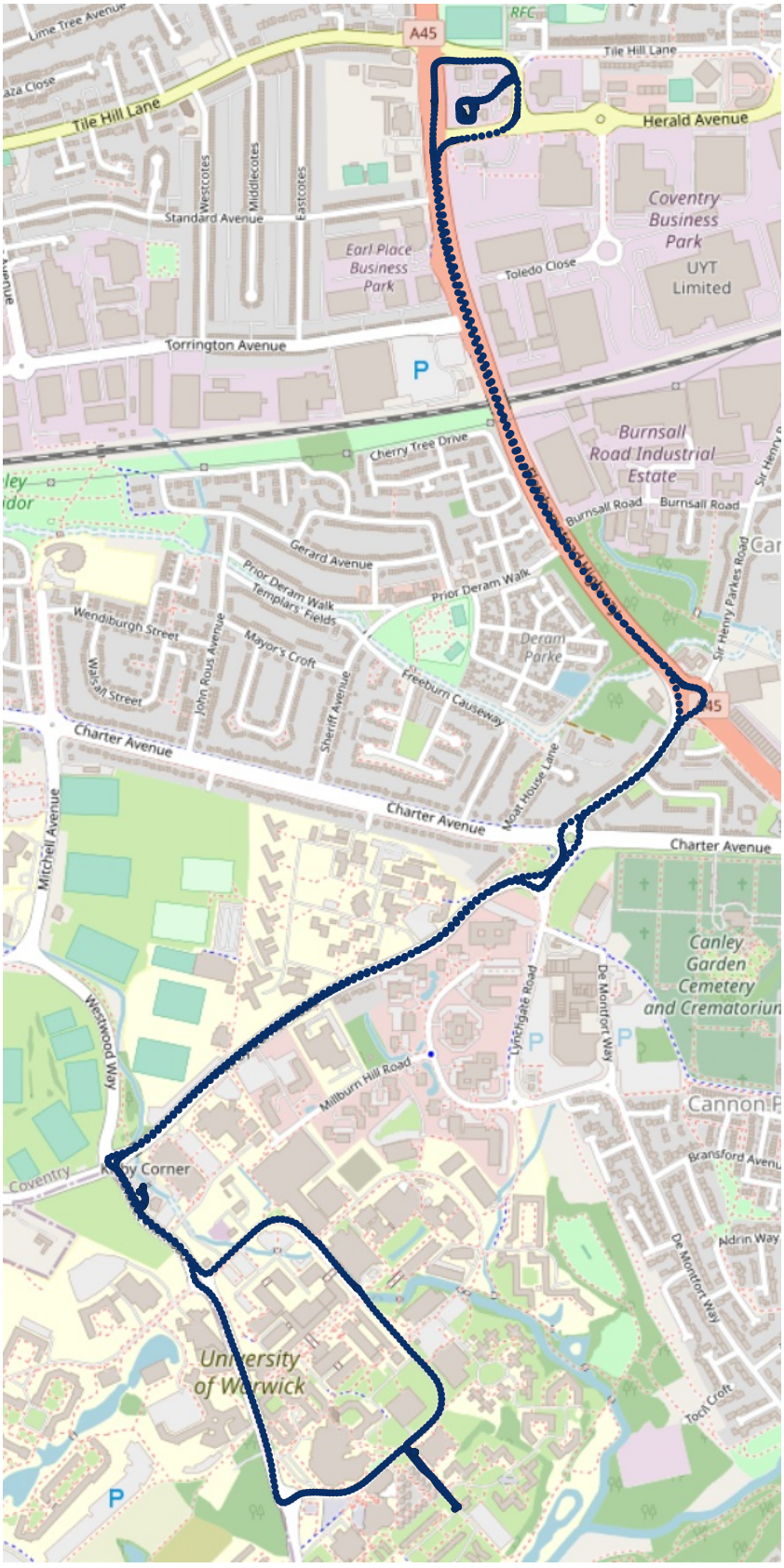



Figure B.9: Route 6.

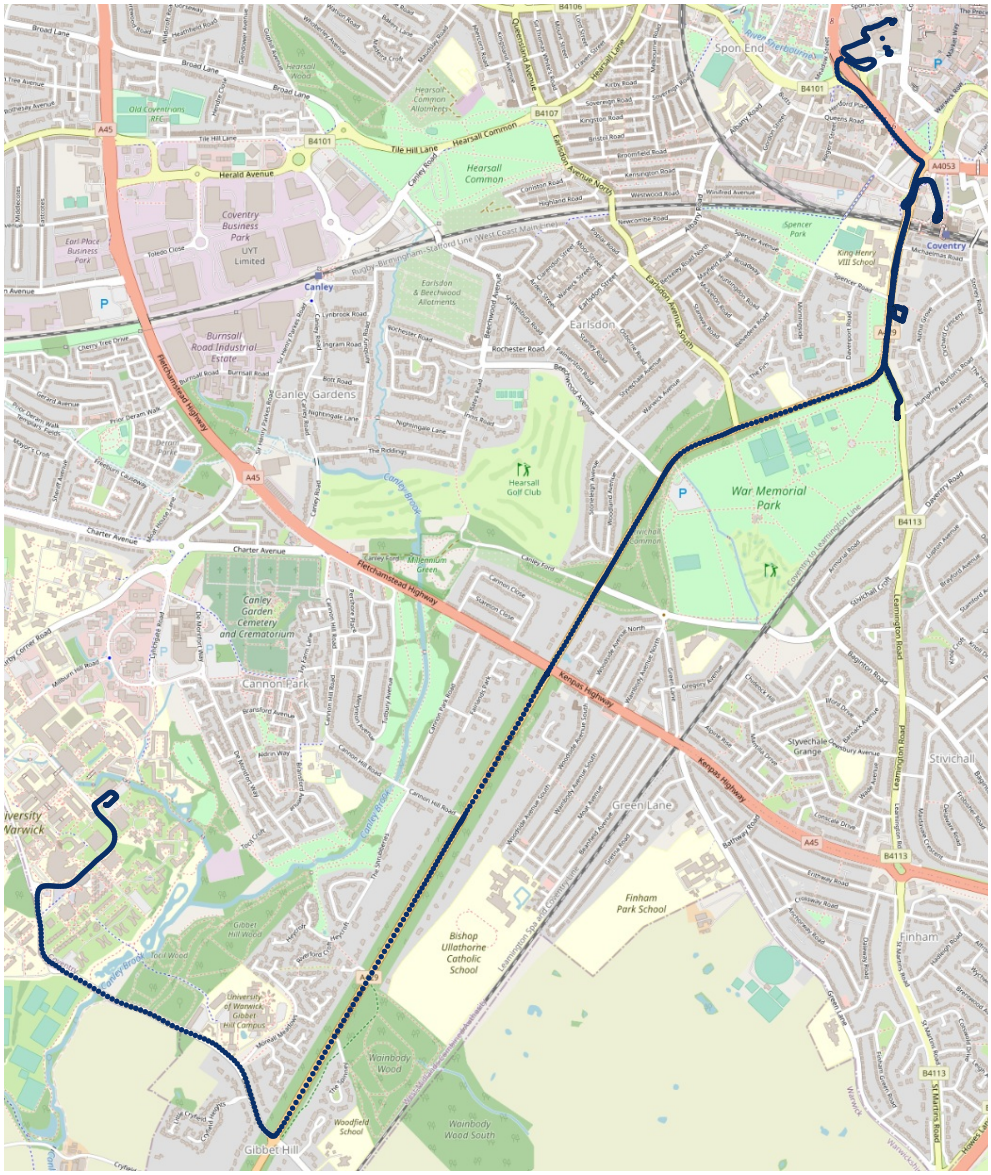



Figure B.10: Route 7.

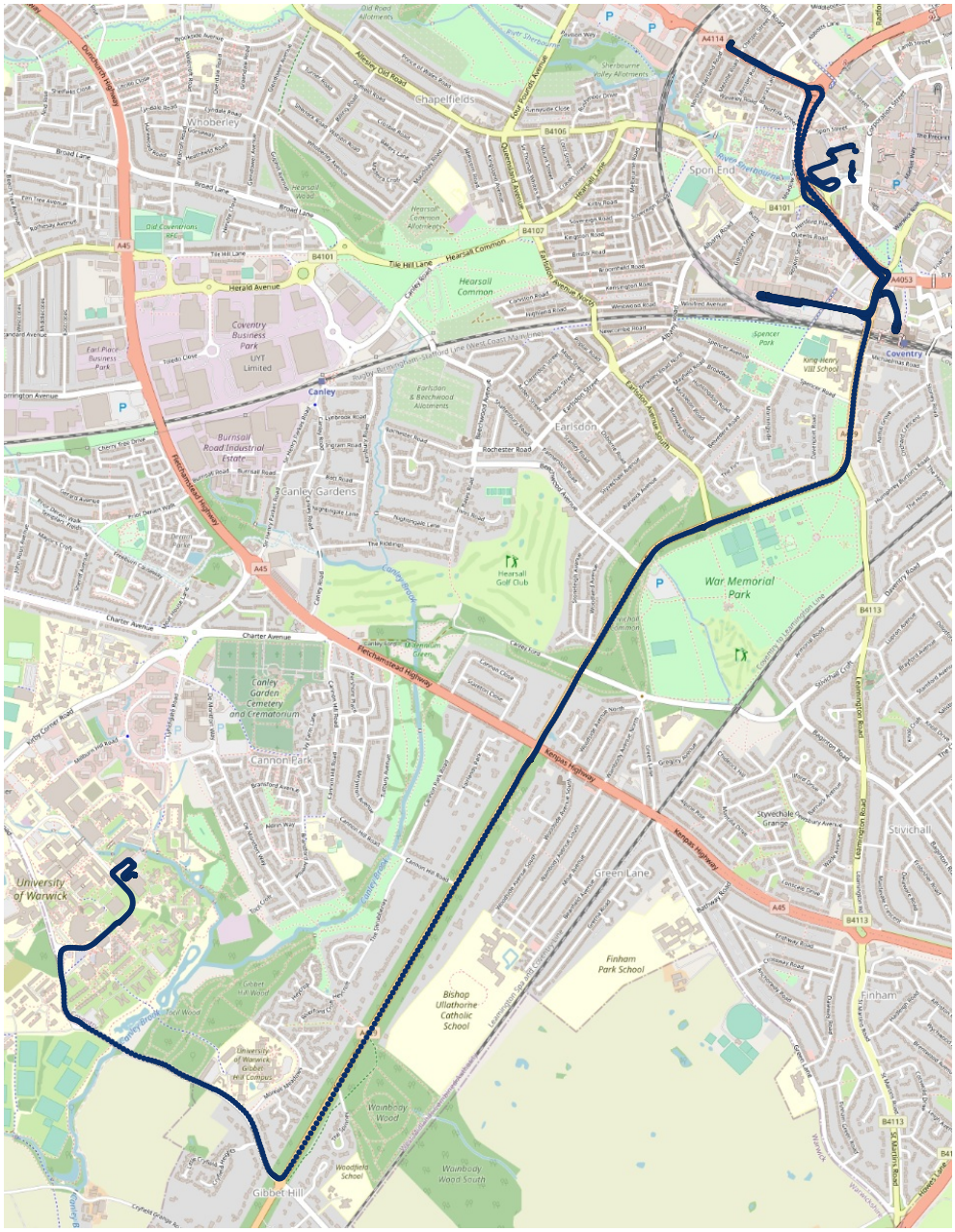



Figure B.11: Route 7 (Estate variant).

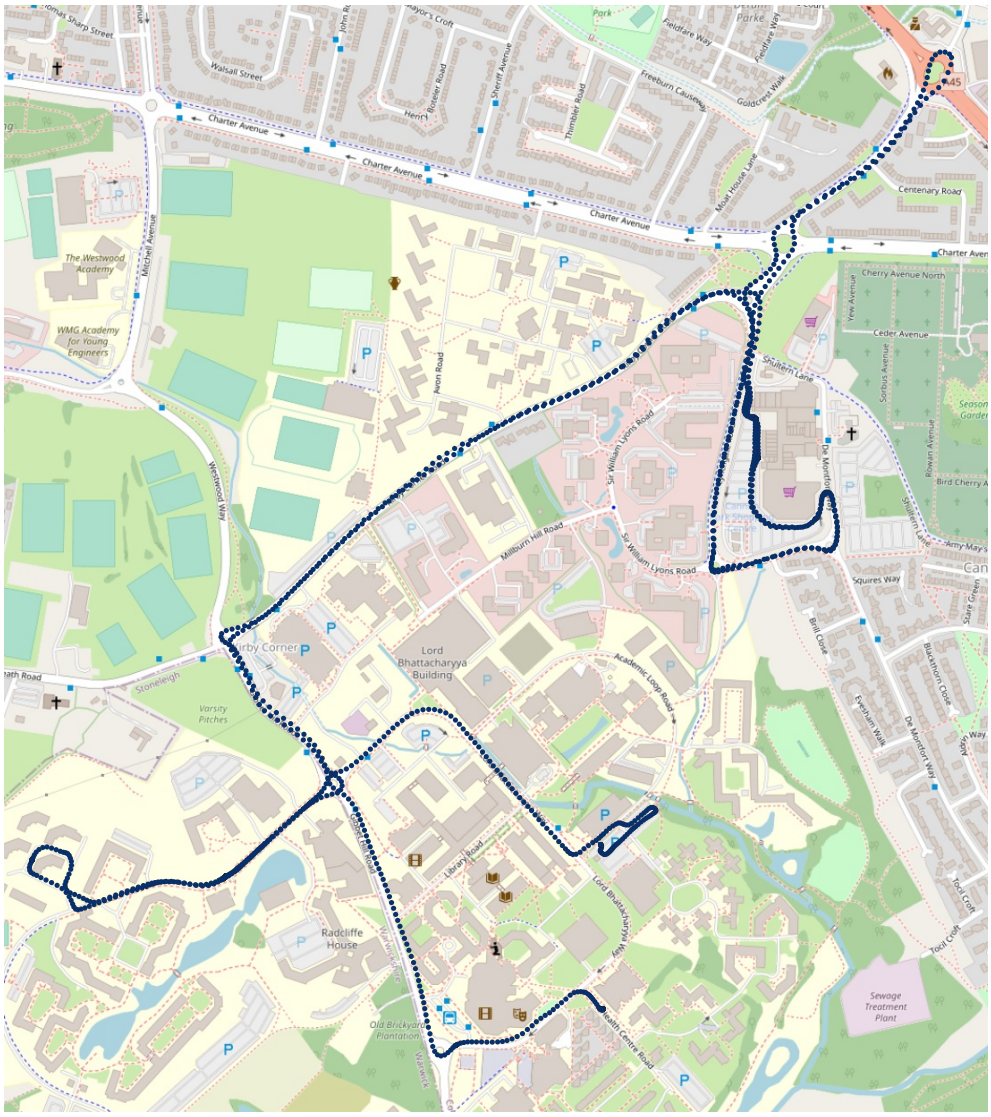



Figure B.12: Route 8.

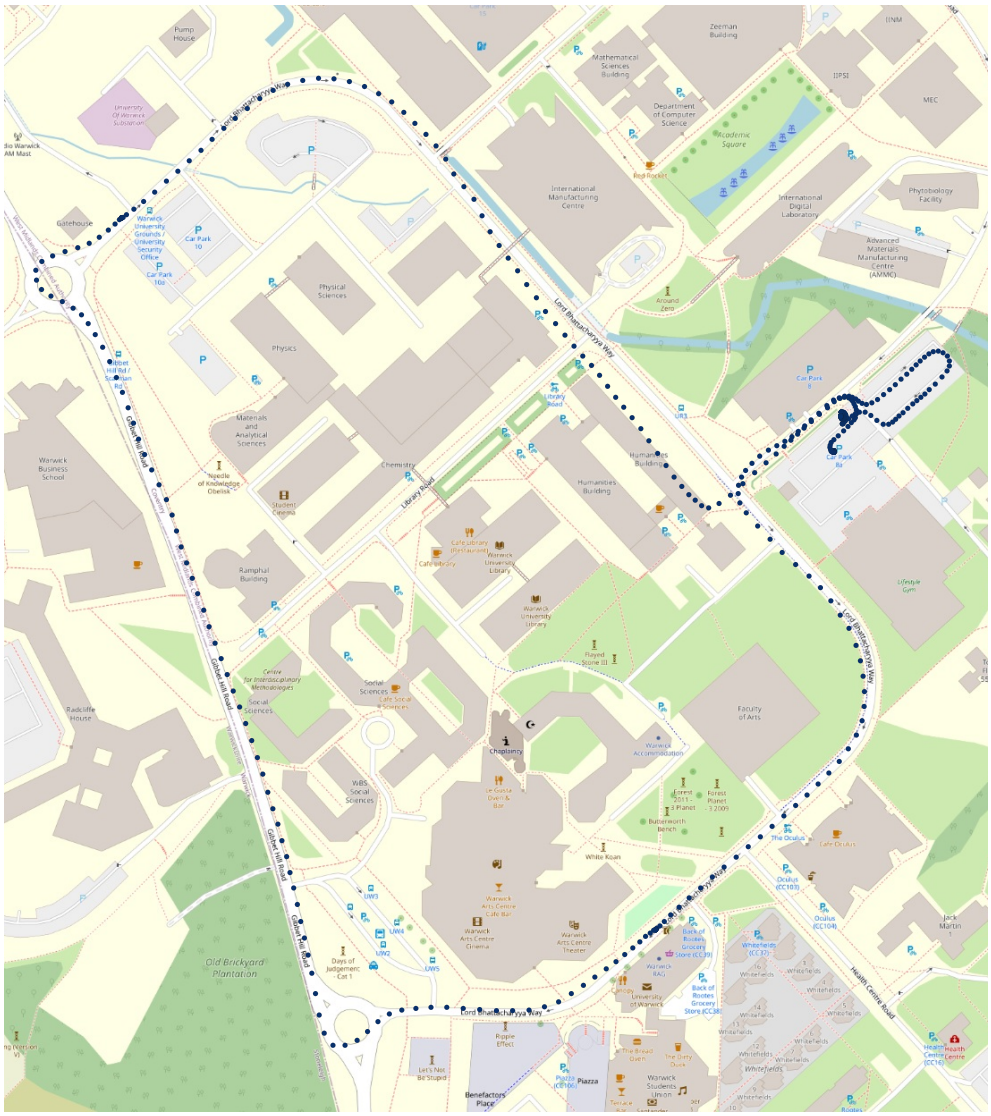



Figure B.13: Route 9.

# Bibliography


[1] Kaggle data set ecml/pkdd 15: Taxi trajectory prediction (1). Downloaded from `https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/data`, April 2015.

[2] Rethink data: Put more of your business data to work - from edge to cloud. Technical report, Seagate Technology, Cupertino, California, United States, 2020.

[3] Mohammed Abdalla, Abdeltawab Hendawi, Neveen ElGamal, Hoda M. O. Mokhtar, Mohamed Ali, and John Krumm. Similarmove: Similarity-based prediction for moving object future path. In *Proceedings of the 2nd ACM SIGSPATIAL Workshop on Prediction of Human Mobility*, pages 15—24, 2018.

[4] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

[5] Jeffrey L. Adler and Victor J. Blue. A cooperative multi-agent transportation management and route guidance system. *Transportation Research Part C: Emerging Technologies*, 10(5-6):433–454, 2002.

[6] Jeffrey L. Adler, Goutam Satapathy, Vikram Manikonda, Betty Bowles, and Victor J. Blue. A multi-agent approach to cooperative traffic management and route guidance. *Transportation Research Part B: Methodological*, 39(4):297–318, May 2005.

[7] Assad Al Alam, Ather Gattami, and Karl H. Johansson. An experimental study on the fuel reduction potential of heavy duty vehicle platooning. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*. IEEE, 2010.

[8] Andreas Allström, Ida Kristoffersson, and Yusak Susilo. Smartphone based travel diary collection: Experiences from a field trial in stockholm. *Transportation research procedia*, 26:32–38, 2017.

[9] Juan A. Alvarez-Garcia, Juan A. Ortega, Luis Gonzalez-Abril, and Francisco Velasco. Trip destination prediction based on past gps log using a hidden markov model. *Expert Systems with Applications*, 37(12): 8166–8171, December 2010.

[10] Mani Amoozadeh, Hui Deng, Chen-Nee Chuah, Michael Zhang, and Dipak Ghosal. Platoon management with cooperative adaptive cruise control enabled by VANET. *Vehicular Communications*, 2(2):110–123, 2015.

[11] Thiago Andrade and João Gama. Identifying points of interest and similar individuals from raw gps data. In *Mobility Internet of Things*, pages 293–305, 2020.

[12] Alex Angove-Plumb. How and why twitter tracks your location data, September 2021. URL `https://www.choice.com.au/consumers-and-data/data-collection-and-use/who-has-your-data/articles/how-and-why-twitter-tracks-your-location`. [Online; posted 20-September-2021].

[13] Behrang Asadi and Ardalan Vahidi. Predictive Cruise Control: Utilizing Upcoming Traffic Signal Information for Improving Fuel Economy and Reducing Trip Time. *IEEE Transactions on Control Systems Technology*, 19(3):707–714, May 2011.

[14] Daniel Ashbrook and Thad Starner. Learning significant locations and predicting user movement with gps. In *Proc. of the International Symposium on Wearable Computers*, pages 101–108, 2002.

[15] Daniel Ashbrook and Thad Starner. Using gps to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5):275–286, October 2003.

[16] Roland Assam and Thomas Seidl. Bodyguards: A clairvoyant location predictor using frequent neighbors and markov model. In *Proc. of the IEEE Conference on Ubiquitous Intelligence and Computing and IEEE Conference on Autonomic and Trusted Computing*, pages 25–32, 2013.

[17] Akin Avci, Stephan Bosch, Mihai Marin-Perianu, Raluca Marin-Perianu, and Paul Havinga. Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey. In *Proc. of the International Conference on Architecture of Computing Systems*, pages 1–10, 2010.

[18] Athanasios Bamis and Andreas Savvides. Lightweight Extraction of Frequent Spatio-Temporal Activities from GPS Traces. In *Proc. of the IEEE Real-Time Systems Symposium*, pages 281–291, 2010.

[19] Ling Bao and Stephen S. Intille. Activity recognition from user-annotated acceleration data. In *Pervasive Computing*, pages 1–17, 2004.

[20] Leonard E. Baum and Ted Petrie. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, December 1966.

[21] Ramon Bauza and Javier Gozalvez. Traffic congestion detection in large-scale scenarios using vehicle-to-vehicle communications. *Journal of Network and Computer Applications*, 36(5):1295–1307, September 2013.

[22] Philippe C. Besse, Brendan Guillouet, Jean-Michel Loubes, and Francǫis Royer. Review and perspective for distance-based clustering of vehicle trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 17(11):3306–3317, 2016.

[23] Philippe C. Besse, Brendan Guillouet, Jean-Michel Loubes, and Francǫis Royer. Destination prediction by trajectory distribution-based model. *IEEE Transactions on Intelligent Transportation Systems*, 19(8):2470–2481, 2018.

[24] Tanusri Bhattacharya, Lars Kulik, and James Bailey. Extracting significant places from mobile user GPS trajectories: A bearing change based approach. In *Proc. of the International Conference on Advances in Geographic Information Systems*, pages 398–401, 2012.

[25] Tanusri Bhattacharya, Lars Kulik, and James Bailey. Automatically recognizing places of interest from unreliable GPS data using spatio-temporal density estimation and line intersections. *Pervasive and Mobile Computing*, 19:86–107, May 2015.

[26] Nicola Bicocchi and Marco Mamei. Investigating ride sharing opportunities through mobility data analysis. *Pervasive and Mobile Computing*, 14:83–94, October 2014.

[27] Derya Birant and Alp Kut. ST-DBSCAN: An Algorithm for Clustering Spatial-Temporal Data. *Data & Knowledge Engineering*, 60(1):208–221, January 2007.

[28] Fabian Bock, Sergio Di Martino, and Monika Sester. Data-driven approaches for smart parking. In *Proc. of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 358–362, 2017.

[29] Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7): 1145–1159, 1997.

[30] Dirk Brockmann, Lars Hufnagel, and Theo Geisel. The scaling laws of human travel. *Nature*, 439:462–465, 2006.

[31] Alberto Broggi, Alex Zelinsky, Ümit Özgüner, and Christian Laugier. *Intelligent Vehicles*, pages 1627–1656. Springer International Publishing, 2016.

[32] D Brück, H Elmqvist, S E Mattsson, and H Olsson. Dymola for multi-engineering modeling and simulation. In *Proc. of the International Modelica Conference*, 2002.

[33] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 46(3):33:1–33:33, January 2014.

[34] Sebastian Büscher, Manuel Batram, and Dietmar Bauer. Using motifs for population synthesis in multi-agent mobility simulation models. In *Proc. of the Workshop on Stochastic Models, Statistics and their Application*, pages 335–349, 2019.

[35] Vadim A Butakov and Petros Ioannou. Personalized Driver Assistance for Signalized Intersections Using V2I Communication. *IEEE Transactions on Intelligent Transportation Systems*, 17(7):1910–1919, January 2016.

[36] Felix Caicedo, Carola Blazquez, and Pablo Miranda. Prediction of parking space availability in real time. *Expert Systems with Applications*, 39(8): 7281–7290, 2012.

[37] Marc Cañigueral and Joaquim Meléndez. Flexibility management of electric vehicles based on user profiles: The arnhem case study. *International Journal of Electrical Power & Energy Systems*, 133(1):1–17, 2021.

[38] Xin Cao, Gao Cong, and Christian S. Jensen. Mining significant semantic locations from GPS data. *Proc. of the VLDB Endowment*, 3(1–2):1009–1020, September 2010.

[39] Joe Castiglione, Mark Bradley, and John Gliebe. *Activity-based travel demand models: A primer*. Number SHRP 2 Report S2-C46-RR-1. The National Academies Press, 2015.

[40] Manoel Castro-Neto, Young-Seon Jeong, Myong-Kee Jeong, and Lee D. Han. Online-svr for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert Systems with Applications*, 36(3, Part 2):6164–6173, 2009.

[41] Ling Chen, Mingqi Lv, and Gencai Chen. A system for destination and future route prediction based on trajectory mining. *Pervasive and Mobile Computing*, 6(6):657–676, December 2010.

[42] Ling Chen, Mingqi Lv, Qian Ye, Gencai Chen, and John Woodward. A personal route prediction system based on trajectory data mining. *Information Sciences*, 181(7):1264–1284, April 2011.

[43] Yunbo Chen, Hongchu Yu, and Lei Chen. A spatiotemporal cluster method for trajectory data. In *Proc. of the International Conference on Computer Engineering and Networks*, pages 3–10, 2015.

[44] Sung-Bae Cho. Exploiting machine learning techniques for location recognition and prediction with smartphone logs. *Neurocomputing*, 176: 98–106, February 2016.

[45] Patrick Pakyan Choi and Martial Hebert. Learning and Predicting Moving Object Trajectory: A Piecewise Trajectory Segment Approach. *Robotics Institute*, 337:1–17, 2006.

[46] Chung-Hua Chu and Se-Hsien Wu. A chinese restaurant recommendation system based on mobile context-aware services. In *Proc. of the International Conference on Mobile Data Management*, pages 116–118, 2013.

[47] Blerim Cici, Athina Markopoulou, Enrique Frias-Martinez, and Nikolaos Laoutaris. Assessing the potential of ride-sharing using mobile and social data: A tale of four cities. In *Proc. of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 201—-211, 2014.

[48] Naznin Sultana Daisy, Mohammad Hesam Hafezi, Lei Liu, and Hugh Millward. Understanding and modeling the activity-travel behavior of university commuters at a large canadian university. *Journal of Urban Planning and Development*, 144, 2018.

[49] Maria Luisa Damiani, Hamza Issa, and Francesca Cagnacci. Extracting stay regions with uncertain boundaries from gps trajectories: A case study in animal ecology. In *Proc. of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 253—-262, 2014.

[50] Cedric De Cauwer, Wouter Verbeke, Joeri Van Mierlo, and Thierry Coosemans. A model for range estimation and energy-efficient routing of electric vehicles in real-world conditions. *IEEE Transactions on Intelligent Transportation Systems*, 21(7):2787–2800, July 2020.

[51] Allan M. de Souza, Roberto S. Yokoyama, Guilherme Maia, and Leandro Villas. Real-time path planning to prevent traffic jam through an intelligent transportation system. *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 726–731, 2016.

[52] Erni Dianawati and Suhono Harso Supangkat. Designing on-street smart parking recommendations. In *Proc. of the International Conference on ICT for Smart Society (ICISS)*, pages 1–5, 2020.

[53] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26:297–302, 1945.

[54] Trinh Minh Tri Do and Daniel Gatica-Perez. Where and what: Using smartphones to predict next locations and applications in daily life. *Pervasive and Mobile Computing*, 12:79–91, June 2014.

[55] Miriam Enzi, Sophie N Parragh, David Pisinger, and Matthias Prandtstetter. Modeling and solving the multimodal car- and ride-sharing problem. *European Journal of Operational Research*, 293(1):290–303, August 2021.

[56] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of the International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[57] Adam Evans, Andrew Kelly, and Matthew Slocombe. National travel survey: England 2018, 2019.

[58] M. Farsi, K. Ratcliff, and Manuel B. Barbosa. An overview of controller area network. *Computing & Control Engineering Journal*, 10(3):113–120, 1999.

[59] João C. Ferreira, Vitor Monteiro, and Afonso Jo ao L. Dynamic range prediction for an electric vehicle. In *Proc. of the World Electric Vehicle Symposium and Exhibition*, pages 1–11, 2013.

[60] Peter A Flach and Nicolas Lachiche. Naive bayesian classification of structured data. *Machine Learning*, 57(3):233–269, 2004.

[61] Zhongliang Fu, Zongshun Tian, Yanqing Xu, and Changjian Qiao. A two-step clustering approach to extract locations from individual GPS trajectory data. *ISPRS International Journal of Geo-Information*, 5(10): 166:1–166:17, September 2016.

[62] Jun Fukano, Tomohiro Mashita, Takahiro Hara, Kiyoshi Kiyokawa, Haruo Takemura, and Shojiro Nishio. A next location prediction method for

smartphones using blockmodels. In *Proc. of the IEEE Conference on Virtual Reality (VR)*, pages 1–4, 2013.

[63] Barbara Furletti, Paolo Cintia, Chiara Renso, and Laura Spinsanti. Inferring human activities from GPS tracks. In *Proc. of the ACM SIGKDD International Workshop on Urban Computing*, pages 5:1–5:8, 2013.

[64] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. Next place prediction using mobility markov chains. In *Proc. of the Workshop on Measurement, Privacy, and Mobility*, pages 1–6, 2012.

[65] Yu Gong, Yong Li, Depeng Jin, Li Su, and Lieguang Zeng. A location prediction scheme based on social correlation. In *Proc. of the IEEE Vehicular Technology Conference (VTC Spring)*, pages 1–5, 2011.

[66] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453:779–782, 2008.

[67] Tao Gu, Zhanqing Wu, Xianping Tao, Hung Keng Pung, and Jian Lu. epSICAR: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition. In *Proc. of the International Conference on Pervasive Computing and Communications*, pages 1–9. IEEE, 2009.

[68] Hendrik-Jorn Gunther, Sandra Kleinau, Oliver Trauer, and Lars Wolf. Platooning at traffic lights. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 1047–1053. Volkswagen AG, Wolfsburg, Germany, IEEE, 2016.

[69] Hong Guo, Jacob Crossman, Yi L. Murphey, and Mark Coleman. Automotive signal diagnostics using wavelets and machine learning. *IEEE Transactions on Vehicular Technology*, 49(5):1650–1662, 2000.

[70] Anant Gupta and Kuldeep Singh. Location based personalized restaurant recommendation system for mobile environments. In *Proc. of the International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 507–511, 2013.

[71] Rula Amjad Hamid and Muayad Sadik Croock. A developed gps trajectories data management system for predicting tourists' poi. *Telkomnika*, 18(1):124–132, July 2019.

[72] Peter Hank, Steffen Müller, Ovidiu Vermesan, and Jeroen Van Den Keybus. Automotive ethernet: In-vehicle networking and smart mobility. In *Proc. of the Design, Automation & Test in Europe Conference (DATE)*, pages 1735–1739, 2013.

[73] Ramaswamy Hariharan and Kentaro Toyama. Project lachesis: Parsing and modeling location histories. In *Geographic Information Science*, pages 106–124, 2004.

[74] Beverly Harrison and Anind Dey. What have you done with location-based services lately? *IEEE Pervasive Computing*, 8(4):66–70, October 2009.

[75] J. A. Hartigan and M. A. Wong. Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[76] Abdeltawab M. Hendawi, Jie Bao, Mohamed F. Mokbel, and Mohamed Ali. Predictive tree: An efficient index for predictive queries on road networks. In *2015 IEEE 31st International Conference on Data Engineering*, pages 1215–1226, 2015.

[77] Xin Hong, Chris Nugent, Maurice Mulvenna, Sally McClean, Bryan Scotney, and Steven Devlin. Evidential fusion of sensor data for activity recognition in smart homes. *Pervasive and Mobile Computing*, 5(3): 236–252, June 2009.

[78] Gwo-Jiun Horng. Using cellular automata for parking recommendations in smart environments. *PLOS one*, 9(8):1–13, August 2014.

[79] Jie Hu, Shijie Cai, Tengfei Huang, Xiongzhen Qin, Zhangbin Gao, Liming Chen, and Yufeng Du. Vehicle travel destination prediction method based on multi-source data. *Automotive Innovation*, pages 1–13, March 2021.

[80] Tâm Huỳnh, Ulf Blanke, and Bernt Schiele. Scalable recognition of daily activities with wearable sensors. In *Proc. of the International Symposium on Location- and Context-Awareness)*, pages 50–67, 2007.

[81] J. W. Inman. *Navigation and Nautical Astronomy for the Use of British Seamen*. C. and J. Rivington, 1835.

[82] Shan Jiang, Gaston A. Fiore, Yingxiang Yang, Joseph Ferreira, Emilio Frazzoli, and Marta C. González. A review of urban computing for mobile phone traces: Current methods, challenges and opportunities. In *Proc. of the ACM SIGKDD International Workshop on Urban Computing*, pages 1–9, 2013.

[83] Shan Jiang, Joseph Ferreira, and Marta C Gonzalez. Activity-based human mobility patterns inferred from mobile phone data: A case study of singapore. *IEEE Transactions on Big Data*, 3:208–219, 2017.

[84] Md Abdus Samad Kamal, Shun Taguchi, and Takayoshi Yoshimura. Efficient Driving on Multilane Roads Under a Connected Vehicle Environment. *IEEE Transactions on Intelligent Transportation Systems*, 17 (9):2541, 2016.

[85] Kiran Kambly and Thomas H. Bradley. Geographical and temporal differences in electric vehicle range due to cabin conditioning energy consumption. *Journal of Power Sources*, 275(1):468–475, February 2015.

[86] Jong Hee Kang, William Welbourne, Benjamin Stewart, and Gaetano Borriello. Extracting places from traces of locations. In *Proc. of the ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*, pages 110–118, 2004.

[87] Hillol Kargupta, Ruchita Bhargava, Kun Liu, Michael Powers, Patrick Blair, Samuel Bushra, James Dull, Kakali Sarkar, Martin Klein, Mitesh Vasa, and David Handy. *VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring*, pages 300–311. 2004.

[88] Hassan A. Karimi and Xiong Liu. A predictive location model for location-based services. In *Proc. of the ACM International Symposium on Advances in Geographic Information Systems*, pages 126—-133, 2003.

[89] Rahul Katarya and Om P. Verma. Restaurant recommender system based on psychographic and demographic factors in mobile environment. In *Proc. of the International Conference on Green Computing and Internet of Things (ICGCIoT)*, pages 907–912, 2015.

[90] Ilkcan Keles, Matthias Schubert, Peer Kröger, Simonas Šaltenis, and Christian S. Jensen. Extracting visited points of interest from vehicle trajectories. In *Proc. of the International ACM Workshop on Managing and Mining Enriched Geo-Spatial Data*, pages 2:1–2:6, 2017.

[91] Eunju Kim, Sumi Helal, and Diane Cook. Human activity recognition and pattern discovery. *IEEE Pervasive Computing*, 9(1), 2010.

[92] Sanling Kim, Mark E. Lewis, and Chelsea C. White. Optimal Vehicle Routing With Real-Time Traffic Information. *IEEE Transactions on Intelligent Transportation Systems*, 6(2):178–188, June 2005.

[93] Scott Kirkpatrick, Daniel C. Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[94] Niko Kiukkonen, Jan Blom, Olivier Dousse, Daniel Gatica-Perez, and Juha Laurila. Towards rich mobile phone datasets: Lausanne data collection campaign. In *Proc. of the International Conference on Pervasive Services*, pages 1–7, 2010.

[95] Mohammad Javad Koohsari, Neville Owen, Rachel Cole, Suzanne Mavoa, Koichiro Oka, Tomoya Hanibuchi, and Takemi Sugiyama. Built environmental factors and adults' travel behaviors: role of street layout and local destinations. *Preventive medicine*, 96:124–128, 2017.

[96] Daniel Krajzewicz, Georg Hertkorn, C. Rössel, and Peter Wagner. SUMO (Simulation of Urban MObility) - an open-source traffic simulation. In *Proc. of the Middle East Symposium on Simulation and Modelling*, pages 183–187, 2002.

[97] Narayanan C. Krishnan and Diane J. Cook. Activity recognition on streaming sensor data. *Pervasive and Mobile Computing*, 10(Part B): 138–154, February 2014.

[98] John Krumm. Real time destination prediction based on efficient routes. In *Society of Automotive Engineers (SAE) 2006 World Congress*, pages 1–6, 2006.

[99] John Krumm and Eric Horvitz. Predestination: Where do you want to go today? *Computer*, 40(4):105–107, April 2007.

[100] Antti Lajunen, Yinye Yang, and Ali Emadi. Review of cabin thermal management for electrified passenger vehicles. *IEEE Transactions on Vehicular Technology*, 69(6):6025–6040, April 2020.

[101] Óscar D. Lara and Miguel A. Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys & Tutorials*, 15(3):1192–1209, 2013.

[102] Óscar D. Lara, Alfredo J. Pérez, Miguel A. Labrador, and José D. Posada. Centinela: A human activity recognition system based on acceleration and vital sign data. *Pervasive and Mobile Computing*, 8(5):717–729, October 2012.

[103] Jeffrey Larson, Kuo-Yun Liang, and Karl H Johansson. A Distributed Framework for Coordinated Heavy-Duty Vehicle Platooning. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):419, 2015.

[104] Sungyoung Lee, Hung Xuan Le, Hung Quoc Ngo, Hyoung Il Kim, Manhyung Han, Young-Koo Lee, et al. Semi-Markov conditional random

fields for accelerometer-based activity recognition. *Applied Intelligence*, 35(2):226–241, 2011.

[105] Ji Li, Xin Pei, Xuejiao Wang, Danya Yao, Yi Zhang, and Yun Yue. Transportation mode identification with gps trajectory data and gis information. *Tsinghua Science and Technology*, 26(4):403–416, January 2021.

[106] Xingxing Li, Xiaohong Zhang, Xiaodong Ren, Mathias Fritsche, Jens Wickert, and Harald Schuh. Precise positioning with current multi-constellation global navigation satellite systems: Gps, glonass, galileo and beidou. *Scientific reports*, 5(1):1–14, 2015.

[107] Xuefang Li, Qiang Zhang, Zhanglin Peng, Anning Wang, and Wanying Wang. A data-driven two-level clustering model for driving pattern analysis of electric vehicles and a case study. *Journal of Cleaner Production*, 206(1):827–837, January 2019.

[108] Zhenhuan Li, Haiyang Wang, Xiaming Chen, Yongkun Wang, Yaohui Jin, and Wenjun Zhang. Discovering mass activities using anomalies in individual mobility motifs. In *Proc. of the Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress*, pages 321–326, 2016.

[109] Lin Liao, Dieter Fox, and Henry Kautz. Location-based activity recognition. In *Proc. of the Neural Information Processing Systems Conference*, pages 1–8, 2006.

[110] Lin Liao, Donald J. Patterson, Dieter Fox, and Henry Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5): 311–331, April 2007.

[111] Yaqing Liu, Yong Mu, Keyu Chen, Yiming Li, and Jinghuan Guo. Daily activity feature selection in smart homes based on pearson correlation coefficient. *Neural Processing Letters*, 51(2):1771–1787, January 2020.

[112] Loqate. 4 reasons insurers are enhancing telematics with accurate location data, February 2021. URL `https://www.loqate.com/resources/blog/4-reasons-insurers-are-enhancing-telematics-with-accurate-location-data`. [Online; posted February-2021].

[113] Zhenbo Lu, Zhen Long, Jingxin Xia, and Chengchuan An. A random forest model for travel mode identification based on mobile phone signaling data. *Sustainability*, 11(21):1–21, October 2019.

[114] Mingqi Lv, Ling Chen, Zhenxing Xu, Yinglong Li, and Gencai Chen. The discovery of personally semantic places based on trajectory data mining. *Neurocomputing*, 173(Part 3):1142–1153, January 2016.

[115] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, April 2015.

[116] Yongchang Ma, M Chowdhury, A Sadek, and M Jeihani. Real-Time Highway Traffic Condition Assessment Framework Using Vehicle-Infrastructure Integration (VII) With Artificial Intelligence (AI). *IEEE Transactions on Intelligent Transportation Systems*, 10(4):615–627.

[117] Takashi Nicholas Maeda, Narushige Shiode, Chen Zhong, Junichiro Mori, and Tetsuo Sakimoto. Detecting and understanding urban changes through decomposing the numbers of visitors' arrivals using human mobility data. *Journal of Big Data*, 6(1):4:1–4:25, January 2019.

[118] Rainer Makowitz and Christopher Temple. Flexray - a communication network for automotive control systems. In *Proc. of the International Workshop on Factory Communication Systems*, pages 207–212, 2006.

[119] Walterio Mayol and David W. Murray. Wearable hand activity recognition for event summarization. In *Proc. of the International Symposium on Wearable Computers*, pages 122–129, 2005.

[120] James McInerney, Sebastian Stein, Alex Rogers, and Nicholas R. Jennings. Breaking the habit: Measuring and predicting departures from routine in individual human mobility. *Pervasive and Mobile Computing*, 9(6): 808–822, December 2013.

[121] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. Wherenext: A location predictor on trajectory pattern mining. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 637—646, 2009.

[122] Pietro Morasso. Three dimensional arm trajectories. *Biological Cybernetics*, 48(3):187–194, October 1983.

[123] Kyosuke Nishida, Hiroyuki Toda, and Yoshimasa Koike. Extracting Arbitrary-shaped Stay Regions from Geospatial Trajectories with Outliers and Missing Points. In *Proc. of the ACM SIGSPATIAL International Workshop on Computational Transportation Science*, pages 1–6, 2015.

[124] Ordnance Survey. Using the national grid, 2016. URL https://www.ordnancesurvey.co.uk/documents/resources/guide-to-nationalgrid.pdf. Accessed: 2021-08-30.

[125] Guchan Ozbilgin, Ümit Özgüner, Onur Altintas, Haris Kremo, and John Maroli. Evaluating the requirements of communicating vehicles in collaborative automated driving. *Intelligent Vehicles Symposium*, 2016.

[126] Andrey Tietbohl Palma, Vania Bogorny, Bart Kuijpers, and Luis Otavio Alvares. A clustering-based approach for discovering interesting places in trajectories. In *Proc. of the ACM Symposium on Applied Computing*, pages 863–868, 2008.

[127] Donald J. Patterson, Lin Liao, Dieter Fox, and Henry Kautz. Inferring high-level behavior from low-level sensors. In *Proc. of the International Conference on Ubiquitous Computing*, pages 73–89, 2003.

[128] Howard J Payne and Sanuel C Tignor. Freeway incident detection algorithms based on decision trees with states. *Transportation Research Record*, 1978.

[129] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, June 2005.

[130] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. CRAWDAD dataset epfl/mobility (v. 2009-02-24). Downloaded from https://crawdad.org/epfl/mobility/20090224, February 2009.

[131] Sabina Plimmer. Why do i get different google results in different locations? URL https://www.lcn.com/blog/get-different-results-google-vs-location-users/. [Online; accessed 29-March-2022].

[132] Nicholas G. Polson and Vadim O. Sokolov. Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies*, 79:1–17, 2017.

[133] Xiangjun Qian, Arnaud De La Fortelle, and Fabien Moutarde. A hierarchical Model Predictive Control framework for on-road formation control of autonomous vehicles. *Intelligent Vehicles Symposium*, pages 376–381, 2016.

[134] Lawrence Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, January 1986.

[135] Parisa Rashidi, Diane J. Cook, Lawrence B. Holder, and Maureen Schmitter-Edgecombe. Discovering activities to recognize and track in a smart environment. *IEEE Transactions on Knowledge and Data Engineering*, 23(4):527–539, April 2011.

[136] Charles Raux, Tai-Yu Ma, and Eric Cornelis. Variability in daily activity-travel patterns: the case of a one-week travel diary. *European transport research review*, 8:26, 2016.

[137] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. Activity recognition from accelerometer data. In *Proc. of the Conference on Innovative Applications of Artificial Intelligence*, pages 1541–1546, 2005.

[138] *CAN Specification*. Robert Bosch GmbH, 9 1991. Version 2.0.

[139] Amin Sadri, Yongli Ren, and Flora D. Salim. Information gain-based metric for recognizing transitions in human activities. *Pervasive and Mobile Computing*, 38:92–109, July 2017.

[140] Arman Sargolzaei, Carl D. Crane, Alireza Abbaspour, and Shirin Noei. A machine learning approach for fault detection in vehicular cyber-physical systems. In *Proc. of the International Conference on Machine Learning and Applications*, pages 636–640, 2016.

[141] Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *Proc. of the International Conference on Distributed Computing Systems*, pages 166–171, 2011.

[142] Riccardo Scarinci, Benjamin Heydecker, and Andreas Hegyi. Analysis of Traffic Performance of a Merging Assistant Strategy Using Cooperative Vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 16 (4):2094–2103, February 2015.

[143] Robert Schlich and Kay W Axhausen. Habitual travel behaviour: evidence from a six-week travel diary. *Transportation*, 30:13–36, 2003.

[144] Christian M Schneider, Vitaly Belik, Thomas Couronné, Zbigniew Smoreda, and Marta C González. Unravelling daily human mobility motifs. *Journal of The Royal Society Interface*, 10:1–8, 2013.

[145] Christian Schüldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *Proc. of the International Conference on Pattern Recognition*, pages 32–36, 2004.

[146] Matthew L. Schwall and Joseph C. Gerdes. A probabilistic approach to residual processing for vehicle fault detection. In *Proc. of the American Control Conference*, pages 2552–2557, 2002.

[147] Gideon Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, March 1978.

[148] Muhammad Awais Shafique and Eiji Hato. Classification of travel data with multiple sensor information using random forest. *Transportation Research Procedia*, 22(1):144–153, January 2017.

[149] Reid Simmons, Brett Browning, Yilu Zhang, and Varsha Sadekar. Learning to predict driver route and destination intent. In *Proc. of the IEEE Intelligent Transportation Systems Conference*, pages 127–132, 2006.

[150] Chaoming Song, Tal Koren, Pu Wang, and Albert-László Barabási. Modelling the scaling properties of human mobility. *Nature Physics*, 6: 818–823, 2010.

[151] Thorvald 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. *Kongelige Danske Videnskabernes Selskab*, 5:1–34, 1948.

[152] Yi-lin Sun, Ari Tarigan, Owen Waygood, and Dian-hai Wang. Diversity in diversification: an analysis of shopping trips in six-week travel diary data. *Journal of Zhejiang University-SCIENCE A*, 18:234–244, 2017.

[153] Vivienne Sze, Yu-Hsin Chen, Joel Emer, Amr Suleiman, and Zhengdong Zhang. Hardware for machine learning: Challenges and opportunities. In *Proc. of the IEEE Custom Integrated Circuits Conference*, pages 1–8, 2017.

[154] Kohei Tanaka, Yasue Kishino, Tsutomu Terada, and Shojiro Nishio. A destination prediction method using driving contexts and trajectory for car navigation systems. In *Proc. of the ACM Symposium on Applied Computing*, pages 190—195, 2009.

[155] Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive Computing*, pages 158–175, 2004.

[156] Nursitihazlin Ahmad Termida, Yusak O Susilo, Joel P Franklin, and Chengxi Liu. Understanding seasonal variation in individual's activity participation and trip generation by using four consecutive two-week travel diary. *Travel Behaviour and Society*, 12:52–63, 2018.

[157] Alasdair Thomason, Nathan Griffiths, and Victor Sanchez. Parameter optimisation for location extraction and prediction applications. In *Proc. of the IEEE International Conference on Pervasive Intelligence and Computing*, pages 2173–2180, 2015.

[158] Alasdair Thomason, Nathan Griffiths, and Victor Sanchez. Identifying locations from geospatial trajectories. *Journal of Computer and System Sciences*, 82(4):566–581, June 2016.

[159] Roberto Trasarti, Riccardo Guidotti, Anna Monreale, and Fosca Giannotti. Myway: Location prediction via mobility profiling. *Information Systems*, 64:350–367, March 2017.

[160] Ian Tu, Abhir Bhalerao, Nathan Griffiths, Mauricio Delgado, Thomas Popham, and Alex Mouzakitis. Deep passenger state monitoring using viewpoint warping. In *Proc. of the International Conference on Image Analysis and Processing*, pages 137–148, 2017.

[161] Ian Tu, Abhir Bhalerao, Nathan Griffiths, Mauricio Muñoz Delgado, Alasdair Thomason, Thomas Popham, and Alex Mouzakitis. Dual viewpoint passenger state classification using 3d cnns. In *IEEE Intelligent Vehicles Symposium*, pages 2163–2169, 2018.

[162] James Van Hinsbergh, Nathan Griffiths, Phillip Taylor, Alasdair Thomason, Zhou Xu, and Alex Mouzakitis. Vehicle point of interest detection using in-car data. In *Proc. of the ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery*, pages 1–4, 2018.

[163] James Van Hinsbergh, Nathan Griffiths, Phillip Taylor, Zhou Xu, and Alex Mouzakitis. Classifying vehicle activity to improve point of interest extraction. *Mobile Information Systems*, 2021(1), September 2021.

[164] James Van Hinsbergh, Nathan Griffiths, Zhou Xu, and Alex Mouzakitis. Using trajectory sub-clustering to improve destination prediction. *Mobile Information Systems*, 2022(1), October 2022.

[165] Tim van Kasteren, Athanasios Noulas, Gwenn Englebienne, and Ben Kröse. Accurate activity recognition in a home setting. In *Proc. of the International Conference on Ubiquitous Computing*, pages 1—-9, 2008.

[166] András Varga and Rudolf Hornig. An overview of the OMNeT++ simulation environment. In *Proc. of the International Conference on Simulation Tools and Techniques for Communications, Networks and Systems*, pages 1–10, 2008.

[167] Sudip Vhaduri and Christian Poellabauer. Hierarchical Cooperative Discovery of Personal Places from Location Traces. *IEEE Transactions on Mobile Computing*, 17(8):1865–1878, August 2018.

[168] Bao Wang, Linjie Gao, and Zhicai Juan. Travel mode detection using gps data and socioeconomic attributes based on a random forest classifier. *IEEE Transactions on Intelligent Transportation Systems*, 19(5):1547–1558, August 2018.

[169] Liang Wang, Zhiwen Yu, Bin Guo, Tao Ku, and Fei Yi. Moving destination prediction using sparse dataset: A mobility gradient descent approach. *ACM Trans. Knowl. Discov. Data*, 11(3), April 2017.

[170] Shen Wang, Soufiene Djahel, Zonghua Zhang, and Jennifer McManis. Next Road Rerouting: A Multiagent System for Mitigating Unexpected Urban Traffic Congestion. *IEEE Transactions on Intelligent Transportation Systems*, 17(10):2888–2899, March 2016.

[171] Jamie A. Ward, Paul Lukowicz, and Hans W. Gellersen. Performance metrics for activity recognition. *ACM Transactions on Intelligent Systems and Technology*, 2(1):6:1–6:23, January 2011.

[172] Jürgen Wiest, Matthias Höffken, Ulrich Kreßel, and Klaus Dietmayer. Probabilistic trajectory prediction with gaussian mixture models. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 141–146, 2012.

[173] Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2016.

[174] Mengwen Xu, Dong Wang, and Jian Li. Destpre: A data-driven approach to destination prediction for taxi rides. In *Proc. of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 729—-739, 2016.

[175] Andy Y. Xue, Rui Zhang, Yu Zheng, Xing Xie, Jin Huang, and Zhenghua Xu. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *Proc. of the IEEE International Conference on Data Engineering (ICDE)*, pages 254–265, 2013.

[176] Yanxu Zheng, S. Rajasegarar, and C. Leckie. Parking availability prediction for sensor-enabled car parks in smart cities. In *2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6, 2015.

[177] Juan Ye, Graeme Stevenson, and Simon Dobson. Kcar: A knowledge-driven approach for concurrent activity recognition. *Pervasive and Mobile Computing*, 19:47–70, 2015.

[178] Quan Yuan, Zhihan Liu, Jinglin Li, Junming Zhang, and Fangchun Yang. A traffic congestion detection and information dissemination scheme for urban expressways using vehicular networks. *Transportation Research Part C*, 47(Part 2):114–127, October 2014.

[179] Kai Zhao, Denis Khryashchev, Juliana Freire, Cláudio Silva, and Huy Vo. Predicting taxi demand at high spatial resolution: Approaching the limit of predictability. In *Proc. of the 2016 IEEE International Conference on Big Data (Big Data)*, pages 833–842, 2016.

[180] Yu Zheng. Trajectory data mining: An overview. *ACM Transactions on Intelligent Systems and Technology*, 6(3):1–41, May 2015.

[181] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from GPS trajectories. In *Proc. of the International Conference on World Wide Web*, pages 791–800, 2009.

[182] Chen Zhong, Michael Batty, Ed Manley, Jiaqiu Wang, Zijia Wang, Feng Chen, and Gerhard Schmitt. Variability in regularity: Mining temporal mobility patterns in london, singapore and beijing using smart-card data. *PloS one*, 11(2):1–17, February 2016.

[183] Chen Zhong, Patrizia Sulis, and Ed Manley. Analysing interlinked urban functions through mobility motifs. In *Proc. of the Conference on Geographical Information Science Research*, pages 1–5, 2017.

[184] Changqing Zhou, Dan Frankowski, Pamela Ludford, Shashi Shekhar, and Loren Terveen. Discovering Personal Gazetteers: An Interactive Clustering Approach. In *Proc. of the ACM International Workshop on Geographic Information Systems*, pages 266–273, 2004.

[185] Changqing Zhou, Dan Frankowski, Pamela Ludford, Shashi Shekhar, and Loren Terveen. Discovering personally meaningful places: An interactive clustering approach. *ACM Transactions on Information Systems*, 25(3): 12:1–12:31, July 2007.

[186] Chun Zhu and Weihua Sheng. Human daily activity recognition in robot-assisted living using multi-sensor fusion. In *IEEE International Conference on Robotics and Automation, 2009.*, pages 2154–2159. IEEE, 2009.

[187] Brian D. Ziebart, Andrew L. Maas, Anind K. Dey, and J. Andrew Bagnell. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *Proc. of the International Conference on Ubiquitous Computing*, pages 322–331, 2008.