JAMES COOK UNIVERSITY

DOCTORAL THESIS

From Spatio-Temporal Trajectories To Succinct And Semantically Meaningful Patterns

Author: Luke Leslie BERMINGHAM
BIT (Hons)

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy (Information Technology)

in the

College of Business, Law and Governance

April 6, 2018

Declaration of Authorship

I, Luke Leslie BERMINGHAM, declare that this thesis titled, "From Spatio-Temporal Trajectories To Succinct And Semantically Meaningful Patterns" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date: 6th April 2018

Acknowledgements

This thesis represents my journey through the world of research and academia, guided every step of the way by my friendly and excellent supervisor, Professor Ickjai (Jai) Lee. I first met Jai in the second year of my undergraduate degree and he taught me about data-structures and algorithms. That subject, and the way Jai taught it, ignited a strong interest in me for computer science. Jai noticed this and helped me go from strength-to-strength over the course of the following seven years. Jai has always been exceptionally accommodating, responding to hundreds of emails and phone calls and, more generally, keeping me on track with my work. Over the duration of the PhD, especially, Jai has been a constant source of help and mentorship. Jai also challenged me to reach for higher goals and constantly encouraged me to be better than I thought I could ever be. Because of this, I have learnt many new skills and brought old skills to new levels: these I will take with me forever. For all this, I owe Jai a huge debt of gratitude and thank him wholeheartedly.

The other major source of support I need to acknowledge is my family. My parents, Matthew and Diana, have been extremely patient and generous throughout my studies; for this I am eternally grateful to them. I would also like to thank my brother, Blake, for his helpful and patient advice, teaching me some tips and tricks about formal writing. To my sisters, Nikki and Anna, I thank them for their joshing and support.

Also, I would like to thank Adam Rehn and Aidan Possemiers, my fellow PhD students, for swapping PhD stories and allowing me to distract them when I needed a break. Special thanks go to my long-time friend Nicholas Pace who was checking up on me and encouraging me in last six months of intense thesis writing, review, and compilation.

Lastly, I would like to acknowledge the great work of the paid external proofreader I enlisted: Briana Possemiers. After all drafting was completed, Briana proofread the thesis, checking spelling, grammar, punctuation, sentence structure, style, and language usage. I declare that I have read the "Guidelines for the Editing of Research Theses by Professional Editors" and believe Briana's editorial work on this thesis complies.

Statement of the Contribution of Others

Nature of Assistance	Contribution	Names
Supervision support	Primary, secondary, and secondary, supervision, respectively.	Professor Ickjai Lee, Dr Joanne Lee, and Dr Jason Holdsworth.
Research support	Drafting, paper/thesis design, data analysis, statistics support, math support, review, and correctness checks.	Professor Ickjai Lee.
Proofreading support	Grammar, spelling, punctuation, sentence structure, and language usage.	Professor Ickjai Lee and Briana Possemiers (paid external proofreader).
Financial support	Australian Postgraduate Award (APA) Research Scholarship stipend 3.5 years	James Cook University.

Abstract

Luke Leslie BERMINGHAM

From Spatio-Temporal Trajectories To Succinct And Semantically Meaningful Patterns

It is now possible to track moving entities such as humans, animals, or vehicles at relatively high sampling-rates, over long durations of time. This produces large, detailed spatio-temporal trajectories that contain millions of geographic positions and timestamps. These large spatio-temporal trajectories capture the potentially-interesting behaviours of individual entities; they are prime candidates for data mining and knowledge discovery. However, within the trajectory data mining and knowledge discovery process I have identified four challenges that hinder the discovery of succinct and semantically meaningful trajectory patterns: spatial uncertainty, trajectory complexity, pattern complexity, and semantic meaning.

The first challenge, spatial uncertainty, is present in many GPS trajectories. As the global positioning technology used to record trajectories is not entirely accurate, it produces noisy recordings for various reasons: antenna quality, satellite availability, and multi-path errors, inclusively. This extra noisiness in the data makes trajectories more difficult to efficiently mine; it increases the likelihood of discovering false patterns, also, potentially, masking real patterns.

The second challenge, trajectory complexity, refers to the large size and redundancy that is now typical due to the high sampling-rate and long-duration of real-world trajectory data collection. High sampling-rates and long durations are both effective techniques to increase the likelihood of capturing more patterns. I expect that trajectories will be sampled at increasingly higher rates, over longer durations, especially due to the low cost of storage and advances in battery technology. Decreasing the sampling-rate or duration is not an ideal solution because it arbitrarily reduces the amount of information captured. Unsimplified, however, these large trajectories do significantly slow down and complicate the mining process; as we mine increasingly larger trajectory datasets a solution is increasingly important.

Pattern complexity is the third challenge, and occurs because many existing trajectory data mining approaches produce pattern outputs that are not succinct or easily interpretable. In fact, the pattern outputs are sometimes so large they can overload the user and make knowledge discovery overly time consuming. Typically, to make sense of such trajectory patterns a human operator is required to use their expertise to further filter the results or manually select key patterns to represent supposed general trends. Ideally, mined trajectory patterns should be succinct enough that any post-processing by a human operator would needlessly discard information: if additional processing is required to aid pattern interpretation, that processing is far better suited as a step in the mining process.

Semantic meaning refers to the inherent absence of any contextual information present in raw spatio-temporal trajectories; it is the fourth challenge. Typically, raw

spatio-temporal trajectories only record the geographic location with a time-stamp. Mining these raw spatio-temporal trajectories alone, limits the type of discoverable patterns. To uncover knowledge beyond pure movement patterns, extra contextual information that is semantically meaningful, in the application domain, is required. Each type of semantic information that could be inferred or combined with trajectories, presents a unique challenge.

Overall, the aim of this thesis is to investigate solutions to these four challenges to ultimately produce succinct and semantically meaningful trajectory patterns. To do so I divided the thesis into four parts and in each part I addressed some combination of these four challenges at various stages in the trajectory data mining and knowledge discovery process. In the first, I investigated a pre-processing solution that addresses the challenge of trajectory complexity. Specifically, I introduced a framework to create spatio-temporal trajectory simplification approaches. Using this framework I created several spatio-temporal simplification algorithms based on well-known poly-line simplification techniques, evaluating them using multiple real-world trajectory datasets. The results indicated that a number of the simplification algorithms produced were both efficient and effective at reducing the trajectory complexity of the tested real-world trajectories.

Second, I investigated a sequential pattern mining approach that addresses all four challenges in the context of vehicle trajectories. They were chosen for this section because they are simpler to process: they are constrained, in that they only travel underlying road networks. In this approach, I map-matched several real-world vehicle trajectory datasets onto road networks, thus removing their spatial uncertainty, reducing their complexity, and transforming them into a series of semantically meaningful street names. When using traditional sequential pattern mining approaches, mining these large, yet highly redundant sequences produced far too many patterns; meaningful interpretation became impossible. Thus, to overcome the challenge of pattern complexity I mined the sequences using an algorithm I created called DC-SPAN. DC-SPAN mines a highly succinct, but lossy, set of contiguous patterns where the user can control the sub-sequence redundancy of the pattern output. Experiment results on real-world bus trajectories showed that compared to existing contiguous sequential pattern mining approaches DC-SPAN was able to achieve as much a 98% compression in its pattern output while trading off only a 20% increase in lossiness. The results of this section were promising, but ultimately, largely specific to the vehicle trajectory domain.

In the third part I introduced a pre-processing approach called POSMIT. POSMIT annotates each spatio-temporal entry, in a trajectory, with a semantic label indicating whether the entity was stopping or moving within that recording. This semantic stop/move label is a step towards the challenge of enriching raw trajectories with semantic meaning because it can be used to infer further semantic information later in the trajectory data mining process. For example, an extended subsequence of stopping entries occurring inside a restaurant may indicate that the tracked entity was dining. Existing stop/move classification approaches are based on geographic and clustering-based concepts. These approaches definitively label each entry as a stop or a move, meaning that the accuracy of the resulting classification is strongly linked to the user's ability to estimate the required parameters. Conversely, POSMIT computes the probability that a given entry is stopping; then, stopping entries with probabilities below a user-specified threshold are filtered out and become moves. Unlike existing approaches, this feature of POSMIT allows users to tend the result towards having less false-positive stop classifications, which is important in applications like data mining where

false-positives can lead to false patterns. The experiment results on real-world ground-truth stop/move annotated trajectories revealed that, compared to the existing approaches that were tested, POSMIT achieved a higher classification accuracy while also being more robust in parameter selection.

Finally, I used several concepts from the previous sections, both directly and indirectly, and introduced STOSEM: an overall semantic trajectory data mining approach that addresses all four of the identified challenges. STOSEM begins by using POSMIT to enrich each trajectory with stop/move information. Then, STOSEM proceeds to cluster these stop/move annotated trajectories into sequences of extended stop episodes. Clustering the individual recordings into stop episodes greatly reduces the raw trajectory complexity, neatly handling the problem of spatial uncertainty by representing many stopping entries within a single stop radius. After the stop episode formulation STOSEM then incorporates a repository of real-world places. This addresses the challenge of semantic meaning: all nearby real-world places are associated with relevant stop episodes to build a list of candidate places where the stop may have actually occurred. STOSEM, then, proceeds to match each sequence of stop episodes to its likely sequence of visited real-world places, using a probabilistic place-matching algorithm. In general, STOSEM's steps, as described thus far, essentially transform each raw trajectory into a discrete, succinct, and human readable series of place visitations that describe the journey of each tracked individual. This transformation makes data mining simple because this sequence of places is an ideal input for traditional frequent itemset and sequential pattern mining algorithms while remaining succinct enough to avoid the issue of pattern complexity.

To evaluate the efficiency and effectiveness of STOSEM I performed quantitative experiments using real-world and synthetic datasets. Some key findings from the quantitative experiments include: that the stop episode clustering is an effective simplification technique that preserves semantic meaning; that the proposed probabilistic place-matching approach was more accurate than the non-probabilistic approaches I tested. Additionally, to empirically test the applicability of STOSEM I performed a case study using a real-world human trajectory dataset called Geolife. The case study revealed that mining these place visitations, using traditional frequent itemset and sequential pattern mining algorithms, produced succinct and semantic trajectory pattern output. A key finding from the patterns was the existence of certain kinds of participants, such as University attendees and Microsoft employees: both specific participant demographics reported by the original Geolife researchers. Another key finding was the observed behaviours of certain kinds of participants, based on their visitations. For example, some participants seemingly revealed their University attendance timetables through their collected trajectories. These kinds of results, extracted neatly from raw spatio-temporal trajectories are very encouraging and motivate us to extend this approach to future works. This technique could also uncover move episodes between extended stops; they could then be classified into the likely mode of transport being used (i.e. car, bus, walking etc.).

Overall, this study produced several approaches that addressed the challenges of spatial uncertainty, trajectory complexity, pattern complexity, and semantic meaning across multiple real-world trajectory data mining problems. In particular, my cumulative work on STOSEM addresses all challenges, producing succinct and semantically meaningful patterns for real-world human trajectory datasets.

Contents

List of Figures

List of Tables

List of Abbreviations

GPS	**G**lobal **P**ositioning **S**ystem
GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
RoI	**R**egion **o**f **I**nterest
PoI	**P**oint **o**f **I**nterest
SPM	**S**equential **P**attern **M**ining
SED	**S**ynchronised **E**uclidean **D**istance

Chapter 1

Introduction

In this thesis I investigate the extraction of valuable information from GPS trails, generated by moving entities. This is otherwise known as *trajectory data mining and knowledge discovery*. Contextually, a trajectory is a sequence describing the movements of an entity; data mining is the process that extracts patterns from these trajectories, while knowledge discovery occurs when a human operator gains valuable information from patterns uncovered. I illustrate an overview of the trajectory data mining and knowledge discovery process in Figure 1.1.

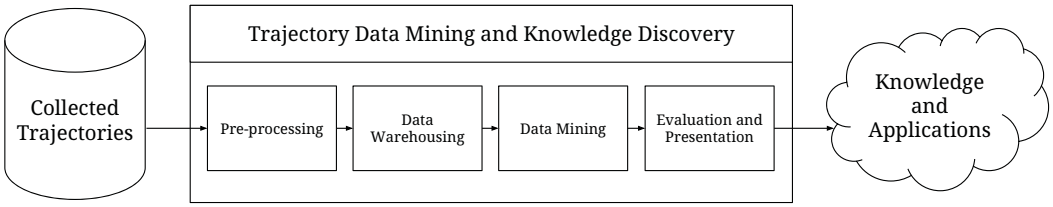


FIGURE 1.1: An overview of the trajectory data mining and knowledge discovery process.

Before proceeding further there are some required concepts in Section 1.1 that the reader should familiarise themselves with as they are referred to throughout this thesis.

1.1 Preliminaries

I refer to the trails of a moving entity as a trajectory. To create a trajectory, the movements of a target entity must be recorded as an ordered sequence. Typically, trajectory data collection methods include: smart phones, GPS trackers, WiFi networks, cellular towers, geo-tagged media, and video surveillance. In this thesis I limit the study of trajectories to those recorded by GPS. The interval between recordings varies between technologies; this is often a factor for consideration with portable devices, as there exists a trade-off between the regularity of the recording interval and battery consumption. I highlight that, due to the lack of continuous real-time sampling a recorded trajectory does not perfectly capture the actual movements of the tracked entity. In fact, a recorded trajectory represents a record of an entity's location history captured at a specific resolution. To illustrate this I present Figure 1.2 which demonstrates the distinction between an entity's true movements and the actual recorded trajectory. Notice two strange artefacts emerge when comparing the true movement to the recorded GPS trajectory. This is because the entity stopped moving at each of these locations but GPS inherently jitters due to inaccuracy (this is discussed more in Section 1.4.1).

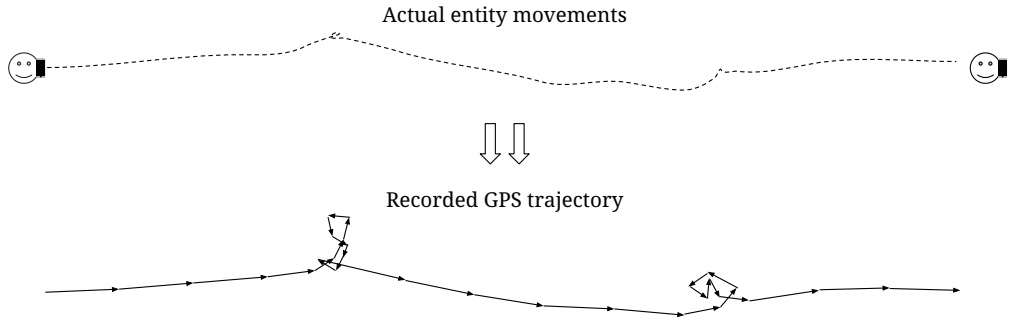


FIGURE 1.2: The distinction between an entity's true movements and its recorded trajectory. GPS inaccuracy causes the two stops to appear noisy.

1.1.1 Spatial Trajectories

All trajectories are sequences —this is an important distinct; not all sequences are trajectories, however. At the absolute minimum a trajectory must always represent the movement of an entity. I call this kind of minimum trajectory a *spatial trajectory*, and provide a formal definition in Definition 1; a visual illustration can be found in Figure 1.3.

Definition 1. Spatial trajectory. A spatial trajectory, T_s, is a list of spatial entries, $(\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle, \ldots, \langle x_n, y_n \rangle)$, where $x_i, y_i \in \boldsymbol{R^2}$ for $i = \{1, 2, \ldots, n\}$.

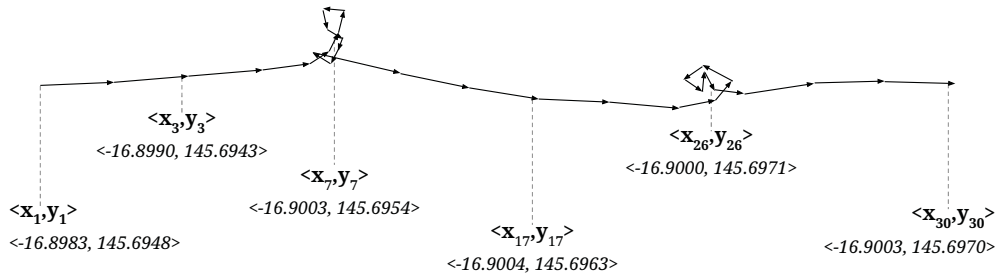


FIGURE 1.3: A spatial trajectory.

Due to this thesis focusing on trajectories collected using GPS, x and y are latitude/longitude pairs unless otherwise stated.

1.1.2 Spatio-temporal Trajectories

Spatial trajectories represent the minimal amount of information possible for a sequence to be considered a trajectory; in practice, however, many other attributes are typically recorded alongside this spatial information. A common and useful attribute found in many recordings is a time-stamp. I call trajectories that incorporate spatial and temporal information *spatio-temporal trajectories*, and provide a formal definition in Definition 2; a visual illustration can be found in Figure 1.4.

Definition 2. Spatio-temporal trajectory. A spatio-temporal trajectory, T_{st}, is a list of spatio-temporal entries, $(\langle x_1, y_1, t_1\rangle, \langle x_2, y_2, t_2\rangle, \ldots, \langle x_n, y_n, t_n\rangle)$, where $x_i, y_i \in \boldsymbol{R^2}$ and $t_i \in \boldsymbol{R}^+$ for $i = \{1, 2, \ldots, n\}$ and $t_1 < t_2 < \ldots < t_n$.

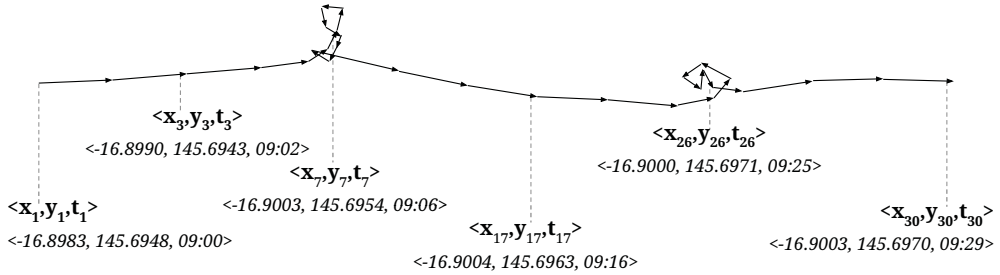


FIGURE 1.4: A spatio-temporal trajectory.

1.1.3 Semantic Trajectories

Broadly, semantic trajectories are trajectories that are enriched with an extra dimension (or set of dimensions) that provides some sort of human interpretable context. This extra contextual information is not strictly defined; common examples include: whether the entity is stopping or moving; the place an entity is visiting; the mode of transport an entity is using; or, the activity an entity is doing. I provide a formal definition for a semantic trajectory in Definition 3; a visual illustration can be found in Figure 1.5.

Definition 3. Semantic trajectory. A semantic trajectory, T_{sem}, is a list of spatio-temporal, semantically annotated entries, $(\langle x_1, y_1, t_1, a_1\rangle, \langle x_2, y_2, t_2, a_2\rangle, \ldots, \langle x_n, y_n, t_n, a_n\rangle)$, where $x_i, y_i \in \boldsymbol{R^2}$, $t_i \in \boldsymbol{R}^+$, and $a_i \in \mathcal{A}$, for $i = \{1, 2, \ldots, n\}$ and $t_1 < t_2 < \ldots < t_n$.

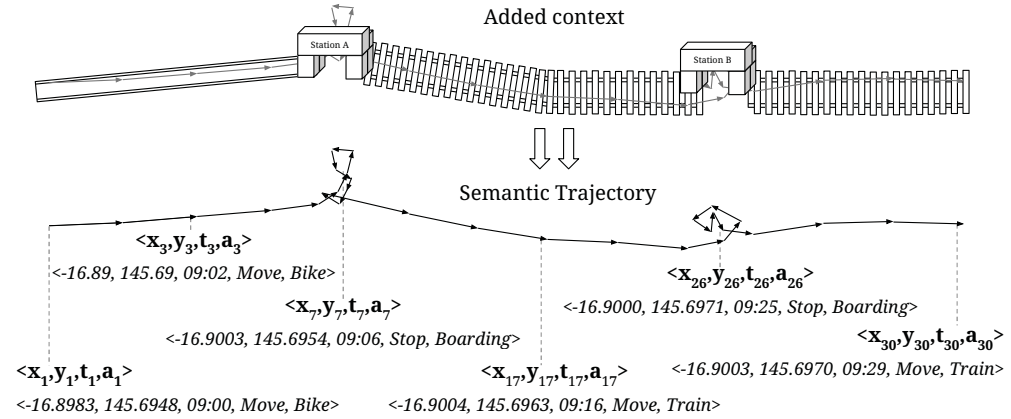


FIGURE 1.5: A semantic trajectory.

In Definition 3, $\mathcal{A}$ is a finite set of semantic labels, any of which could be used to annotate a trajectory entry. The set of labels is context-specific based on the user's data mining and knowledge discovery goals. For example, to indicate whether the entity was stopping or moving one could use a set of labels such as $\{Stop, Move\}$.

Other examples include using: $\{Car, Bus, Bike, Walking\}$ to indicate the transport mode, or $\{Commuting, Working, Exercising, Resting\}$ to indicate the activity likely being performed. I highlight that trajectory entries can also be annotated with semantic labels from multiple sets to further enrich their meaning, such as: $\{16.9186, 145.778, 09:15, 14/08/17, Move, Bus, Commuting\}$.

1.1.4 Semantic Episodes

Often the same semantic label is applied to a number of contiguous entries, as the semantic event being annotated occurs over an extended duration. For example, a stopping entity may stop for multiple minutes or hours, which would mean many contiguous entries are labelled with $\{Stop\}$. Thus, a common processing step, after semantic enrichment is episode formulation: contiguous entries with the same labels are combined together into a single entry that I call a *semantic episode*. I provide a formal definition of a semantic episode in Definition 4; a visual illustration can be found Figure 1.6. Note, how the semantic trajectory shown in Figure 1.5 is greatly simplified when it is transformed into a sequence of semantic episodes: shown in Figure 1.6. This is useful because compressed, but still semantically meaningful, sequences are far simpler and more efficient to mine than the large trajectories that they are derived from.

Definition 4. Semantic episode. A semantic episode, $E_{j,k}$, is a single vector that represents a portion of movement from a semantic trajectory, T_{sem}. The portion of movement starts from the trajectory's j-th index and ends at its k-th index (inclusive), where $j \leq k$. Note that a semantic episode always maximises the number of contiguous entries in T_{sem} with the same semantic label, which means that $T_{sem}.a_j = T_{sem}.a_{j+1} = \ldots = T_{sem}.a_k \wedge T_{sem}.a_{j-1} \neq T_{sem}.a_j \wedge T_{sem}.a_{k+1} \neq T_{sem}.a_k$. The actual vector of the semantic episode $E_{j,k}$ contains a geometry, g, which represents a portion of an entity's movement from indices j to k; a start and end time; ts and te respectively; and a semantic label, a. Specifically, $E_{j,k} = \langle g, ts, te, a \rangle$, where g is constructed using $\{T_{sem}.xy_j, T_{sem}.xy_{j+1}, \ldots, T_{sem}.xy_k\}$, $ts = T_{sem}.t_j$, $te = T_{sem}.t_k$, and $a = T_{sem}.a_j = T_{sem}.a_{j+1} = \ldots = T_{sem}.a_k$.

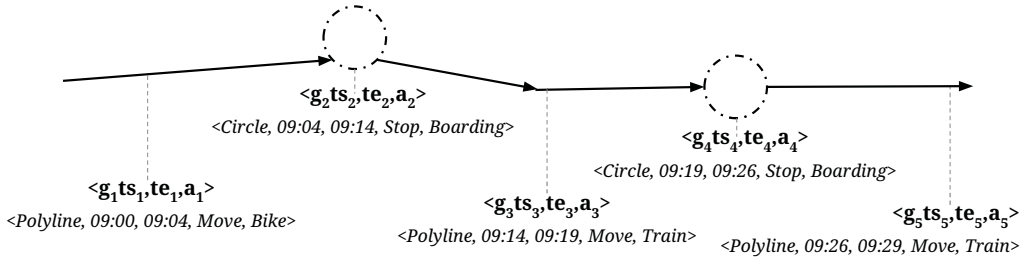


FIGURE 1.6: A sequence of semantic episodes formulated from the semantic trajectory shown in Figure 1.5.

The geometry, g, of a semantic episode is often used to infer additional semantic information. For example, the geometry of a semantic episode with the semantic label $\{Stop\}$ (a stop episode), is useful to discover the real-world place where the stop occurred. There are many types of semantic episodes though: how g is, therefore, constructed, varies. In Figure 1.6 stop episode geometries are constructed

as circles (to account for GPS noise, see: Section 1.4.1) and the move episode geometries are constructed as poly-lines that capture the underlying footpaths and railways.

1.1.5 Choice of Wording

I highlight that definitions, 1, 2, 3, and 4, are used and built upon throughout this thesis: they will, however, not repeat again in any other section or chapter. I encourage readers to refer back to this section as frequently as needed. Additionally, it should be noted that, from this point forward, the word “trajectory" will be used to refer to the general sequence of tracked entity movements; where more specificity is required, the terms “spatial trajectory", “spatio-temporal trajectory", or “semantic trajectory" will be used. Additionally, it is important to recognise such distinctions: a sequence of semantic episodes no longer conforms to the minimum trajectory outlined in Definition 1. Thus, sequences of semantic episodes are not called trajectories in this thesis; when I refer to them, I qualify them with specific names such as “stop/move episodes".

With the preliminaries defined, I present the background of the field in Section 1.2.

1.2 Background

On the morning of May 1st, 2000, the White House issued a statement from U.S. president Bill Clinton:

> “Today, I am pleased to announce that the United States will stop the intentional degradation of the Global Positioning System (GPS) signals available to the public beginning at midnight tonight. We call this degradation feature Selective Availability (SA). This will mean that civilian users of GPS will be able to pinpoint locations up to ten times more accurately than they do now. [...] This increase in accuracy will allow new GPS applications to emerge and continue to enhance the lives of people around the world." (The White House, 2000)

Once SA was deactivated civilian GPS accuracy did vastly improve: measurements taken by the U.S. Air Force Space Command, using a dual frequency receiver, a few hours after SA was disabled, showed civilian GPS error decreasing from approximately 100 metres to 4.6 metres (GPS.gov, 2000). This improved accuracy, paired with increased affordability, encouraged the widespread adoption of GPS technology into the civilian, industrial, and scientific sectors as we recognize it today. Specifically, the removal of SA, coupled with the general improvement of GPS receivers and satellites made a wide variety of tasks much more viable, including: navigation, geo-locating, recommendation, and tracking.

The *literal* overnight emergence of reasonably accurate, large-scale, positional tracking for public use, meant that, quite suddenly, all kinds of moving entity data was being collected. Thus, it is no coincidence that research interest and techniques for pre-processing, storing, querying, and mining trajectory data surged in the 2000s.

1.2.1 The First Generation

Various researchers (Laube, 2015; Amato et al., 2018) chronicle the early 2000s as a time where there was an abundance of movement data and many unexplored

challenges towards processing it. Laube (2015) finds these conditions as responsible for a “gold-rush" amongst researchers. Laube (2015) suggests that “in the beginning, the field of movement analysis was so wide open, there were so many open and interesting problems". I refer to that decade, 2000 to 2010, as the *first generation* of trajectory data mining and knowledge discovery. Briefly, I characterise the first generation as a time when many core concepts and techniques were defined, allowing users to extract knowledge from raw trajectories for the first time. Analysis of relevant surveys (Zheng and Zhou, 2011; Andrienko and Andrienko, 2013; Amato et al., 2018) combined with my own knowledge, leads me to this conclusion: during the first generation, a great deal of research was done translating and extending relevant concepts and techniques from the fields of data mining and database engineering. Pre-processing, indexing, querying, clustering, classification, outlier detection, pattern mining, and visualisation, are all techniques that experienced huge advances in this time. In general, I classify the first generation as being largely focused on discovering knowledge *directly* from the raw trajectories: that is, discovering patterns related to the pure spatial or spatio-temporal movements of the entities.

During this first generation, however, various researchers (Alvares et al., 2007; Spaccapietra et al., 2008; Palma et al., 2008) presented concepts and techniques to transform trajectories from pure movement sequences into contextually meaningful sequences. This formed the basis for what is now referred to as a *semantic trajectory* (Parent et al., 2013). A semantic trajectory is produced by enriching a raw trajectory with extra data that describes the movements of the entity in a way that is more contextually meaningful in some application areas (a more formal definition is given in Section 1.1.3). For example, for urban planning purposes, a raw trajectory of person moving around a city may be transformed into a sequence of visited-places using a database of known-places. Fortunately, late in the first generation (some sources record activities around 2007 (Tauberer, 2014)) there was a general shift in industry and government towards open-data. Some of this open-data, such as land usage regions and property boundaries, has been directly relevant for formulating semantic trajectories. This boost in relevant data sources, alongside large community volunteered repositories of geographic data such as OpenStreetMap [1], further motivated research interest in semantic trajectories. Ultimately, this had led to the *second generation* of trajectory data mining and knowledge discovery, from 2010 to current times (2017).

1.2.2 The Second Generation

Survey analysis (Parent et al., 2013; Bao et al., 2015; Zheng, 2015; Feng and Zhu, 2016), and my own knowledge of the field, leads me to classify the second generation as moving away from processing raw trajectories and shifting, generally, towards uncovering contextual patterns and building real-world applications from semantic trajectories. For example, in the second generation we have seen approaches that: construct location-based social networks to recommend user-specific locations, travel sequences, and friends (Zheng et al., 2010); find the different functions of various geographic regions in a city based on human movements (Yuan et al., 2012); and give personalised driving directions based on weather, traffic conditions, and personal preference (Yuan et al., 2013). These kind of works are a clear step beyond purely movement-based patterns characterised by

[1] `https://www.openstreetmap.org`

the first generation. Though there are many proposed techniques for semantic trajectory applications, I argue that the semantic gap between raw trajectories and human interpretation is still flourishing, with much-room left for research contribution. I argue that each type of contextual information is a unique challenge to combine with trajectories during semantic enrichment; thus, the field as a whole will close the semantic gap incrementally, solving one contextual data source at a time.

Additionally, I highlight that while the notion of semantic trajectories have certainly expanded the field, the second generation is by no means wholly-characterised by them. Effectively formulating semantic trajectories and real-world applications from raw trajectories requires one to lean heavily on many of the foundations established in the first generation. Many first-generation-techniques, fundamental as they are, do not stand as absolute solutions to the varied and complex scenarios encountered when processing real-world trajectories. Thus, much work has also been conducted in the second-generation to refine and specialise many of the approaches from the first generation. For example, if we consider trajectory pre-processing approaches: the first generation introduced approaches to remove input complexity and mitigate GPS inaccuracy, such as (Meratnia and By, 2004; Hightower and Borriello, 2004) (respectively); the second generation introduced newer approaches such as (Richter et al., 2012), which handle complexity and inaccuracy while also retaining trajectory recordings that are relevant during semantic enrichment.

1.3 Future Influences

It should be clear, following Section 1.2, that the field of trajectory data mining and knowledge discovery is well established. Within the field this thesis is situated toward the latter-end of a decade of work in the so-called 'second generation'. The aim of this thesis is to build upon work done before; however, to specifically determine the direction of this thesis we, as a researchers aware of the current state-of-the-art, must consider where we think the field is heading. Reflecting on the first and second generations, I argue that the major influences were: the new-found availability and affordability of technology (i.e. GPS receivers and computers); the subsequent abundance of movement data; and, the possibility to combine contextual data sources with trajectories, to discover patterns that are more meaningful in their domain. In the subsequent sections I consider various technological forecasts with regards to the current trends in the field and present my own speculations on the future influences of the field.

1.3.1 Outdoor Positional Accuracy

Global navigation satellite system (GNSS) availability is increasingly becoming a non-issue due to number of satellites being launched by different global powers (i.e USA's GPS, Russia's GLONASS, Europe's Galileo, Japan's ZQSS, China's BeiDou-2) (Alexander, 2014). Studies have shown that using these multiple GNSSs provides continuous positioning up to 99.5% of the time, even in challenging environments like city centres (Li et al., 2015). Thus, the major remaining problem hindering outdoor GNSS accuracy is multipath. Multipath occurs when the path to the receiver is blocked and reflected by some object, causing the satellite signal to take an unnecessarily long path that introduces positional error (Misra and Enge, 2011,

pp.175). In order to mitigate the impact of multipath, some works have shown the viability of a real-time kinematic precise point positioning (PPP-RTK) system that would use a network of terrestrial reference stations to provide real-time corrections to nearby consumer devices. Ultimately, they would produce centimetre-accurate outdoor positioning (Teunissen and Khodabandeh, 2015). Centimetre-accurate trajectories would greatly increase the effectiveness of existing data mining approaches across the board and would likely motivate more research towards complex inference tasks, like uncovering accurate human behaviour profiles.

1.3.2 Device Sensors

Smart phones have become pervasive; as a result, they have been a popular tool for collecting trajectory data. Many trajectories generated through smart phones are collected in much the same way as trajectories from pure GPS receivers: only collecting the geographic position and the time-stamp. In the future of the field, as we tend more towards semantic trajectories, smart phone sensor data could play an increasingly important role in providing additional context inferring semantic information. In an early study it was shown that by collecting raw data from over twenty different sensors attached to a person various daily activities such as walking, running, or sitting could be inferred automatically (Parkka et al., 2006). It is by no means a leap to say that such inferences could be made by combining raw trajectories and future smart phone sensor data, too. For example, we have already seen the inference of transport mode from spatio-temporal trajectories recorded with accelerometer data (Xia et al., 2014). In addition to the sensors we have today, future smart phones may have sensors that measure other dimensions such as air quality or body temperature. Furthermore, considering the increasing adoption of wearables, paired with smart phones, extra dimensions such as heart rate and blood pressure may potentially be recorded at scale as part of the trajectory data collection process. Extra dimensions such as heart rate, blood pressure, body temperature, and air quality will most certainly further increase the accuracy and robustness of semantic trajectory inference tasks, like activity detection (i.e. identifying that the user was working out at a gym, as opposed to sleeping at home, or walking through the inner city).

1.3.3 Active Trajectory Recording

Active trajectory recording is when the entity's movements are directly tracked (Zheng, 2015, pp.4), typically using a device like a smart phone or GPS receiver. However, active-recording of trajectories using such devices has always been constrained by device limitations. For example, smart phones and GPS receivers have limited battery life, which is drained more quickly if the position is recorded more-frequently. Due to battery limitations, devices recording trajectories are either set to record locations infrequently or the total collection duration is made relatively small. Both compromises limit the potential knowledge discoverable if the entities were recorded frequently over large durations. However, large-scale, high-frequency recording may become a possibility in the future, particularly with the promise of technologies like ultrasonic charging (Ashdown et al., 2013; Radziemski and Makin, 2016) and rapidly charging batteries (Lin et al., 2015; Tang et al., 2017). We have already seen some research towards mining knowledge from streaming (transmitted) trajectory data sources (Yang et al., 2013; Zheng et al.,

2014a; Silva et al., 2016). However, if device batteries were continuously being charged, or being charged rapidly, then near real-time continuous trajectory data streaming would become far more viable: this would undoubtedly motivate even more research towards mining streaming-trajectory data.

1.3.4 Passive Trajectory Recording

Passive trajectory recording occurs when an entity's movements are not directly tracked (Zheng, 2015, pp.4), yet the entity's movements are still able to be approximately recovered. Passive recording techniques are typically made possible because the tracked entity interacts with a series of objects that have known locations. Passive recording, through cellular towers, has been of particular-interest within the second generation, most likely because of the huge number of recoverable human movements. This style of passive tracking is possible because telecommunication companies store call detail records (CDRs) for billing purposes: these contain a record of each cellular tower the customer connected-to and used. Then, using a reference of known cellular tower locations, it is possible to recover the movements of customers. Already, we have seen researchers use passively recorded trajectories from CDRs to mine patterns of popular trips throughout city regions (Jiang et al., 2017), construct origin-destination trip timetables (Alexander et al., 2015), and even estimate road network usage (Toole et al., 2015). However, existing passive recording techniques, CDRs especially, generate trajectories that are sparse in both space and time. This means, these passively generated trajectories are based on extremely simplified versions of the tracked entity's true movements; ultimately, the discoverable patterns are limited. However, in the future, if the unified urban ICT infrastructure promised by smart-cities is realised (Hernández-Muñoz et al., 2011) there will likely be a network of devices installed throughout the city that could double as passive trackers. For example, consider unifying tolls, public transport turn-styles, parking meters, ticketing machines, and public Wi-Fi access points, into one smart-city passive-tracking system: that system alone could generate huge and accurate trajectory datasets. Such datasets would be particularly useful for urban planning and real-time congestion forecasting.

1.4 Motivating Challenges

It is difficult to directly act on many of the potential influences presented in Section 1.3 because they rely on technology that is not yet widely available. However, if we imagine a future where some, or all, of these influencing factors come to fruition, there are some persistent challenges, existing now, that will be further-exacerbated in the future. Specifically, I have identified four challenges that the field faces both now, and moving forward. They are: spatial uncertainty, input complexity, pattern complexity, and semantic meaning.

1.4.1 Challenge One: Spatial Uncertainty

With trajectory data, spatial uncertainty typically refers to two separate issues: sampling rate and measurement inaccuracy (Pfoser and Jensen, 1999). The sampling rate is the rate at which recordings are added to a trajectory; because the sampling rate is not continuous, a real-world trajectory only captures a certain resolution snapshot of an entity's true movements (recall Figure 1.2). For example, consider two consecutive spatio-temporal trajectory recordings: it is a commonly

held assumption that any travel between those two points in space and time can be treated as a linear interpolation from one to the other (Parent et al., 2013, pp.4); that is, it is assumed that the entity travelled in a straight line at a constant speed between those points. However, that assumption may not hold if the duration between recordings or the speed of the entity is sufficiently large. In such cases, we cannot know the true movements of the tracked entity: we are left referring to its trajectory as spatially uncertain. Spatial uncertainty caused by low sampling rates has been an issue throughout much of the first generation, and we have seen many various solutions proposed to handle sparse trajectories (Pelekis et al., 2009; Trajcevski et al., 2004; Lou et al., 2009). However, in the second generation, storage and transmission costs have decreased, which has ultimately afforded users the ability to record using increased trajectory sampling rates. Increasing the trajectory sampling rate solves the issue of spatial uncertainty due to sparseness. Thus, as Long et al. (2013) says, we have seen a general trend "to oversample initially, and then to simplify". However, collecting large and complex trajectories does introduce other issues, which I will elaborate on in Section 1.4.2.

The second spatial uncertainty issue, measurement inaccuracy, is the actual challenge I am referring to when I use the term spatial uncertainty throughout the rest of this thesis. In contrast to sampling rate, spatial uncertainty caused by measurement inaccuracy has been a persistent challenge throughout the first (Diggelen, 2007; Defence, 2008) and second generations (Diggelen and Enge, 2015; Lee et al., 2016); it appears it will remain so, for some time yet. One reason measurement error is so pervasive in trajectory data, is because mass market GPS receivers, like the single frequency receivers commonly found in smart phones, are highly subject to varying accuracy levels (Pesyna et al., 2014). This measurement inaccuracy directly introduces error into the recorded trajectory, thus, perturbing and masking the tracked entity's true movements. Even with SA disabled (see Section 1.2) and current clock, orbit, and atmospheric models producing signal-in-space (SIS) transmission accuracies of approximately one meter (Alexander, 2014), GPS receivers are still subject to large accuracy fluctuations under certain common conditions like multipath (as I mentioned in Section 1.3.1). Worse still, single frequency GPS receivers are likely to remain the most common solution used by the mass market due to their low cost; however, it is these devices that are particularly impacted by multipath due to their low quality antennas (Pesyna et al., 2014). As I mentioned in Section 1.3.1, this inaccuracy may be mitigated by PPP-RTK systems; however, unlike the existing GNSSs used today, a PPP-RTK approach is not a global system and relies on various geographic stakeholders for roll-out. Thus, if PPP-RTK is introduced in the future, it may have a similar roll-out to the internet: suffering coverage and various other issues in some locales, at first. Regardless of PPP-RTK's roll-out, for now, GPS trajectories are somewhat spatially uncertain, which masks both the true patterns in the data, and potentially, introduces errors that can lead to discovering false trajectory patterns. Thus, spatial uncertainty is a real challenge in accurately discovering knowledge from GPS trajectories now and into the future.

One technique to address the issue of spatial uncertainty in trajectories is to transform them from raw spatio-temporal coordinates into discrete entries. For example, transforming raw trajectories into a series of roads travelled, stops made, or places visited. How this is done and accuracy of the result are the two main topics explored in the current state-of-the-art. In this thesis I also explore removing spatial uncertainty by discretising trajectories. Specifically, in Chapter 5 I build upon existing stop and move detection algorithms (Alvares et al., 2007; Palma et al.,

2008) to introduce a new, more parameter-robust, probabilistic algorithm to transform raw trajectories into a series of stops and moves. Additionally, in Chapter 6 I extend my work from Chapter 5, building upon existing approaches (Yan et al., 2013; Lv et al., 2016) for matching stops with places, I introduce a new algorithm to convert a raw spatio-temporal trajectory into a series of discrete place visitations.

1.4.2 Challenge Two: Trajectory Complexity

Trajectory complexity is the term I use to describe the increasing redundancy and size of trajectories that result from long data collection periods, increased recording features, or high sampling rates. Sufficiently large and complex trajectories can cause a number of issues: storage limitations, transmission throttling, processing inefficiency, and visualisation overload, inclusively (Parent et al., 2013, pp.11). Much like spatial uncertainty, trajectory complexity has been a persistent issue throughout the first and second generations, resulting in a number of mitigating approaches (Meratnia and By, 2004; Richter et al., 2012; Vrotsou et al., 2015). However, if any of my forecasts from Section 1.3 are realised —especially continuous trajectory streaming (see Section 1.3.3) or large-scale passive tracking (see Section 1.3.4) —the complexity of trajectories will likely increase an order of magnitude. To illustrate the scale of the potential data, consider, for example, that recording a single entity every second for a month will produce approximately 2.6 million recordings. If we assume each spatio-temporal recording contains approximately 40 bytes of raw data this results in 104 megabytes of raw data produced for a single entity. If city-scale tracking scenarios like those I discussed in Section 1.3.4 are realised trajectory data will likely be generated from thousands, if not millions, of entities. This could equate to terabytes of raw data generated for a single month of tracking. Storage for this amount of data is trivial; however, the time taken to extract patterns from it is, arguably, less so. Using a less frequent sampling rate is one solution to reduce the data size; in many contexts it is not ideal because it arbitrarily lowers the resolution of the recorded movements, and potentially introduces spatial uncertainty through sparseness (recall Section 1.4.1). Alternate solutions involve detecting insignificant or redundant recordings and removing those; however, the significance of individual recordings vary depending on the type of knowledge discovery being conducted and, as we move towards more semantic trajectory data mining, simplification approaches will have to become context considerate, too. Thus, removing complexity from large trajectory datasets is another challenge going forward.

In this thesis I specifically investigate addressing the issue of trajectory complexity by using the aforementioned approach of detecting insignificant entries and removing them from the raw trajectory. In Chapter 3 I generalise the problem of spatio-temporal trajectory simplification and extend several existing poly-line and spatial-only trajectory simplification algorithms (Douglas and Peucker, 1973; Visvalingam and Whyatt, 1993; Lee et al., 2007) to the problem. The result is a general trajectory simplification framework, which I used to create eight, easily tunable, new spatio-temporal trajectory simplification algorithms.

1.4.3 Challenge Three: Pattern Complexity

Pattern complexity is the term I use to refer to the over-abundance of output produced by the trajectory data mining process. In an ideal scenario the data

mining approach would produce a succinct and useful set of patterns for a human operator to interpret. However, if the input trajectories are sufficiently large and homogeneous some trajectory data mining approaches will produce too many patterns to meaningfully interpret. As Yin et al. (2011) say, when there are too many trajectory patterns "it is difficult for users to go through all the patterns in the list to discover the interesting ones. As a result, the interesting trajectory patterns are buried in the massive result set". I argue that pattern complexity is generally tied to trajectory complexity: as trajectory dataset sizes increase generally so will the number of patterns produced. Thus, much the same as Section 1.4.2, if any of my trajectory complexity forecasts, like continuous trajectory streaming (see Section 1.3.3) or large-scale passive tracking (see Section 1.3.4), are realised, then overly large pattern outputs may become a problem. A particularly challenging trait of pattern complexity is that it is implicitly driven by underlying trends in the raw data. Thus, even though simplification may reduce the input complexity of a trajectory dataset it will not (or at least should not) modify or remove the underlying trends in the data; pattern complexity is still potentially present even after pre-processing. Therefore, I argue that the challenge of pattern complexity requires a unique solution, different than that of trajectory complexity.

One solution is to mine trajectory patterns in such a way that the output does not surpass a certain size or redundancy constraint. Doing so allows the pattern complexity to remain bounded during the mining step, which is where the problem of pattern complexity usually appears. This is the approach I take in Chapter 4 where I transform trajectories into a series of visited road segments (each with a unique id) and then introduce new redundancy controlled sequential pattern mining approach. Borrowing concepts from both the fields of map-matching and traditional sequential pattern mining, my approach directly extends a number of existing works from those fields (Newson and Krumm, 2009; Song et al., 2014; Wang et al., 2014; Zhang et al., 2015).

1.4.4 Challenge Four: Semantic Meaning

Collections of raw trajectories track the location histories of many moving entities. In turn, this can reveal knowledge such as similarly moving entities (Buchin et al., 2011), frequently visited regions (Giannotti et al., 2007), or clusters of co-located sub-trajectories (Lee et al., 2007). Such findings are interesting, but drawing and discovering contextually meaningful knowledge typically requires combination with additional semantic data sources relevant to the domain (Parent et al., 2013, pp.5). For example, consider analysing raw movement patterns and trying to discover: people visiting sequences of places; people performing sequences of activities; or, vehicles travelling along specific roads. Raw trajectories do not contain any information required to make these associations, and manually inferring it is time consuming and tedious for a human operator. Thus, this task is typically deferred to the semantic enrichment process —where trajectories are associated with all kinds of data sources, such as place repositories, road networks, or land usage maps. This semantic enrichment of raw trajectories is hugely important in the extraction of human interpretable knowledge because, "the integration of trajectory data with semantic geographical information is the main step for trajectory data analysis in real applications" (Alvares et al., 2007). Of course, much work has already been done in the first (Alvares et al., 2007; Spaccapietra et al., 2008; Monreale et al., 2009) and second generations (Bogorny et al., 2014; Furletti et al., 2013; Renso et al., 2013) towards bridging the semantic

gap between raw trajectories and real-world applications. However, we must keep in mind that there is no single, "one size fits all", semantic trajectory enrichment process for all types of contextual data. In other words, even though each type of semantic information we wish to enrich trajectories with adds potential value to the knowledge discovery step, incorporating each type of semantic information is also a unique challenge.

In this thesis the issue of adding semantic meaning to raw trajectories is investigated in Chapter 6 where I first transform trajectories into a series of stops and then introduce a new probabilistic matching scheme to associate each stop with a likely place from OpenStreetMap. As I mentioned in Section 1.4.1, the place-matching algorithm I introduce in Chapter 6 directly extends a number of existing semantic trajectory data mining works (Yan et al., 2013; Lv et al., 2016).

1.4.5 Summary of Trajectory Data Mining and Knowledge Discovery Challenges

To summarise, the challenges faced when mining and discovering knowledge from GPS trajectories are:

1. **Spatial uncertainty -** GPS trajectories are commonly inaccurate due to positional error introduced from the recording devices. This positional error masks the true movements of the tracked entity and introduces error during the mining step.

2. **Trajectory complexity -** Trajectory datasets are often large and redundant, which makes storing, processing, and visualisation in their raw format inefficient.

3. **Pattern complexity -** Mined trajectory patterns are sometimes too numerous and redundant, which makes interpretation a difficult and time consuming task for a human operator.

4. **Semantic meaning -** Raw trajectories do not inherently contain any semantically meaningful context. Without context much of the meaning behind an entity's journey is hidden, and, by extension the variety of potential knowledge discovered is limited.

1.5 Aims

Given the four challenges introduced in Section 1.4, I have formulated several research aims that are tied to specific topics in the trajectory data mining and knowledge discovery process. Each of the aims focuses on addressing at least one of the four challenges. The aims are as follows:

1. Investigate trajectory simplification to reduce input trajectory complexity for general trajectory data mining.

2. Investigate mining succinct sets of sequential patterns from trajectories to reduce pattern complexity.

3. Investigate classifying raw trajectories as stops and moves to semantically enrich them while mitigating spatial uncertainty.

4. Investigate transforming stopping and moving trajectories into episodes and matching those stop episodes to real-world places to increase semantic meaning, reduce trajectory complexity, and ultimately facilitate a semantic trajectory data mining case-study.

Readers: please note that these aims are further refined into testable research hypotheses in Section 2.6, after the relevant literature in each of the topics has been reviewed and the research gaps identified.

1.6 Thesis Outline

Following this introductory chapter is the literature review in Chapter 2. In Chapter 2 I review a collection of relevant existing approaches and their relation to the four identified challenges of trajectory data mining and knowledge discovery. From this review, I identify several research gaps that form the basis of my choice of topics in the following data chapters: simplification (Chapter 3), distinct pattern mining (Chapter 4), stop/move detection (Chapter 5), and place-matching (Chapter 6).

In Chapter 3, I present a framework for constructing spatio-temporal trajectory simplification approaches; I create several of these approaches using this framework, then evaluate their efficiency and effectiveness against existing spatial-only simplification approaches. The results indicate that the approaches constructed, using this framework were more effective at reducing the complexity of spatio-temporal trajectories than the existing spatial-only approaches that were compared against.

In Chapter 4, I transform a set of vehicle trajectories into a sequence of road network visitations. Additionally, I introduce a new concept called distinct pattern mining that mines lossy, but redundancy controllable, sequential patterns. I then mine the set of distinct patterns from these vehicle trajectories, which succinctly reveals the most frequently travelled roads.

In Chapter 5, I present a probabilistic approach to detect stops and moves in noisy GPS trajectories. I evaluate the accuracy and efficiency of the approach and several existing approaches using various ground-truth, real-world GPS trajectories. The results indicate that the approach achieves higher accuracies and is more robust to parameter selection than the approaches that were compared against.

In the final data chapter, Chapter 6, I introduce the problem of place ambiguity when matching trajectory stops to real-world places; I then present a probabilistic place-matching algorithm to overcome this problem. I quantitatively evaluate the efficiency and effectiveness of the place-matching algorithm by comparing it to similar approaches that do not consider the place ambiguity problem. The results indicate my place-matching algorithm is more accurate than a benchmark approach, that did not consider place ambiguity. Additionally, to evaluate the real-world applicability of my place-matching algorithm, I conduct a case-study where I use my place-matching algorithm to transform a large dataset of real-world human trajectories into sequences of place visitations. I use existing itemset and sequential mining algorithms to mine these place visitations to reveal popular places and popular sequences of places travelled in the dataset. Analysis of these patterns reveals several seemingly real patterns, which hints at the validity of the overall process.

Lastly, in Chapter 7, I present a summary of each chapter, highlighting the key findings and contributions. I also conclude by addressing some limitations, recommendations, and future research directions based on this thesis.

The following is a breakdown of intellectual contributions by relevant authors for each chapter in this thesis:

Ch.	Publication Arising	Intellectual contributions
1	None	Entirely the work of Bermingham.
2	None	Entirely the work of Bermingham.
3	**[Published]** Bermingham, L. and Lee, I. (2017). A framework of spatio-temporal trajectory simplification methods. *International Journal of Geographical Information Science*, 31(6), pp.1128-1153.	Bermingham designed the approach, wrote the code, conducted the experiments, and wrote the paper. Lee directed the paper design, edited the draft, and helped design some of the formal definitions.
4	Under review.	Bermingham designed the approach, wrote the code, conducted the experiments, and wrote the paper. Lee directed the paper design and edited the draft.
5	**[Accepted]** Bermingham, L. and Lee, I. (2017). A Probabilistic Stop and Move Classifier for Noisy GPS Trajectories. *Data Mining and Knowledge Discovery*.	Bermingham designed the approach, wrote the code, conducted the experiments, and wrote the paper. Lee directed the paper design, edited the draft, and suggested models for parameter estimation and significance testing.
6	Under review.	Bermingham designed the approach, wrote the code, conducted the experiments, and wrote the paper. Lee directed the paper design, edited the draft, and suggested some results interpretation in the case study.
7	None	Entirely the work of Bermingham.

TABLE 1.1: Breakdown of intellectual contribution of relevant authors by thesis chapter.

Chapter 2

Literature Review

In this literature review I do not attempt to cover the entirety of trajectory data mining and knowledge discovery; such a task could easily fill a book. Instead, I divide my review of the literature into two parts: a broad overview of some relevant topics in Section 2.1; and, a detailed review of specific approaches in Sections 2.2, 2.4, 2.5, and 2.3. In Section 2.1, I introduce the following topics used throughout this thesis: simplification, stop/move detection, map-matching, place-matching, region-of-interest mining, item-set mining, and sequential pattern mining. In Sections 2.2, 2.4, 2.5, and 2.3, I present a more detailed review of the four topics that I have made contributions to in this thesis: simplification, stop/move detection, place-matching, and sequential pattern mining. I highlight that I do not make any contribution to map-matching or region-of-interest mining in this thesis; I do, however, refer to those approaches in later chapters, so, I provide a brief introduction in Section 2.1: they are then omitted from my more detailed reviews in Sections 2.2, 2.4, 2.5, and 2.3. Additionally, I highlight that I do not provide a background or brief history of the overall field in this section: for that, please refer back to Chapter 1, Section 1.2.

2.1 Breadth Analysis of Trajectory Data Mining

I present Figure 2.1 as an overview for this chapter. Figure 2.1 illustrates a set of topics relevant to this thesis, each categorised under the appropriate steps within the trajectory data mining and knowledge discovery process. Additionally, I repeat this figure at the start of each data-chapter with the relevant topics outlined in bold to visually illustrate their use within that chapter. Again, readers, please note that the topics that are shown in each step of Figure 2.1 are not an exhaustive summary from the field, but rather, a collection of those most relevant to this thesis.

2.1.1 Trajectory Preprocessing

Trajectory preprocessing is the task of modifying, removing, or cleaning raw trajectories to make them more appropriate for later data mining tasks. Common examples include removing noisy entries or projecting coordinates into a suitable coordinate system for mining. Generally, I consider any modification that is applied to raw trajectories prior to mining, a preprocessing task. In this section I introduce and briefly explain two trajectory preprocessing tasks relevant to this thesis: trajectory simplification and trajectory stop/move detection.

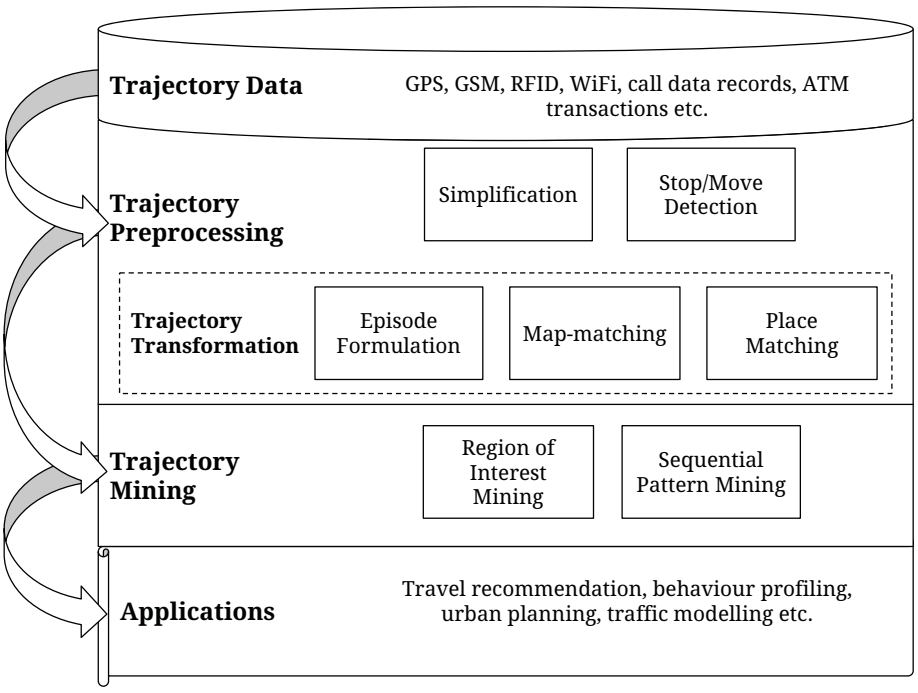


FIGURE 2.1: Overview of the trajectory data mining and knowledge discovery stages and the specific topics (the rectangles) covered in this thesis.

Simplification

Trajectory simplification is the process of reducing the complexity of an input trajectory. Typically, simplification is realised by passing the trajectory into an algorithm that removes specific entries with the goal of preserving the general motion of the tracked entity. Given that it is a practice to capture as much movement detail as possible by oversampling when initially recording trajectories (Long et al., 2013) it follows that simplification is often necessary. Even disregarding overly sampled trajectories, simplification is still useful, in the general cases, for removing redundant entries: doing so directly decreases the processing requirement in the subsequent data mining and knowledge discovery steps.

To simplify a trajectory, a simplification algorithm must utilise some criteria to determine which entries to keep and which to discard. A common criteria used in a number of approaches (Cao et al., 2006; Gudmundsson et al., 2009; Muckell et al., 2011) is the perpendicular distance between the current entry and its neighbours (or trajectory start/end points). A large perpendicular distance not caused by GPS error indicates a large movement, which means that the entry likely defines some motion in the trajectory; a small perpendicular distance indicates little or straight movement, which can be simplified away. Specifically, using this perpendicular distance criteria, a trajectory simplification algorithm can compute the perpendicular distance at each entry, then remove a certain percentage, or all those below a perpendicular distance threshold. To further illustrate trajectory simplification and this example, see Figure 2.2.

In Figure 2.2, entries that form straight segments are removed. This is desirable if the goal is to minimise redundant entries that could be represented by assuming a straight path between two other entries. However, if the goal is to reduce the spatial uncertainty (i.e the noisiness) of the recorded trajectory, then this

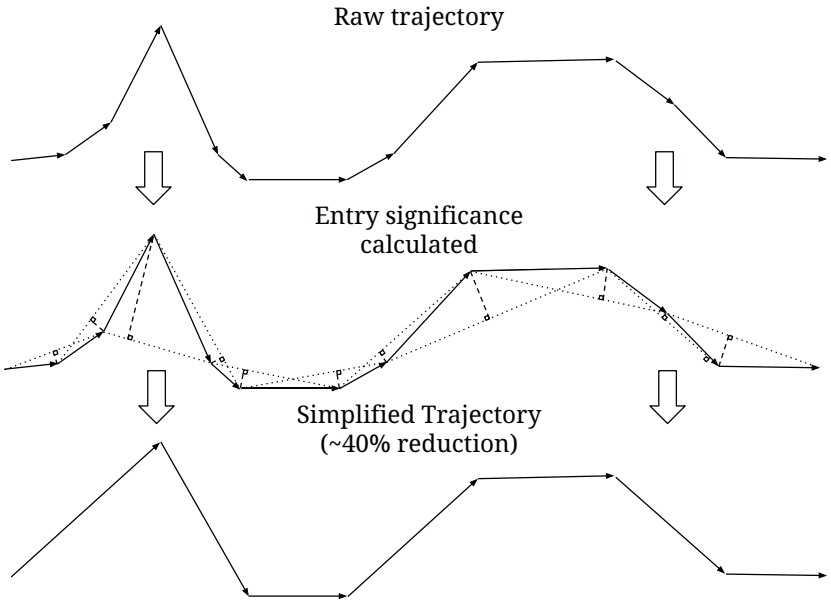


FIGURE 2.2: Example of trajectory simplification using perpendicular distance between an entry and its neighbours as the entry significance criteria.

perpendicular approach is of little use. This highlights an important lesson: there is no such thing as the correct or optimal simplification of trajectory; rather, there are only simplifications that tend towards a certain criteria which may be more-or-less suitable to achieve certain user goals (like reducing redundancy or dampening spatial uncertainty). For example, if the goal is to reduce spatial uncertainty, there are more appropriate simplifications, such as, pruning illogically high speed recordings that could only be explained by GPS inaccuracy (Newson and Krumm, 2009; Atev et al., 2010; Lv et al., 2016).

Stop/Move Detection

Stop/move detection is the process of finding and labelling entries in a trajectory that are either stopping or moving. Stopping entries, particularly those that form a contiguously stopping subsequence for an extended duration, can indicate that the tracked entity was doing some activity or waiting at some specific place. Of course, an extended stop itself is not evidence enough to indicate such things, but when it is paired with context, such as geography, more reasonable inferences can be made. For example, a stop, approximately, in a restaurant for one hour may imply dining. Likewise, moves can also be paired with additional data sources and used to infer meaning. Consider, for example, inferring that a tracked entity caught a train by combining a series of entries labelled as moves with the geometry of a railway network. Moves can also be used more-generally to indicate the activity of transiting from one place to another, which is useful in many applications such as traffic modelling and urban planning applications (Yuan et al., 2015). Thus, trajectory entries labelled as stopping or moving denote some semantic information about the recorded entity and, when paired with contextual information, are a useful step towards knowledge inference.

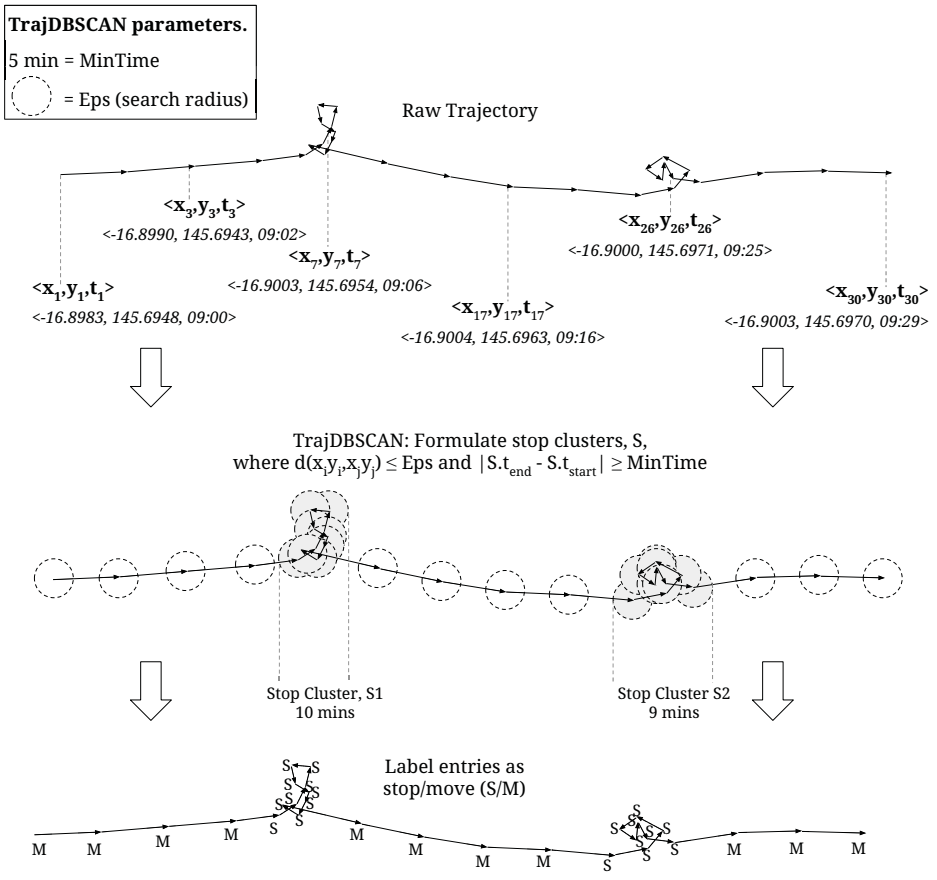


FIGURE 2.3: Example of trajectory stop/detection using TrajDBSCAN by Tran et al. (2011).

To find stops and moves in trajectories, a common approach, adopted by a number of existing works, is to treat it as a binary problem: that is, find the stops first then label everything else as a move (Parent et al., 2013). Thus, the task is then reduced to finding stops in the trajectory. The notion of a stop is defined in multiple different ways in existing works; however, I will attempt to provide a generalisation of a trajectory stop in Definition 5.

Definition 5. Trajectory Stop. A trajectory stop is a contiguous subsequence of entries in a trajectory that are spatially collocated and have a total change in displacement that tends toward zero.

A common notion in existing works is to use density-based concepts to find a cluster of spatially-nearby contiguous entries; if that cluster's total duration surpasses some user-specified minimum duration, then, those entries are labelled as stopping. One such approach is TrajDBSCAN (Tran et al., 2011), which is based on the DBSCAN (Ester et al., 1996) algorithm. In lieu of DBSCAN's $MinPts$ and spatial Eps parameters, TrajDBSCAN has a temporal $MinTime$ and a spatial Eps parameter to find a cluster of contiguous trajectory entries that form a stop cluster. To illustrate the process of stop/move detection, and better explain TrajDBSCAN's algorithm, I present Figure 2.3.

The last step in Figure 2.3 is labelling the entries as stopping or moving. In practice there is often another step, where contiguous subsequences of stops or moves are combined into one stop or move episode (recall, this was demonstrated

in Section 1.1.4). This is done because it is a more efficient representation of the trajectory, and because it is a more-ideal representation to add contextual information to one discrete episode as opposed to many individual entries. However, this highlights an important conceptual distinction I make in this thesis: specifically, trajectory transformation (such as episode formulation) is a pre-processing task, but it is a specialisation that merits its own conceptual step in my version of the trajectory data mining and knowledge discovery process (recall Figure 2.1).

2.1.2 Trajectory Transformation

Transformation is a special category of preprocessing tasks that can be performed on trajectories. I separate transformation from other preprocessing tasks by: defining transformation tasks as those that alter the trajectory data in such a way that it no longer conforms to its original point representation (i.e Definition 1 does not hold). To illustrate, I consider the task of enriching trajectories with stop/move labels a pure pre-processing task because the result is still a sequence of points. However, I do not consider formulating contiguous stop/move entries into an episode with some geometry a pre-processing task. That is a transformation task because the result is no longer a sequence of points, as we can recall from Figure 1.6. Other transformation tasks include converting a trajectory from a sequence of spatio-temporal coordinates into a sequence of roads travelled (map-matching) or places visited (place-matching). Such transformations are useful because, typically, they compress many numerical spatio-temporal coordinates into single, nominal labels: they are far more efficient to mine, and far more semantically meaningful for a human operator to interpret. Map-matching and place-matching are used in the later chapters of this thesis; I provide further explanation below.

Map-matching

Map-matching is the process of fitting raw trajectory recordings onto an underlying structure. The typical use-case of trajectory map-matching is to constrain recorded vehicles to a road network. The first stage of map-matching is strictly a preprocessing task, as the spatio-temporal coordinates of the original trajectory are fitted onto some known structure (i.e a road network). However, due to these underlying structures, themselves, often being semantically meaningful these map-matched trajectories can additionally be transformed into a sequence of semantic labels (i.e a sequence of street names that were driven by the recorded vehicle). Much like episode formulation, map-matching transforms the trajectory into a more succinct format that is more efficient to mine and more semantically meaningful to interpret.

As with many tasks in trajectory data mining and knowledge discovery, map-matching would be trivial if not for varied sampling-rates, the presence of spatial uncertainty, and general inconsistencies in the recorded trajectory. To mitigate these, trajectory map-matching approaches arrive at their result either by: probabilistically considering many potential routes (Newson and Krumm, 2009; Wei et al., 2012); incrementally looking ahead and snapping to road segments based on a geometric measure such as Fréchet distance (Brakatsoulas et al., 2005); or, building a graph of candidate routes based on spatial, temporal, historical, and topological factors, and selecting the best scoring path (Lou et al., 2009; Zheng

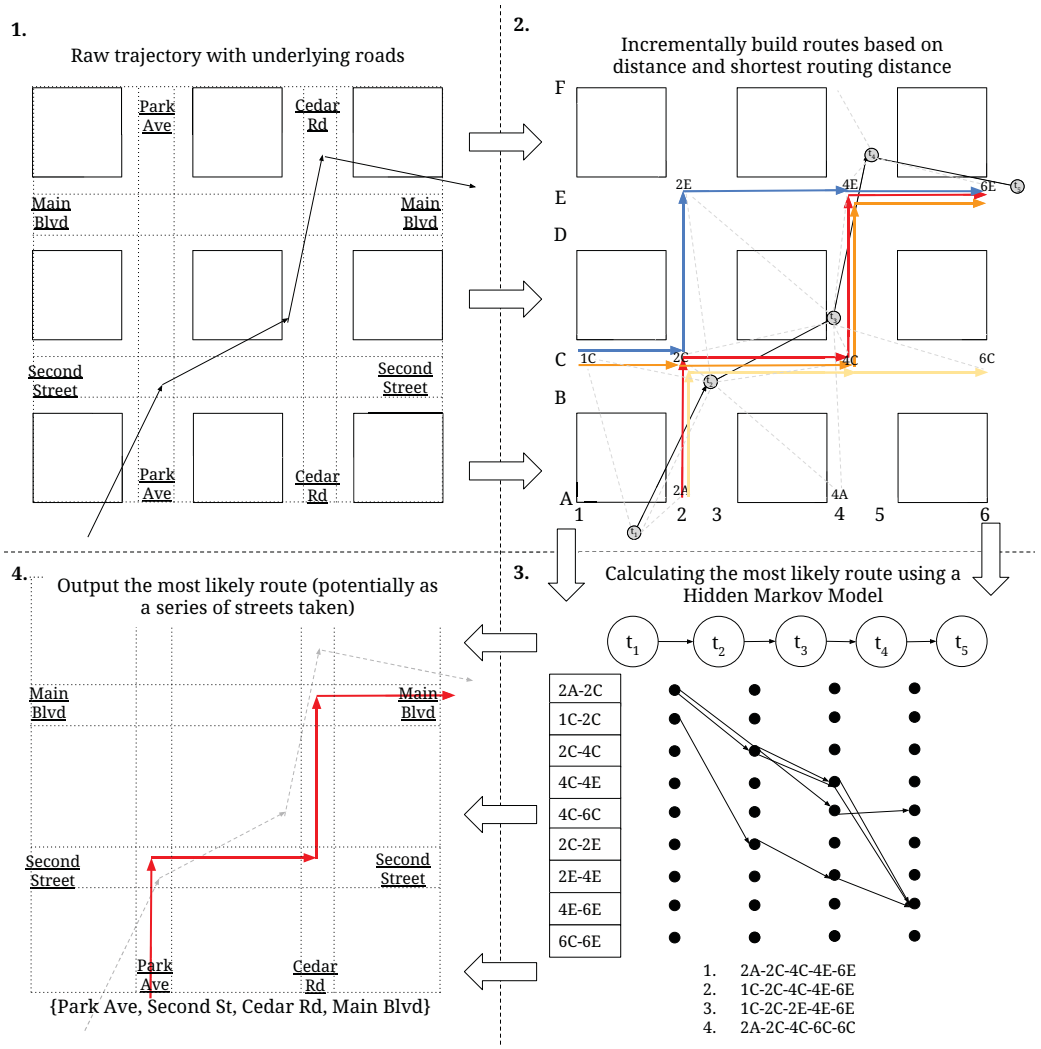


FIGURE 2.4: Example of trajectory map-matching using an approach similar to that of Newson and Krumm (2009).

et al., 2012). My intention is not to compare all existing map-matching approaches as I have no contribution in this area; I do wish, however, to present Figure 2.4 to visually illustrate the probabilistic trajectory map-matching approach by Newson and Krumm (2009) (comprehension of this approach is useful as I refer to it in Chapter 4).

Map-matching does have its limitations. For example, if a recorded entity does not travel along any known structure, map-matching cannot be used. In such cases, episode formulation and place-matching may be more suitable.

Place-matching

Place-matching is the process of associating a trajectory with a series of visited places. As Hightower (2003) says, “manually labelling places does not scale so the research challenge is to automate the process". Automating the process does not mean reverse geo-coding though; as Lv et al. (2016) suggests, “for reverse geo-coding, the obtained semantics is always represented as postal address (e.g., x Road, y City), which is often as challenging to interpret as raw locations". Thus, I

define place-matching as an automatic semantic enrichment task that transforms the journey of the tracked entity into a more human interpretable format of visited places.

Typically, a trajectory is not simply segmented into portions that pass through, or nearby to geographic places; instead, the trajectory is first transformed into a series of stops and moves, and those episodes are matched to underlying places or transport networks (Yan et al., 2013; Furletti et al., 2013). A bonus, is that numerous raw coordinates are transformed into a series of stops, and then, places, which effectively compresses the trajectory while maintaining semantics (Richter et al., 2012). The ultimate result of place-matching is a sequence of visited places and visit times; that output, however, is often further enriched in semantic data mining approaches. For example, place-matching and transport mode inference are both often used as previous steps to infer the types of activities, and overall behaviour profile, of a tracked entity (Beber et al., 2016; Alencar et al., 2015). Consider a trajectory where a person goes to a shopping centre, then to a cultural attraction, and then to an airport. An activity recognition approach may infer shopping, then sightseeing, and then travelling; when put together, might imply that the tracked person was a tourist. Activity inference approaches and automatic behaviour profiling are not areas I make contributions to in this thesis; highlighting that place-matching is, however, often used as input by these approaches, hopefully enforces the importance of accurate place-matching. Put plainly, inaccurate place-matching will lead to inaccurate semantic patterns.

As I mentioned in the map-matching section, semantic enrichment tasks, such as map-matching and place-matching, would be straightforward if not due to varied sampling-rates, the presence of spatial uncertainty, and general inconsistencies in the recorded trajectory. Though, compared to map-matching, the problem of spatial uncertainty and sampling gaps, are perhaps, even more exaggerated when place-matching. This is largely because GPS suffers heavy accuracy degradation when indoors (Mautz, 2009), which is where one can find most visited places. Even more so than map-matching, place-matching approaches must draw on other factors to determine which place was likely visited. One approach, used by some existing works in the literature is to probabilistically model the problem matching a trajectory to likely visited places. Specifically, the model treats a sequence of trajectory stops as observations and nearby candidate places for each stop as states that may have caused these stops (Yan et al., 2013; Lv et al., 2016). The occurrence of each place, with respect to the stop, is given an emission probability; then, the transition between possible places is also given a probability. By taking this approach, one can model the probability of every possible sequence of visited places, and output the most likely sequence as the final place-matching result. To avoid the spatial uncertainty of the raw trajectory/stops, existing approaches have calculated these probabilities using extra information, such as the type of place (Spinsanti et al., 2010); place opening hours, with regard to the stop time (Furletti et al., 2013); and, the likelihood of certain types of place-to-place transitions (Yan et al., 2013). In this way, these probabilistic place-matching approaches determine the likely sequences of visited places without dealing with unreliable spatial information, and, instead, use the more reliable semantic, temporal, and sequential information to infer a likely sequence of visited places. Additionally, the probabilistic models naturally incorporate the sequential aspect of the trajectory data when calculating the result; many other approaches simply perform matching one stop at a time, which does not incorporate the decision making or routine behaviour of the tracked person who

goes from one place to another. In other words, people do not arbitrarily go from place-to-place; therefore, during place-matching, we should assume that certain types of place-to-place visitations are more likely than others.

To conclude this brief explanation of place-matching, I present Figure 2.5 which illustrates the general procedure of a probabilistic place-matching approach, such as those presented by Yan et al. (2013) and Lv et al. (2016). The details of how the stop/place probabilities and the place-to-place transition probabilities are calculated, in Figure 2.5, or any place-matching approach, are specific to each approach and are broadly the main differentiating factors between approaches. In fact, disregarding the emission and transition probability calculations, I consider Figure 2.5 to serve as a representation of a general probabilistic place-matching approach. Additionally, note that in Figure 2.5, the emission probabilities are notated as using the stops and actual places (i.e. P(A | S1)), whereas the transition probabilities are notated using the place types (i.e. P(Bank | Shop)). This is done purposefully to illustrate that probabilistic place-matching approaches may calculate probabilities based on the actual places, their types, or some combination of both (as seen in the example). Lastly, the reason I choose to illustrate a probabilistic approach, as opposed to some other paradigm, is because I use a probabilistic approach in my place-matching approach in Chapter 6.

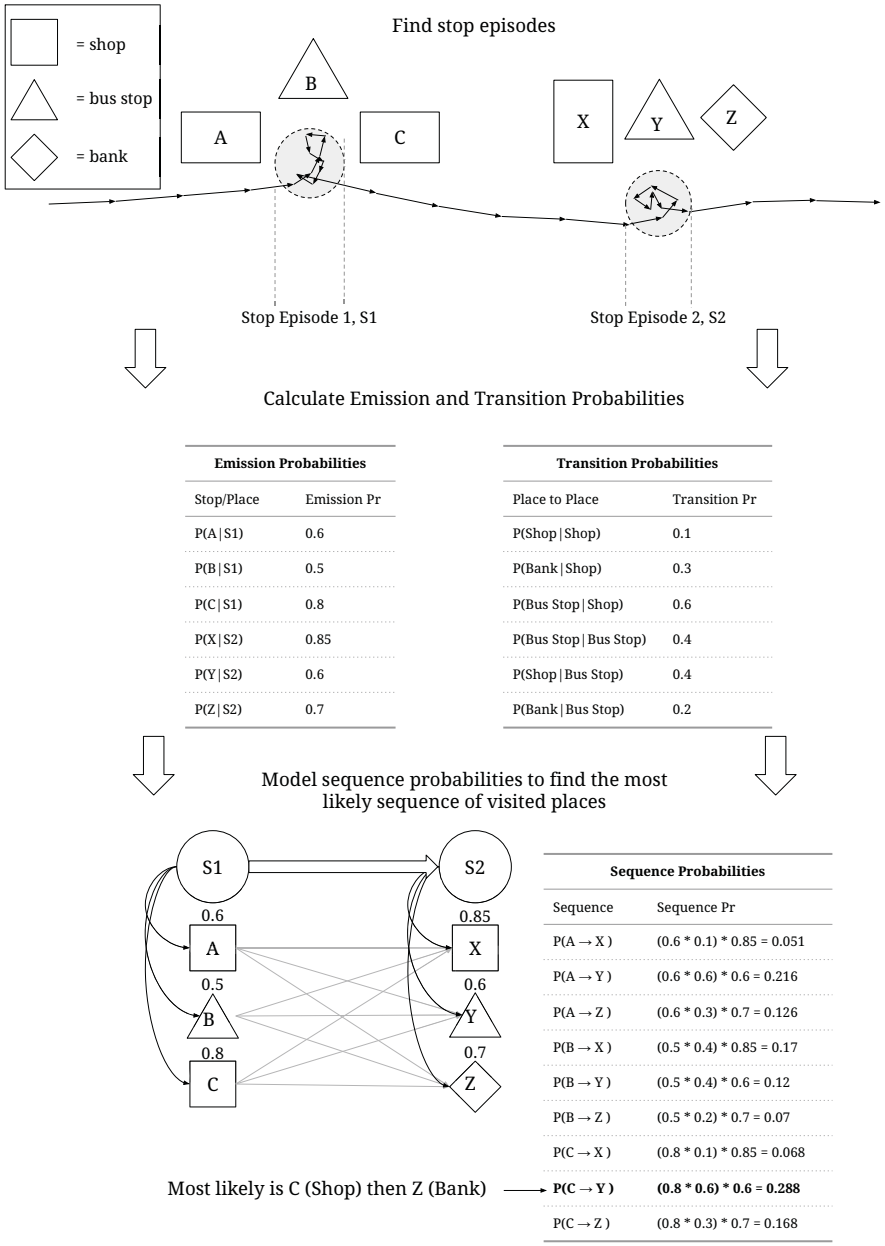
2.1.3 Trajectory Mining

Broadly, I consider trajectory mining as any task that operates on trajectory data and produces patterns or results that can lead to knowledge discovery. A few common trajectory data mining tasks include: clustering, outlier detection, classification, identifying moving group patterns, and inferring origin-destination networks (Zheng, 2015). However, in this review section, I limit myself to two approaches relevant to the later chapters, specifically: trajectory region-of-interest mining and sequential pattern mining.

Trajectory Region-of-Interest Mining

First, I highlight that the spatial boundary that encompasses a trajectory dataset is called the study region. Given the study region, the task of trajectory region-of-interest (RoI) mining is to find all frequently visited sub-regions, within the study region. The definition and criteria to formulate these RoIs is what differentiates one trajectory RoI mining approach from another. For example, Giannotti et al. (2007) presents an approach that partitions the study region into uniform grid cells, traces the trajectory through each cell to determine the cell density, then, expands the cells to form RoIs, in a rectangular way, until the average RoI density would fall below a user-specified threshold if any further expansion occurred. Other examples include my previous works: where arbitrary shaped RoIs are produced by partitioning the study region using a delaunay triangulation (Bermingham et al., 2014); and another, that divides the dataset into spatio-temporal 3d grid cells and finds spatio-temporal volumes of interest (Bermingham and Lee, 2014).

In general, discovering visited RoIs in trajectory data is useful because it highlights popular areas based purely on the original data; there is no need for any external data sources to find patterns. Additionally, transforming a raw trajectory, which is simply a sequence of spatio-temporal coordinates, into a sequence of visited RoIs, is also an effective way to discretise the trajectory so that it can be used in further mining tasks, such as sequential pattern mining (Giannotti et al., 2007)



Emission Probabilities

Stop/Place	Emission Pr
P(A | S1)	0.6
P(B | S1)	0.5
P(C | S1)	0.8
P(X | S2)	0.85
P(Y | S2)	0.6
P(Z | S2)	0.7

Transition Probabilities

Place to Place	Transition Pr
P(Shop | Shop)	0.1
P(Bank | Shop)	0.3
P(Bus Stop | Shop)	0.6
P(Bus Stop | Bus Stop)	0.4
P(Shop | Bus Stop)	0.4
P(Bank | Bus Stop)	0.2

Sequence Probabilities

Sequence	Sequence Pr
P(A → X)	(0.6 * 0.1) * 0.85 = 0.051
P(A → Y)	(0.6 * 0.6) * 0.6 = 0.216
P(A → Z)	(0.6 * 0.3) * 0.7 = 0.126
P(B → X)	(0.5 * 0.4) * 0.85 = 0.17
P(B → Y)	(0.5 * 0.4) * 0.6 = 0.12
P(B → Z)	(0.5 * 0.2) * 0.7 = 0.07
P(C → X)	(0.8 * 0.1) * 0.85 = 0.068
P(C → Y)	**(0.8 * 0.6) * 0.6 = 0.288**
P(C → Z)	(0.8 * 0.3) * 0.7 = 0.168

FIGURE 2.5: Example of trajectory place-matching using a probabilistic approach.

and place-matching (Yan et al., 2013). To further illustrate the process of RoI mining, I present Figure 2.6, which demonstrates a RoI mining approach introduced by Giannotti et al. (2007).

Sequential Pattern Mining

Given a sequence database, sequential pattern mining is the task of discovering all sub-sequences with occurrences greater-than or equal-to some user-specified threshold. Trajectories are sequential in nature; it follows that sequential pattern mining may reveal interesting trends about popularly visited routes, or even reveal distinct user groups within a trajectory dataset. However, trajectories are not naturally suited to sequential pattern mining, as the sequences used in sequential pattern mining must consist of discrete items (geographic coordinates are far to granular to make such comparisons). Fortunately, various authors have proposed different techniques for discretising trajectories for sequential pattern mining. For example, Song et al. (2014) transformed trajectories into a sequences of visited road nodes using map-matching; Giannotti et al. (2007) transformed trajectories into sequences of region visitations. Thus, following some form of discretisation, trajectories can be mined using sequential pattern mining algorithms and potentially reveal previously unknown movement patterns present in the dataset. I now present a series of definitions to explain the various types of sequential pattern mining —all of which are referred to and used in Chapter 4.

Definition 6 (Items). Let $I = \{a_1, a_2, \ldots, a_n\}$ be a set of items. An item is represented as an integer or character.

Definition 7 (Sequence). A sequence S is an ordered list of items, $\langle a_i, a_j, \ldots, a_m \rangle$, where a_k is an item in I for $i \leq k \leq m$ and $i < j < m$.

This broad definition of a sequence means that many different types of data are candidates for sequential pattern mining. Some examples within the field include retail transactions (Srikant and Agrawal, 1996), nucleic acid sequences (Exarchos et al., 2008), and, in this case, discretised trajectories. For readers familiar with sequential pattern mining, you may note a sequence is often defined as a list of item sets. This is useful for some datasets where multiple items can occur in a sequence simultaneously. For example, in retail transactions a customer can buy $\{bread, milk, apple\}$ in one transaction, then later, $\{juice, chocolate\}$. However, in the context of mining trajectories, one can assume no entity can be in two places at once and therefore define sequences as ordered lists of single items.

Definition 8 (Sequence Containment). A sequence $S_a = \langle a_1, a_2, \ldots, a_n \rangle$, is said to be *contained* in a sequence $S_b = \langle b_1, b_2, \ldots, b_m \rangle$ iff there exist integers $1 \leq i_1 < i_2 < \ldots i_n \leq m$ such that $a_1 = b_{i_1}, a_2 = b_{i_2}, \ldots, a_n = b_{i_n}$ (denoted as $S_a \sqsubseteq S_b$).

Additionally, I highlight that if S_a is *contained* in S_b, then S_a is a *sub-sequence* of S_b, and, by extension, S_b is a *super-sequence* of S_a.

Definition 9 (Sequence Database). A sequence database is a list of sequences, $SDB = \langle S_1, S_2, \ldots, S_n \rangle$.

Typically a sequence database is simply a plain-text file where there is one sequence per line and each item in the sequence is consistently delimited.

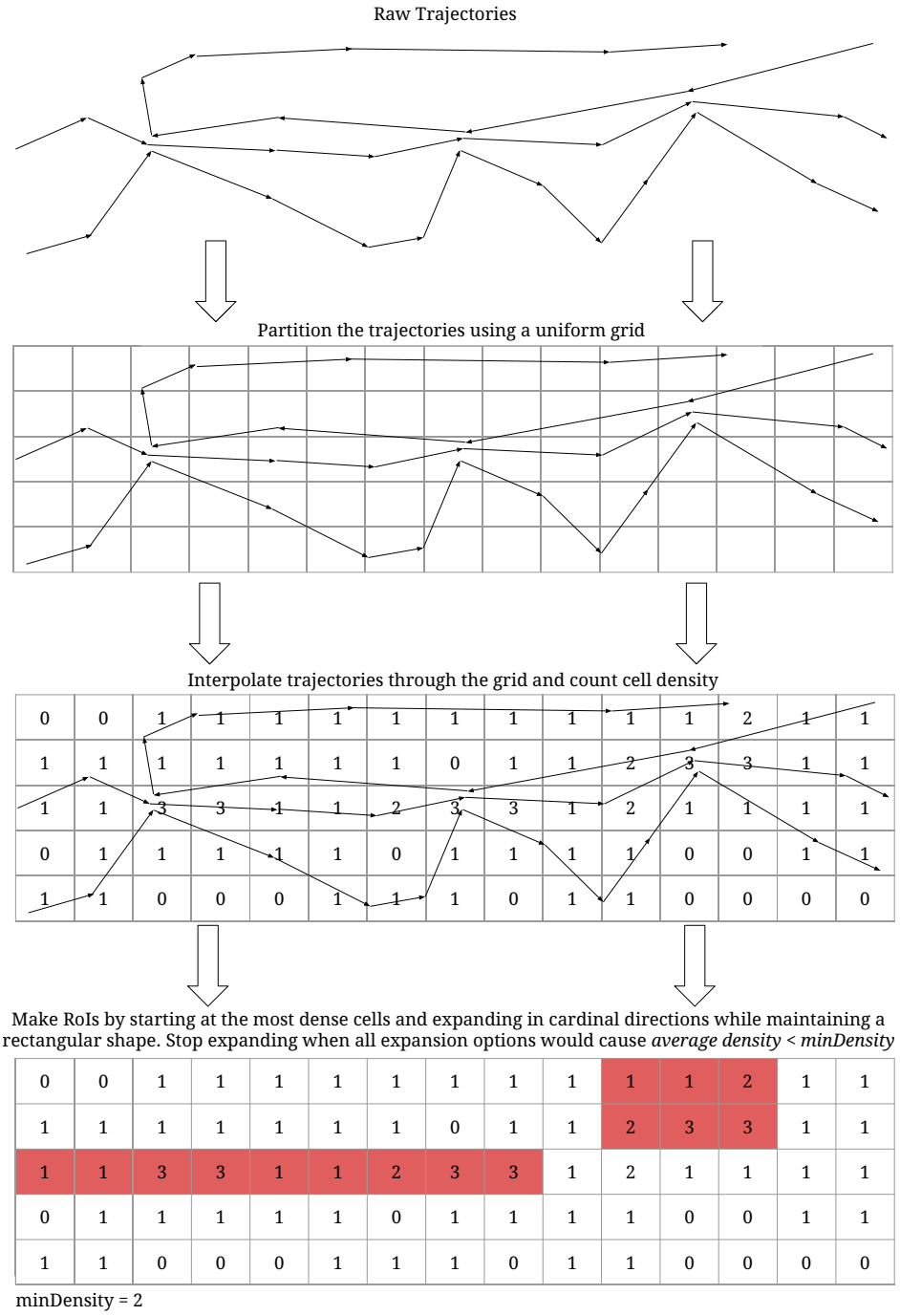


FIGURE 2.6: Example of a trajectory RoI mining approach.

Definition 10 (**Sequence Support**). Given a sequence S_a its support is the number of sequences in a sequence database SDB that *contain* S_a. Finding the support of a sequence S_a is denoted as $sup(S_a, SDB)$ and for a sequential pattern that has its support stored it is denoted $sup(S_a)$.

Support is typically used to find frequently occurring sequences within the sequence database, or, in other words, frequent sequential patterns.

Definition 11 (**Sequential Pattern**). Given a user specified minimum support threshold $minSup$ and a sequence database SDB, a sequence S_a is considered a sequential pattern if $sup(S_a, SDB) \geq minSup$.

A sequential pattern is usually output as a sequence with its support value, like so, $\{a, b, c\}$ [SUP:10]. A sequential pattern represents a frequently occurring trend within a sequence database. Automatically identifying such trends through sequential pattern mining is useful because it can lead to knowledge discoveries which would be extremely time consuming and tedious for a human to identify manually. Examples of algorithms that mine sequential patterns are GSP (Srikant and Agrawal, 1996), SPADE (Zaki, 2001), SPAM (Ayres et al., 2002), PrefixSpan (Pei et al., 2004), LAPIN (Yang et al., 2007), and CM-SPAM (Fournier-Viger et al., 2014a).

In large sequence databases that contain many long sequences, it is common to uncover a truly massive number of sequential patterns. This is because a large sequential pattern contains a combinatorial number of smaller sub-patterns. The example given by Pei et al. (2000) clearly illustrates the problem with discovering all sequential patterns. Consider, a sequential pattern of length 100, $\{a_1, a_2, \ldots, a_{100}\}$, it contains $\binom{100}{1} = 100$ length 1 sub-patterns: $\{a_1\}, \{a_2\}, \ldots, \{a_{100}\}$; $\binom{100}{2}$ length 2 sub-patterns: $\{a_1, a_2\}, \{a_1, a_3\}, \ldots, \{a_{99}, a_{100}\}$; and so on. The total number of sub-patterns the length 100 sequential pattern contains is:

$$\binom{100}{1} + \binom{100}{2} + \cdots + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}. \tag{2.1}$$

This is a truly huge number of sequential patterns — far too many to compute, let alone meaningfully interpret. A common solution that one may employ to reduce the pattern output, is to increase the $minSup$ parameter. This will require candidate sequences to occur in more sequences from the database to become patterns. Increasing $minSup$ does, however, also increase the likelihood that important sequential patterns will be missed during mining.

Another approach for reducing the pattern output is to use a mining approach that mines a *concise representation* of the sequential patterns. "A concise representation is a subset of all sequential patterns that is meaningful and summarizes the whole set of sequential patterns" (Fournier-Viger et al., 2017). Concise representations come in two varieties, *lossless* and *lossy*. The pattern output is lossless if the set of all sequential patterns (with their support scores) can be recovered without scanning the sequence database; the pattern output is lossy if the set of all sequential patterns cannot be recovered without scanning the database. Two common concise representations are closed patterns and max patterns.

Definition 12 (**Closed pattern**). Given a set of all sequential patterns AS, a sequential pattern S_a is *closed* iff $S_a \in AS \wedge \nexists S_b \in AS$ such that $S_a \sqsubset S_b \wedge sup(S_a) = sup(S_b)$.

The set of all closed patterns is denoted CS and $CS \subseteq AS$. Closed patterns considerably reduce the pattern output by ensuring that for every sequential

pattern in the output there exists no sub-pattern in the output with the same support. Additionally, because of this rule closed patterns are *lossless* (Fournier-Viger et al., 2017). Examples of algorithms that mine closed sequential patterns are CloSpan (Yan et al., 2003), BIDE (Wang and Han, 2004), and ClaSP (Gomariz et al., 2013).

Definition 13 (**Max pattern**). Given a set of all sequential patterns AS, a sequential pattern S_a is *maximal* iff $S_a \in AS \wedge \nexists S_b \in AS$ such that $S_a \sqsubset S_b$.

The set of all max patterns is denoted MS and $MS \subseteq CS \subseteq AS$. The set of all max patterns is generally even more concise than the set of all closed patterns. This is because max patterns discard many redundant sequential patterns by ensuring that no pattern in the output is a sub-pattern of any other. The trade-off for reducing redundancy in this way, is that max patterns are *lossy* (Fournier-Viger et al., 2017). Examples of algorithms that mine maximal sequential patterns are SPEED (Raissi et al., 2006), FMMSP (Lin et al., 2007), and VMSP (Fournier-Viger et al., 2014c).

Definition 14 (**Contiguously Contained**). A sequence $S_a = \langle a_1, a_2, \ldots, a_n \rangle$, is said to be *contiguously contained* in a sequence $S_b = \langle b_1, b_2, \ldots, b_m \rangle$ iff there exist integers $1 \leq i, i+1, i+2, \ldots, i+n-1$ such that $a_1 = b_i, a_2 = b_{i+1}, a_3 = b_{i+2}, \ldots$, and $a_n = b_{i+n-1}$.

Contiguous containment is an additional constraint that is applied to sequential pattern mining to reduce the pattern output. Additionally, it is also used to discover specific patterns that appear in contiguous chunks in the underlying sequence database. Sequential patterns that are discovered using the contiguous constraint are called *contiguous sequential patterns*. Contiguous sequential pattern mining algorithms are, basically, also a special case of gap-constrained sequential pattern mining, where the size of the allowed gap between items is set to one. Examples of algorithms that can mine contiguous sequential patterns include GenPrefixSpan (Antunes and Oliveira, 2003a), CCSM (Orlando et al., 2004), Gap-BIDE (Li and Wang, 2008), and CC-SPAN (Zhang et al., 2015).

Readers, please note, that for brevity I will not define the contiguous version of all the different types of sequential patterns (Definitions 11, 12, and 13) and instead, when I refer to the contiguous versions of these patterns, I ask the reader to keep in mind a small change to Definition 10 (calculating the support of a sequence): replace "*contained*" (Definition 8) with "*contiguously contained*" (Definition 14). Making this change, the definitions for the different types of sequential patterns (Definitions 11, 12, and 13) all hold, and now define, their respective contiguous versions.

2.2 Depth Analysis of Trajectory Simplification Approaches

In this section I review trajectory simplification approaches, which are also called trajectory data reduction, trajectory sampling, or trajectory compression approaches in the field (Zheng, 2015). To begin my review of trajectory simplification approaches, I define the limits of the review. Most importantly, I am only interested in works that reduce trajectory complexity by discarding some redundant, or less significant, entries. Therefore, I will not consider smoothing approaches like Kalman filters (Lee and Krumm, 2011) or spline fitting approaches (Marino and Manic, 2016) that modify the trajectory coordinates; I will not consider map-matching-based approaches (recall Section 2.1.2) that snap the coordinates to some underlying structure and simplify the journey as a path

between nodes. Furthermore, I will not consider domain-specific trajectory simplification approaches, such as those that specifically require vehicle trajectories (Kellaris et al., 2009; Richter et al., 2012; Liu et al., 2014; Song et al., 2014).

2.2.1 Overview

With the scope of the trajectory simplification review now defined I present the features that I use to compare each approach.

Attributes. In general, I assume that trajectories being simplified are spatio-temporal. This means that there are, at the very least, spatial and temporal attributes to consider; additional attributes, however, such as speed and direction, are also commonly inferred and considered. Due to varied sampling-rates, recording gaps, and general inconsistency of spatio-temporal trajectories, an effective trajectory simplification approach should consider the temporal dimension in some way. In fact, there is already a whole class of algorithms used in cartography, GIS, and computational geometry that only consider simplifying poly-lines using spatial attributes (Shi and Cheung, 2006); however, many authors (Cao et al., 2006; Lange et al., 2008; Potamias et al., 2006) consider these to be not entirely suitable for simplifying spatio-temporal trajectories, as they ignore the temporal dimension. Thus, to differentiate themselves from poly-line simplification approaches, trajectory simplification approaches should, ideally, use some combination of features beyond the purely spatial. Ultimately, the way attributes are considered by each simplification approach is what determines the simplification result; thus, I consider comparing the attributes used an effective feature to differentiate various approaches.

Entry Significance. Simplifying trajectories requires reducing the size of the original trajectory by somehow removing entries. Thus, each simplification approach must have some function or criteria to determine which entries to keep and which to discard. In other words, the significance of each entry, with regard to some simplification goal, must somehow be calculated by each simplification approach. For example, the significance of an entry may be calculated as the change in speed and heading it caused, compared to the previous entry, with greater scores being considered more significant. Overall, each approach is largely defined by the significance scoring function it uses, and this makes the type of significance scoring function an effective feature for comparing different approaches.

Processing Strategy. I define the processing strategy of a simplification approach as: how it processes entries, when entry significance is calculated, how many other entries will be considered when calculating a single entry's significance, and what is done when an entry is simplified out of the trajectory. Thus, it follows that the processing strategy is also what defines the type of use-cases each simplification approach is suited to. Some use-cases call for fast processing times and little memory overhead because the algorithm is required to run *online* (i.e in real-time), often on a constrained device. In online cases, a common processing strategy is to maintain a small buffer of recent entries and calculate entry significance as new data arrives, discarding entries that are too insignificant, and storing tail entries that are significant enough. In other use-cases the whole trajectory is already available, and processing time is less important than low-error simplifications, so simplification approaches can run *offline* (i.e on a PC with the whole dataset passed as input). In offline cases, the processing strategy used can be far more inefficient: sometimes performing a full scan of the trajectory

to find the most significant entry, storing that, then repeating the process to find the next most significant entry. Thus, I consider the processing strategy another effective feature to compare simplification approaches.

I present my overview of relevant trajectory simplification approaches in Table 2.1, which is then followed by a discussion of the unique contributions of each reviewed approach.

1. Meratnia and By (2004) present one of the earliest works directly focusing on simplification of spatio-temporal trajectories, and, as such, they are possibly the first to establish the similarity between this problem and the problem of poly-line simplification that has been tackled in cartography by Douglas and Peucker (1973). Meratnia and By (2004) introduce the notion of synchronised time-ratio distance, which is a distance function considering space and time. Meratnia and By (2004) explain how this measure is important for spatio-temporal trajectory simplification because it considers both space and time dimensions of the problem; traditional poly-line simplification approaches, such as Douglas and Peucker's 1973, and Visvalingam and Whyatt's 1993, only consider spatial/geometric features. In general, I argue that synchronised time-ratio distance is an important early foundation for the field because it is later independently introduced by several authors (Cao et al., 2006; Potamias et al., 2006); ultimately synchronised time-ratio distance ends up becoming a common trajectory simplification error metric, under the name synchronised Euclidean distance. Meratnia and By (2004) also introduce the notion of preserving derived speed during compression, as this naturally incorporates space and time, too. Following this, Meratnia and By (2004) present a taxonomy of strategies for simplification: they introduce, specifically, two essentially split-based approaches (called top-down and bottom-up), and two single-pass approaches (called fixed window and opening window). They combine the top-down and opening window strategies with synchronised time-ratio distance and derived speed to create a number of approaches that they quantitatively compare against the so-called Douglas-Peucker poly-line simplification algorithm (Douglas and Peucker, 1973). Despite being an early work, I highlight that this is still one of the only works to generalise the problem of trajectory simplification and make several simplification algorithms by combining various entry significance functions (i.e. synchronised time-ratio and speed) and processing strategies (i.e. top-down, opening window etc.).

2. Cao et al. (2006) present an approach, that at a high-level, conceptual view, is quite similar to that of Meratnia and By (2004): both approaches borrow the Douglas-Peucker poly-line simplification algorithm (Douglas and Peucker, 1973) and apply it to the problem of trajectory simplification. Additionally, Cao et al. (2006) also introduce four distance functions (two spatio-temporal) that they use to measure trajectory simplification error. Similarly to Meratnia and By (2004), Cao et al. (2006) combine these various distance functions with the Douglas-Peucker algorithm, to make several algorithms. Notably, it appears that one of the distance functions Cao et al. (2006) introduces time uniform distance, which is equivalent to the synchronised time-ratio distance from (Meratnia and By, 2004), which, as I mentioned above, is later popularly used by the field as an error metric under the name synchronised Euclidean distance. I argue that the most novel contribution of Cao et al. (2006) is that they extensively investigate the soundness of various queries after trajectory

Approach	Attributes	Entry Significance	Processing Strategy
1. Meratnia and By (2004)	S,T,SP	SED and Speed	Split-based/Single-pass (online)
2. Cao et al. (2006)	S,T	Perpendicular distance	Split-based
3. Potamias et al. (2006)	S,T	SED	Single-pass (online)
4. Lee et al. (2007)	S	Distance, Angle, Perpendicular distance	Single-pass
5. Lange et al. (2008)	S	Distance	Single-pass (online)
6. Chen et al. (2009)	S,D	Distance, Angle	Split-based
7. Gudmundsson et al. (2009)	S,T	Distance	Split-based
8. Muckell et al. (2011)	S,T	SED	Single-pass (online)
9. Liu et al. (2013)	S	Area	Split-based (online)
10. Long et al. (2013)	D	Angle	Single-pass
11. Liu et al. (2015)	S	Distance	Single-pass (online)
12. Lin et al. (2016)	S,T	Perpendicular distance	Split-based

TABLE 2.1: Overview of relevant trajectory simplification literature. Parameter legend: S = spatial, T = temporal, D = directional, SP = speed.

simplification using their various distance functions. I speculate that perhaps it was their investigation and conclusions about the various distance functions that led to the popularity of synchronised Euclidean distance in the field today.

3. Potamias et al. (2006) present "STTrace", which is one of the earliest online trajectory simplification approaches. Additionally, Potamias et al. (2006) introduce a spatio-temporal distance function for simplification approaches: an entry under consideration is projected back onto the simplified trajectory by linearly interpolating along the relevant line segment using the entry's temporal component. Finally, the distance is calculated as the Euclidean distance between the entry under consideration and its projected counterpart. This is the same distance function introduced independently by both Meratnia and By (2004) and Cao et al. (2006); however, in this work it is called synchronised Euclidean distance (SED): that is the name that ends up being used in the field. STTrace functions, by storing a small buffer of recent entries, which are used to calculate the SED of new incoming entries. This SED calculation determines whether or not the incoming entry should be added to the buffer or whether some entry already in the buffer should be discarded in its place. STTrace's main contribution, is that it is an online approach that guarantees a user-specified trajectory size requirement will not be exceeded during simplification; this is useful for memory constrained devices running in the field.

4. Lee et al. (2007) introduce a novel simplification approach to aid in their goal of trajectory clustering. Their simplification approach is based on the minimum description length principle (Rissanen, 1978) [1], where the length in bits, of a hypothesis is calculated $L(H)$, the length, in bits, of applying that hypothesis to the data is calculated $L(D|H)$, and then, the hypothesis that is chosen, is the one that minimises $L(H) + L(D|H)$. In this case, Lee et al. (2007) process the data incrementally starting from the first entry and modelling the hypothesis as the log of the distance between the start, C_j, and end, C_{j+1}, entries of a currently processed trajectory data. Then, $L(D|H)$ is calculated as the sum of the log of the perpendicular distance, and the log of the angular distance between each line segment and the line segment formed by $\overline{C_j, C_{j+1}}$. I highlight that this approach simplifies the trajectory, based purely on the data, which is in contrast to other approaches, where the user must input an error tolerance parameter, or a number of points to preserve. While data-driven, this is not necessarily ideal because often a user wishes to achieve a certain level of compression, or, at least, to maintain a certain error-bound. Although it is possible to tune this approach by introducing a parameter that offsets the $L(D|H)$ sum, I highlight that this sum, and hence the offset, is in bits, which makes setting such an offset non-intuitive for a user. However, I do find the fusion of different distance measures (perpendicular and angular distance), by a log function, to be a novel approach that is not further-explored in later approaches.

5. Lange et al. (2008) introduce two online trajectory simplification algorithms, "CDR" and "CDR^M", that are both based on the concept of dead-reckoning. Dead-reckoning is typically used for navigation to determine one's current

[1] For more details on MDL and its formulation I refer readers to the tutorial by Grünwald (2005).

position using a previously determined position. In this context, it is adapted by Lange et al. (2008) to determine the simplification of a trajectory based on previous positions and estimated headings. Basically, a small buffer of previously received locations is kept and used to calculate an approximate direction of the next position update. When the position update arrives, it is compared against the approximated direction to calculate the deviation: if that deviation surpasses a user-specified threshold then the update is added to the buffer. The CDR algorithm ensures that the a user-specified error-bound is not surpassed, and CDR^M ensures that the error-bound is kept as low as possible without the buffer surpassing a user-specified size constraint. In later works (Lange et al., 2009) these authors extend this online trajectory simplification scheme to use metrics from existing line simplification approaches, such as the Douglas-Peucker algorithm (Douglas and Peucker, 1973). I consider the main contribution of this work to be the formulation of both an error-bounded online method and a size-bounded method, whereas most approaches can only achieve one or the other.

6. Chen et al. (2009) present one of the earliest works to consider simplifying trajectories to preserve some semantic aspects of the data. In this case, the author's goal is to preserve interactions and visitations within a location-based social network. Specifically, the algorithm presented divides the trajectory into a series of walk and non-walk segments, based on velocity. Then, each segment is simplified using a combination of direction change and speed; the authors claim that bigger distances between entries, and bigger changes in heading, are more semantically meaningful for their purpose. I argue that these attributes are not as semantically meaningful as preserving visited places or even stops; however, this approach is interesting nonetheless, simply because it is one of the earliest to introduce this alternate trajectory simplification goal of semantic preservation.

7. Gudmundsson et al. (2009) present an approach that is entirely focused on examining and extending the four distance measured introduced by Cao et al. (2006) for trajectory simplification. Gudmundsson et al. (2009) present a formulation of their own distance measure, where time units are mapped to space units using a conversion parameter. Using this model they demonstrate that Euclidean distance and SED are two extremes of their function and they can therefore exploit trade-offs in favouring one dimension or another. Similarly to (Cao et al., 2006), after they have introduced their distance measure, they apply it to propose a modified version of the Douglas-Peucker line simplification algorithm (Douglas and Peucker, 1973) (however experimental results are lacking). I consider this work distinct from others, as it provides a straightforward approach for extending other geometric simplification approaches to the problem of trajectory simplification, although, the authors do not capitalise on this opportunity in this work.

8. Muckell et al. (2011) introduce an online trajectory simplification algorithm called "SQUISH". SQUISH incrementally receives entries into a buffer of user-specified size. When the buffer is full, SQUISH calculates which entry would introduce the least SED between the buffered trajectory and the streamed trajectory, if removed. Initially each entry has an attached score associated with it, describing how much SED would be introduced if it were removed from the buffer. However, in subsequent removals following this

initial calculation, only the entries neighbouring the removed entry need to have their SED updated (as other entry's SED scores will not be impacted). To efficiently facilitate this sort of updating, entries are stored in a heap alongside their SED scores. In contrast to most online trajectory simplification approaches, SQUISH's final result tends towards a user-specified size; however, the size of stream cannot be known ahead of time. Therefore, to avoid an oversimplification, many other online approaches prefer to have an unbounded buffer and instead maintain a user-specified maximum allowable error. I argue that most of the concepts introduced in this approach are the same as those introduced by Meratnia and By (2004) and Potamias et al. (2006); thus, I would expect a performance comparison to identify the unique contribution of the work. However, such a comparison is lacking. Fortunately, SQUISH is extended to "SQUISH-E" in an additional work by the same authors, such that the SED error is bounded (Muckell et al., 2014) and a performance comparison is provided. The results demonstrate that at the same compression rates, "SQUISH-E" achieves faster running times and less error than similar approaches.

9. Liu et al. (2013) introduce an online simplification approach that uses a novel entry significance metric: area. The approach begins by first partitioning the trajectory into a series of sub-trajectories using minimum bounding rectangles (MBRs) containing every n-entries. The MBRs are then split and merged using a set of five heuristics, so that MBRs all tend towards having equal areas. Then, each sub-trajectory is simplified by using these heuristics to determine which entries in each MBR should be preserved. These heuristics are not necessarily justified by any distribution or trend in the data; thus, I question their effectiveness. In some sense they are justified by the experimental results, because the results demonstrate the approach outperforms the Douglas-Peucker line simplification algorithm in terms of error; however, comparisons to similar online trajectory simplification approaches are lacking: the justification is limited. Overall, other than the area-based significance scoring technique, this approach seems similar to other online trajectory simplification algorithms.

10. Long et al. (2013) introduce "DPTS", which is the first specifically direction-preserving trajectory simplification approach. DPTS is bounded to ensure no more than a user-specified amount of directional error is introduced during simplification of the trajectory. The authors also claim this approach reduces positional error, while preserving directionality, which, they say, is a feature purely positional-error reducing approaches (like the Douglas-Peucker approach) do not have. In their experiments the authors also demonstrate this claim; however, the effectiveness of the approach is only compared against the Douglas-Peucker line simplification algorithm and not more equivalent contemporary trajectory simplification approaches. I highlight that, in an extension of their work (Long et al., 2014), the authors introduce an approach that does not bound error because the direction tolerance parameter is difficult to know ahead of time. Instead, they ask the user to specify the amount of compression (i.e. the number of entries to preserve) and then their approach simplifies the trajectory accordingly while minimising direction error. Unfortunately, the extended work also lacks a comparison to contemporary trajectory simplification approaches; however, I

still consider their formulation for scoring entries based on direction a unique contribution.

11. Liu et al. (2015) introduce an online trajectory simplification approach that use a tolerance parameter to ensure a bounded amount of error. This approach is unique because it is able to process newly received entries in $O(1)$ time. Briefly, the algorithm builds up a buffer of entries, and the error of the current buffer is the maximum distance between any entry in the buffer and the line segment formed by the start-and-end entries of the buffer. When a new entry is received, it redefines the line segment and the error of the buffer must be recalculated. If the error exceeds the user-specified tolerance, that line segment is stored as part of the simplification, and the in-between entries are discarded. This error recalculation occurs as each new entry is received; however, the authors introduce a novel technique to reduce this calculation to O(1) by dividing the space of the buffered entries into quadrants and using some convex hull calculations. I consider this approach a unique contribution to online trajectory simplification approaches that must run on highly constrained platforms.

12. Lin et al. (2016) introduce a split-based approach that performs two stages of splitting. In the first stage, the whole trajectory is divided into partitions based on velocity. This is to ensure local velocity information is preserved for tasks such as transport mode inference. Then, each of these partitions is simplified using an estimated error tolerance parameter, passed into the Douglas-Peucker line simplification algorithm. Lastly, the simplified sub-trajectories are combined together to form the overall simplified trajectory. The aim of this process is to simplify the trajectory adaptively by considering the local (partitioned) variances in position and velocity. I highlight this approach as unique from many others because it makes this consideration part of its processing strategy, as opposed to its entry scoring function. Additionally, it is one of the only approaches to estimate the error tolerance variable for the Douglas-Peucker algorithm. Ultimately, this means that unlike other approaches, that give the user an error tolerance variable to tune, the simplification in this approach has, arguably, an even more unintuitive parameter: an abstract threshold for partitioning.

2.2.2 Critical Review

From Table 2.1, and the subsequent explanation of approaches, I identify three overarching topics that constitute my review of trajectory simplification approaches. The first topic is extending poly-line simplification approaches to handle spatio-temporal trajectories. In the survey by Shi and Cheung (2006), it is apparent that there are many existing algorithms for effective poly-line simplification; however, overwhelmingly, the existing trajectory simplification approaches have focussed on one in particular: the Douglas-Peucker algorithm (Douglas and Peucker, 1973). Most existing trajectory simplification approaches that compare against, or extend, the Douglas-Peucker algorithm justify their choice by stating that the Douglas-Peucker method is “popular" and “well-known" (Meratnia and By, 2004; Cao et al., 2006; Gudmundsson et al., 2009; Muckell et al., 2011; Long et al., 2013; Lin et al., 2016). While this is undoubtedly true, I argue that this does not necessarily make it the only poly-line simplification approach worth extending or comparing against. Furthermore, while the

Douglas-Peucker algorithm may be effective in its own domain, the plethora of trajectory simplification works that use it as a comparison certainly demonstrate the error and ineffectiveness of the original algorithm for trajectory simplification. Thus, it seems contradictory that it is one of the only poly-line simplification approaches being extended to the problem of trajectory simplification.

In general, simplification approaches always tend toward optimising some criteria. What the criteria is, should be selected based on the use-case and characteristics of the data. The Douglas-Peucker approach scores the significance of each entry based on perpendicular distance, which, as the literature has shown, preserves the shape of the poly-line well. However, shape is just one metric; in trajectory simplification one may also wish to preserve local speed, density, or even stops when simplifying. Thus, it appears to me that the field has wrongly ignored many other existing poly-line simplification concepts simply because the Douglas-Peucker approach is the most well-known to them. I argue that many existing poly-line simplification concepts have unique significance scoring functions that are worth investigating as they may yield novel trajectory simplification approaches useful for preserving some specific criteria.

My second review topic is the constraint various approaches use to produce their simplifications. There appears to me to be two distinct methods for trajectory simplification, each a result of a specific use-case; these methods are error-bounded and size-bounded. Error-bounded approaches only retain entries that do not introduce more than a user-specified amount of error into the final simplification. In this context, the term “error" is approach specific; for example, in some algorithms, this is the amount of heading change, and in others, it is a spatio-temporal distance measure such as SED. Error-bounded approaches are particularly useful in online use-cases where the size of the whole trajectory is not available or known ahead of time (i.e. see Meratnia and By (2004), Lange et al. (2008), Muckell et al. (2014), Liu et al. (2013), and Liu et al. (2015)). The trade-off error-bounded approaches make, is that they require the user to set a domain-specific and algorithm-specific error tolerance parameter. While an error tolerance parameter does give the user tuning control over the algorithm, it is also sensitive, and enforces the implicit assumption that the user has an understanding of the underlying significance scoring function of the algorithm; it also assumes that the user can successfully select a reasonable parameter without analysing the data (i.e. the data may not be available).

Conversely, size-bounded approaches (see Long et al. (2013)) do not require any domain or algorithm specific parameters. Simply, they reduce the simplified trajectory to a certain size, or percentage, of the original trajectory, by discarding the most insignificant entries first. The trade-off is that error is unbounded, and that such approaches are largely unsuitable for online use-cases. However, in an offline setting size-bounded approaches provide a straightforward generalisation of the problem. Specifically, size-bounded approaches allow a user to simply specify the amount of reduction desired (i.e an 80% reduction) regardless of the algorithm chosen: the user is afforded a concrete parameter to tune, while also relieving them of the need to deeply understand each algorithm’s inner workings. This brings me to the final and key topic in this part of the review: generalising the trajectory simplification problem.

From Table 2.1, and the subsequent explanation of approaches it seems apparent to me that all the reviewed trajectory simplification approaches can, effectively, be described as a combination of some significance scoring function, executed using some processing strategy. In fact, in nearly all the reviewed

trajectory simplification approaches, there is a clear separation between the significance scoring functions and processing strategies used. Thus, there is an opportunity to generalise the problem of trajectory simplification as simply combining various scoring functions and processing strategies. Specifically, a user could easily iterate on various combinations of significance scoring functions and processing strategies to create new and customised trajectory simplification algorithms that are more suitable for their requirements. The closest existing work, that attempts something like this, is that of Meratnia and By (2004). In their work, they combine two scoring functions, one using SED and the other using speed, with two processing strategies they call opening-window and top-down. These combinations are performed manually and the authors provide no generalisation in regard to creating further approaches in this manner.

The most straightforward generalisation to begin with, seems to be a framework that creates offline approaches using a size-bounded design. This is because, as I mentioned above, a size-bounded approach eliminates the requirement for various algorithm-specific parameters for each approach, and replaces them with a single user-specified simplification percentage parameter. As per my first review topic, such a framework could also trivially incorporate many significance scoring functions from existing poly-line simplification approaches. Lastly, I highlight that any such generalisation of the trajectory simplification problem has not been explored in existing works despite surely being a boon to many trajectory data mining pipelines.

2.2.3 Literature Gaps

Each of the review topics I covered in Section 2.2.2 has revealed a gap in the trajectory simplification literature. Firstly, existing trajectory simplification has focused mainly on extending the Douglas-Peucker poly-line simplification algorithm, when many other potentially effective poly-line simplification approaches exist that can also be extended to the problem of trajectory simplification. Secondly, the size-bounded criteria for trajectory simplification seems to provide a generalisation of the problem under a single parameter interface, yet, no existing works have exploited this. Thirdly, there is an opportunity to generalise trajectory simplification approaches as a combination of scoring function and processing strategy; however, no existing approach has done so. Therefore, I propose investigating a generic, trajectory simplification framework, that can combine various significance scoring functions (some from existing poly-line approaches) and processing strategies, together, to produce size-bounded, offline, trajectory simplification algorithms. I conduct an investigation into this framework in Chapter 3.

2.3 Depth Analysis of Mining Sequential Patterns From Vehicle Trajectories

Unlike the other sections of this literature review, this topic finds itself at the intersection of two fields: sequential pattern mining and vehicle trajectory data mining. Both fields are well established, with much research activity having taken place; however, at the *intersection* of the two (i.e. approaches that mine sequential patterns from vehicle trajectories), research is limited. Thus, I do not present a tabular overview of the field, like I do for the other review topics but, rather an

textual discussion of key developments that have led to the current few approaches that are closest to the problem of sequential pattern mining of map-matched vehicle trajectories.

2.3.1 Overview

Since the late 1990s there has been a growing number of increasingly efficient sequential pattern mining algorithms that find frequently occurring subsequences from a set of input sequences (Srikant and Agrawal, 1996; Zaki, 2001; Ayres et al., 2002; Pei et al., 2004; Yang et al., 2007; Fournier-Viger et al., 2014a). Trajectories are essentially sequences, and it is desirable to know which sub-trajectories are frequently occurring: it is useful for identifying traffic congestion, finding popular routes, making travel predictions and recommendations, and generally understanding the movements trends of the tracked entities. Thus, as trajectories are sequential in nature, it intuitively follows one could mine sequential patterns from trajectories. However, all sequential pattern mining approaches require sequences of discrete items as input (i.e. sequences of strings or integers), which is a mismatch with trajectories that are typically a series of geographic coordinates. Despite this conceptual incompatibility between trajectories and sequential pattern mining, a number of approaches have been presented over the years that mine sequential patterns from trajectories (Cao et al., 2005; Yang and Hu, 2006; Giannotti et al., 2007; Morzy, 2007; Lee et al., 2009; Gidófalvi and Pedersen, 2009; Savage et al., 2010; Shaw and Gopalan, 2014). Broadly speaking, the technique that such works employ, is to transform the trajectory into a series of discrete items, and then perform sequential pattern mining on those discrete sequences. To convey how a trajectory dataset can be discretised, I present the general process that two existing-approaches use to mine sequential patterns from trajectory datasets.

Firstly, Cao et al. (2005) transforms raw geographic trajectories into discrete items by using line simplification to find key segments within their dataset. Each segment is given a unique id, then, every trajectory entry within a user-specified distance of these segments is associated with the corresponding segment id. Once all the trajectory entries are associated with segments, sequential pattern mining is performed on the sequences of visited segments. Secondly, Giannotti et al. (2007) partition the bounding box of the trajectory dataset into a spatial grid of uniform cells and then count how many trajectories pass through these grid cells. These grid cells are then iteratively expanded to include their neighbours, as long as a minimum overall trajectory count is maintained across the cells. These expanded groups of cells become RoIs (recall Section 2.1.3). Once all RoIs are formulated the original trajectory dataset is transformed into a series of RoI visitations, which makes it discrete and suitable for sequential pattern mining.

In contrast to approaches such as Cao et al. (2005) and Giannotti et al. (2007), that can mine sequential patterns from trajectories moving around in free space, I focus on a more specific version of the problem: mining sequential patterns from vehicle trajectories. With vehicle trajectories, one can assume they are constrained to an underlying road network and, therefore, they should be treated differently than other types of trajectories that are free to move around in space. This assumption of constrained movement means that even under heavy spatial uncertainty one can use map-matching techniques (recall Section 2.1.2) to estimate, with some confidence, the true route that vehicle travelled. Additionally, map-matching vehicle trajectories to an underlying road network effectively discretises trajectory coordinates into a series of visited road nodes; thus, making

the trajectories suitable for sequential pattern mining. This exact process is described by Zheng (2015, p.21) who says, "[w]hen the sequential pattern mining problem is applied to a road network setting, we can first map each trajectory onto a road network by using map-matching algorithms. A trajectory is then represented by a sequence of road segment IDs".

However, I argue that even after vehicle trajectories are discretised using map-matching, there still exists some unsuitability with general sequential pattern mining algorithms. Since vehicle trajectories are implicitly contiguous, driving from one road to the next without suddenly skipping around, one should try to mine sequential patterns that reflect the exact routes they travel. General sequential pattern mining algorithms such as GSP (Srikant and Agrawal, 1996), SPADE (Zaki, 2001), SPAM (Ayres et al., 2002), PrefixSpan (Pei et al., 2004), LAPIN (Yang et al., 2007), and CMSPAM (Fournier-Viger et al., 2014a) offer no such guarantees about the contiguous nature of the mined patterns. For example, such approaches may find a pattern such as, $regionA \rightarrow regionB$. This kind of high-level pattern tells one nothing of the actual roads the vehicles travelled along; such information would be extremely valuable in domains such as urban planning, route planning, and traffic management (Yuan et al., 2010; Herrera et al., 2010).

Fortunately, this constraint of mining patterns that consist of contiguous items in the underlying sequence database is fairly common, particularly in domains like biology (Exarchos et al., 2008) and geography (Atev et al., 2010) where patterns, consisting of nearby items, are more meaningful. In fact, there is a whole class of algorithms that do, so-called, *gap-constrained* sequential pattern mining, that can be used to mine contiguous sequential patterns (Antunes and Oliveira, 2003b; Zhu and Wu, 2007; Li et al., 2012; Zhang et al., 2015). Specifically, these gap-constrained approached mine sequential patterns such that the underlying items that constitute a given pattern must be within at least some user-specified *max-gap* of one another in the input sequences. Thus, if one wanted to mine contiguous sequential patterns from vehicle trajectories, assuming that vehicles always move from one road node to the next, the max-gap parameter of gap-constrained algorithm would simply be set to one. However, max-gap constrained or otherwise, there is a problem with sequential pattern mining approaches that generally makes them unsuitable for mining long and homogeneous input sequences, such as map-matched vehicle trajectories.

Vreeken et al. (2011) calls the problem: "pattern explosion". As Vreeken et al. (2011) explains, when the input sequences are sufficiently homogeneous, a huge number of patterns are produced as a result of pattern mining: pattern outputs become so large that they are orders of magnitude larger than the size of the input. Datasets such as vehicle trajectories which are long and mostly similar (i.e. driving many similar roads between sequences) are ideal candidates to exacerbate such pattern explosions. The simplistic solution to mitigate pattern explosion, is to increase the minimum support parameter used to formulate patterns; however, the trade-off is that only a small number of well-known patterns are found. The problem of pattern explosion is that each pattern is stored in the pattern mining output regardless of the already discovered patterns. Thus, as Vreeken et al. (2011) highlights, "we end up with a rather redundant set of patterns, in which many patterns essentially describe the same part of the database". A highly redundant pattern output is difficult for a user to interpret; thus, tasks like visualisation offer little value. I highlight that another technique to reduce the number of patterns is to apply additional constraints to the pattern mining process, such as finding only the set of all closed or maximal patterns (recall Section 2.1.3). While closed and

maximal constraints do reduce the number of patterns discovered they do not necessarily tend towards a less redundant set of patterns; thus, as Vreeken et al. (2011) notes “redundancy remains an issue".

Vreeken et al. (2011) propose an algorithm called “KRIMP", in the related problem of frequent itemset mining. KRIMP reduces the output redundancy by finding the set of patterns that best compresses the input dataset, with respect to the Minimum Descriptive Length (MDL) principle. The concept of KRIMP is then extended by Lam et al. (2014) who propose “GOKRIMP" for the top-k sequential patterns that best compress the input dataset. Unfortunately, GOKRIMP is unsuitable for use on vehicle trajectories because it does not support mining contiguous sequential patterns. Overall, it seems that existing general, gap-constrained (contiguous), concise (closed and maximal), and compressing sequential pattern mining algorithms are all unsuitable for handling map-matched sequences of vehicle trajectories for one reason or another. Despite this, some approaches in the literature have used gap-constrained approaches for tasks related to vehicle trajectories.

For example, Song et al. (2014) and Wang et al. (2014) both use map-matching to discretise vehicle trajectories in order to mine sequential patterns from them. Specifically, Song et al. (2014) mines contiguous sequential patterns from vehicle trajectories using a Suffix Tree, transforms them into Huffman codes, and then presents a compression format for vehicle trajectories based on the codes. Wang et al. (2014) borrows this same Suffix-Tree-based approach, and mines contiguous sequential patterns, to use as travel time estimates between road nodes. However, both approaches mention they limit the depth of their tree to twenty road nodes, which, I assume is a setting to limit the impacts of pattern explosion and increase mining efficiency. Additionally, limiting the maximum pattern length does nothing towards reducing the redundancy of the pattern output. Thus, an approach that mines contiguous sequential patterns from vehicle trajectories without producing a huge and redundant output seems non-existent.

2.3.2 Critical Review

While an approach for my purpose does not appear to exist, map-matching, vehicle sequential pattern mining, and the combination of the two, have all been explored in the literature before. Thus, for my review, in this section I pose and speculate on two questions, in regards to why vehicle trajectories have not been mined in this way before.

Are succinct, or less redundant, contiguous sequential patterns of vehicle trajectories that useful? Such a set of patterns would ideally describe the popular routes of the tracked vehicles in a way where there is little or no overlap between patterns. Thus, each pattern would represent a mostly-unique route, that a number of different vehicles from the input dataset had driven. In this way, the pattern output may serve as a sort-of overview of the general trends of the tracked vehicles. Intuitively, such information seems useful for domains such as urban planning, traffic management, and personalised navigation. Specifically, less redundant contiguous sequential patterns of vehicles would be far more useful than the full set, simply because they would be easier to use and interpret (for both humans and machines). For example, a lack of redundancy opens up the potential to visualise the popular routes, which, if done before using the full set of contiguous sequential patterns, would be a visual mess.

In what ways can the problem of redundancy in the pattern output be solved? What compromises are there? There appears to me to be at least two viable approaches for reducing the redundancy in the pattern output of the contiguous sequential pattern mining process. The first, is to extend algorithms such as KRIMP (Vreeken et al., 2011) and GOKRIMP (Lam et al., 2014), and invent a contiguous sequential pattern mining algorithm that finds the top-k patterns and compress the input sequence with regard to some criteria. However, if the goal is to mine a pattern output that presents an overall view of the popular routes driven by vehicles, a top-k approach seems somewhat problematic for the user. Specifically, the user cannot know how many routes is reasonable to obtain an overview of the dataset.

The second approach, is to mine the set of contiguous sequential patterns in a lossy way: iteratively keeping the most representative patterns and discarding all patterns that exceed some redundancy criteria. In this way, the user can control the level of redundancy they wish to allow in the pattern output. However, the trade-off is, that some patterns will be discarded in favour of others, and ultimately those patterns may not be fully represented in the output. In the context of vehicle trajectories, this will likely lead to the popular, long, routes receiving more representation that smaller routes simply because the longer routes represent a greater portion of the input sequences. This option seems to err towards providing an overview of the vehicle movements, as opposed to a highly detailed list of all route variations that a contiguous or gap-constrained sequential pattern mining approach would produce.

Overall, I argue contiguous sequential patterns of vehicle trajectories are useful, but the problem of pattern explosion and output redundancy have hampered the combination of the map-matched vehicle trajectories and contiguous sequential pattern mining approaches, so far.

2.3.3 Literature Gaps

I identify only a single, but important, gap in the literature in this section. Specifically, there is no existing approach to mine contiguous sequential patterns, from map-matched vehicle trajectories in a way that does not produce a huge or highly redundant pattern output. Thus, I investigate such an algorithm in Chapter 4 where I introduce a new type of sequential pattern called a “distinct pattern".

2.4 Depth Analysis of Stop/Move Detection Approaches

Late in the first generation of trajectory data mining, Spaccapietra et al. (2008) introduced a model for representing a moving entity’s journey as a series of stop episodes with move episodes in-between. This stop/move model semantically enriches the trajectory, and turns out to be a useful representation for a number of problems. Specifically, since it was introduced, the stop/move model has been used to semantically enrich trajectories for: semantic trajectory data mining (Cao et al., 2010; Khetarpaul et al., 2011; Zheng et al., 2009), location recommendation (Leung et al., 2011; Takeuchi and Sugimoto, 2006; Ying et al., 2014), and activity recognition (Boukhechba et al., 2015; Huang et al., 2010; Spinsanti et al., 2010; Xie et al., 2009). However, to facilitate any of the use-cases I just listed, there is, of course, the actual task of, first, finding the trajectory stops and moves. Typically, trajectories only contain spatio-temporal information so it is a classic unsupervised classification

problem to determine the correct stop or move label for each trajectory entry. Over the last decade there have been many stop/move trajectory detection approaches, even more trajectory applications that use stops/moves in some way, and a number of approaches for all of the steps in-between; however, I wish to clarify that in this review I limit the scope solely to the problem of stop/move detection.

2.4.1 Overview

I present an overview of the relevant stop/move approaches I review in Table 2.2, followed by a brief explanation of the unique contributions of each approach.

1. Alvares et al. (2007) present SMoT, which is one of the foundational works for detecting stops and moves, and also the first approach I know of to introduce the concept of geographic stop/move detection. Additionally, one could also argue that SMoT was the approach that popularised the application of the stop/move model as presented by Spaccapietra et al. (2008). Lastly, I highlight that SMoT is one of the few stop/move approaches that, at the time of writing, has publicly available source code [2].

2. Palma et al. (2008) present a density-based clustering approach called "CB-SMoT". CB-SMoT is a direct extension of SMoT (Alvares et al., 2007) and is heavily based on DBSCAN (Ester et al., 1996), with some modifications to handle temporal and sequential data. I highlight that CB-SMoT is one of the earliest stop/move approaches to extend DBSCAN and also the first approach that can find stops with or without a collection of known places. Similarly to SMoT, CB-SMoT also has publicly available source code at the time of writing [3].

3. Zimmermann et al. (2009) present "StopFinder", which is an extension of OPTICS (Ankerst et al., 1999). I highlight that StopFinder is one of few approaches to extend OPTICS, where, most clustering-based, stop/move approaches extend DBSCAN. Unfortunately, the authors do not provide comparisons against any DBSCAN-based stop/move approaches, so it is difficult to evaluate whether their choice of algorithm improves the accuracy of stop/move detection or not. Another, aspect that I highlight is that, unlike the majority of other works, Zimmermann et al. (2009) provide estimation techniques for some of their algorithm's parameters. Specifically, the spatial and temporal parameters are estimated by calculating the average entry displacement and recording gap. There is no quantitative experiments demonstrating parameter sensitivity or parameter estimation effectiveness for StopFinder; thus, I am left to speculate that, while better than nothing, these parameter estimations are most likely too simple to be useful because they capture moving entries too.

4. Zheng et al. (2009) present a simple stop detection algorithm. This algorithm formulates stops by finding sub-trajectories where the distance between entries does not surpass a user-specified spatial threshold and the whole sub-trajectory has a duration that is at least as long as a user-specified temporal threshold. This requires the user set an optimal spatial parameter, that is not so large that stops grow too much, and not so small that actual

[2] `https://github.com/yipeng/WEKA-STPM`
[3] See footnote 2

Approach	Type	Parameters	Estimate All Parameters?	Ground-truth available?	Source Code?
1. Alvares et al. (2007)	Geographic.	G, T.	✗	✗	✓
2. Palma et al. (2008)	Clustering-based.	S, T, G.	✗	✗	✓
3. Zimmermann et al. (2009)	Clustering-based.	S, T, N.	✗	✗	✗
4. Zheng et al. (2009)	Clustering-based.	S, T.	✗	✗	✗
5. Rocha et al. (2010)	Clustering-based.	D, T, N.	✗	✗	✗
6. Tran et al. (2011)	Clustering-based.	S, T	✗	✗	✗
7. Thierry et al. (2013)	Clustering-based.	S.	✗	✗	✓
8. Gong et al. (2015)	Clustering-based.	S, P, D, N	✗	✗	✗
9. Fu et al. (2016)	Clustering-based.	S, T.	✗	✗	✗
10. Xiang et al. (2016)	Clustering-based.	S, T, T, T, N, N.	✗	✗	✗
11. Hwang et al. (2017).	Clustering-based.	S, T, P.	✗	✗	✗
12. Luo et al. (2017)	Clustering-based.	S, P, N.	✗	✗	✗

TABLE 2.2: Overview of relevant stop/move literature. Parameter legend: S = spatial threshold, T = temporal threshold, D = directional threshold, G = geographic places, P = minimum number of points, N = non-attribute related threshold.

stops are split into a series of smaller stops. Unfortunately, no parameter estimation techniques, effectiveness experiments, parameter sensitivity analysis, or general comparison to other stop/move approaches are presented in this work. This is somewhat expected though, as the stop detection algorithm in this approach is part of a much larger semantic trajectory data mining application for mining interesting locations and travel sequences from human trajectory datasets (i.e. the main focus of the work is not stop detection).

5. Rocha et al. (2010) present an approach called “DB-SMoT", which is a direct extension of CB-SMoT (Palma et al., 2008). I highlight that DB-SMoT is one of the first and only approaches to incorporate the directionality of trajectory entries when calculating if the tracked entity is stopping. Rocha et al. (2010) evaluate the effectiveness of DB-SMoT and CB-SMoT in the domain of fishing vessel trajectories. Their experiment results indicate that DB-SMoT is more effective in this domain because their approach is built to handle the case where stops drift around; however, no claim is made, nor experiments performed, to measure whether DB-SMoT is suitable, or more effective, in other cases, such as human trajectories.

6. Tran et al. (2011) present “TrajDBSCAN", which is another extension of DBSCAN (Ester et al., 1996). I highlight that Tran et al. (2011) are one of the few authors who use publicly available datasets: this makes their experiments fully reproducible. However, the datasets used are of raw trajectories with no indication of the true stops or moves that the entity made. Due to this, the authors are forced to evaluate their approach by visualising stops on map and discussing their validity. The authors also perform some quantitative tests, measuring the number of stops as they vary algorithm parameters and input trajectory sampling-rates. I argue that simply counting the number of stops is a poor metric, because there is no indication whether these are true or false stops. Another aspect I highlight is that Tran et al. (2011) introduce several concepts beyond pure stop/move discovery. Specifically, they propose techniques to form a hierarchy of stop types, such as: concrete stops (at specific places); generic stops (at large regions like shopping centres); personalised stops (where only a single entity stops); and, shared stops (where many entities stop). These additional stop types are formulated by splitting and merging stop clusters, and they are likely useful for semantic pattern mining; however, the authors do not investigate this research direction.

7. Thierry et al. (2013) present a density-based approach that uses kernel density surfaces to formulate stop clusters. I highlight that this approach only requires a single spatial parameter, which, while simpler to tune, is used in a way that does not consider the sequential nature of the trajectory data being processed. Experiments are performed using a synthetic ground-truth; however, there is no comparison to other approaches. Additionally, at the time of writing, a publicly available implementation of their algorithm is available for ArcGIS users [4].

8. Gong et al. (2015) present “C-DBSCAN", which, as the name implies, is another extension of DBSCAN (Ester et al., 1996). I highlight a unique feature

[4] http://www.spherelab.org/tools

of C-DBSCAN: it calculates the distribution of per-entry direction changes to further refine whether a cluster is a stop or move. Specifically, the authors introduce an assumption that a small distribution of direction changes is likely a move and a larger, more erratic, distribution of direction changes is likely a stop, caused by GPS spatial uncertainty. This extra step introduces two new parameters that are required by their approach. To ease the task of parameter selection, Gong et al. (2015) introduce a cumulative frequency technique to estimate all of C-DBSCAN's parameters. However, their parameter estimation approach is largely impractical because it requires samples from a labelled ground-truth dataset. Lastly, the authors introduce a post-processing step to improve classification accuracy. Specifically, the authors use labelled ground-truth trajectories to train a support vector machine (Cortes and Vapnik, 1995) to further refine stop/move classifications. The results indicate that the SVM is an effective technique in a supervised setting; however, I highlight that it is entirely unhelpful when there is no labelled ground-truth.

9. Fu et al. (2016) present "TDBC", which is a two-step approach to find stop clusters and their centres. In the first step the authors present a simple threshold-based clustering algorithm that handles three specific stop cases: stops indicated by a recording gap, a single uninterrupted stop at some place, and a stop where a person begins or ends their tracking. Once these stop clusters have been formulated they use a modified point clustering approach based on "CFSFDP" (Rodriguez and Laio, 2014) to find the centres of each cluster using local density. This is one of the only approaches that attempts to find the actual, physical location of the entity stops: most approaches present the stop as a polygon formulated by the convex hull of the stop cluster, or by the average coordinate of the stop cluster with a radius. Fu et al. (2016) quantitatively compare the efficiency and effectiveness of their approach to some others using several ground-truth datasets. The results indicate that compared to other approaches TDBC demonstrates some improvement in accuracy; the major improvement, however, is efficiency, with TDBC achieving running times nearly an order of magnitude faster than the other clustering-based approaches (though the algorithm implementations are not provided).

10. Xiang et al. (2016) present "SOC", which is based on both DBSCAN (Ester et al., 1996) and OPTICS (Ankerst et al., 1999). Similarly to Fu et al. (2016), SOC is designed to handle stops indicated by a large temporal gap. Furthermore, similarly to Tran et al. (2011), SOC can merge nearby stop clusters that may have been separated due to spatial uncertainty. Additionally, similarly to Gong et al. (2015), SOC uses an additional criteria to filter out likely false-positive stop clusters. In SOC, the cluster straightness, duration, direction change, and centre-distance are calculated, and if some combination of these metrics exceed their respective user-specified thresholds, the cluster is considered a move and not a stop. While this assumption seems reasonable, it introduces the same problem as in C-DBSCAN (Gong et al., 2015), because the user now has to estimate a number of additional parameters when using SOC. Unlike, Gong et al. (2015) though, Xiang et al. (2016) provide no techniques to set the parameters in the supervised or unsupervised case. Instead, they simply provide some default values that work for their test datasets. Experiment results indicate that SOC

outperforms some existing density-based stop/move detection algorithms; however, ultimately the number of hand-tuned parameters cannot be overlooked when using this approach in an unsupervised setting.

11. Hwang et al. (2017) present some pre-processing and post-processing techniques that are used in combination with an unmodified version of DBSCAN (Ester et al., 1996). The first pre-processing technique is to remove some outliers based on unreasonably high-speed entries. The second pre-processing step is to handle recording gaps in the trajectory data, creating new trajectory recordings at a constant time-step, by linearly interpolating between the recording gaps in the trajectory. Lastly, the post-processing technique they introduce is to run an odd-numbered, fixed-size sliding window over the final labelled entries, and homogenize all entry labels within the window based on the majority label. While the actual stop/move approach presented is simply DBSCAN, the pre-processing and post-processing techniques presented are likely applicable to nearly all existing stop/move approaches.

12. Luo et al. (2017) present “DF-DBSCAN", which is a combination of both DBSCAN (Ester et al., 1996) and kernel density concepts. Luo et al. (2017) introduce a novel concept called “move ability", which, when given a sub-trajectory, is calculated as the distance between the start and end point, divided by the distance along the actual sub-trajectory trail. The assumption is, that a stopping sub-trajectory may have quite a lot of movement, due to spatial uncertainty, but doesn’t actually displace itself very far overall; a move, however, is expected to have a distance between the start and end point that is fairly similar to the distance of the path travelled. Move ability is calculated at each trajectory entry by sampling a user-specified number of entries on either side of the current entry. Each entry’s move ability is then passed into a kernel density function that has weightings for the move ability and the impact of surrounding entries. The result is a density score calculated at each entry, which is then, ultimately, used in a slightly modified sequential DBSCAN to formulate stop clusters. Unlike other authors, Luo et al. (2017) provide a data-driven technique to estimate a reasonable value for the minimum density parameter used in DF-DBSCAN. Unfortunately, the user is still left to manually set three other parameters, two of which are abstract weighting parameters. Fortunately, Luo et al. (2017) do provide a set of experiments demonstrating some of the effects of varying these parameters for their datasets. Some of their observations, likely, hold for their algorithm in the general case; however, I argue that others are specific to their data (such as the setting for the number of points parameter). Additionally, Luo et al. (2017) perform two quantitative experiments using real-world, ground-truth trajectories, and both experiments indicate DF-DBSCAN achieves a higher accuracy than the compared approaches.

2.4.2 Critical Review

Table 2.2 indicates that two main stop/move paradigms are used throughout all the approaches I considered: geometric containment and clustering. Geometric containment, simply, intersects some known place with the trajectory, and all the contiguous sub-trajectories that stay within the place, for some user-specified duration, are labelled as stops. Conversely, clustering-based approaches build up

groups of nearby (and typically sequential) entries, and if the group conforms to some user-specified density (number of entries) or time (stop duration) constraint, then it becomes a cluster and is labelled as a stop. Geographic containment and clustering-based stop move detection were both presented relatively-early-on by Alvares et al. (2007) and Palma et al. (2008) respectively, and I argue that subsequent approaches have only made fairly minor changes to the overall process since then.

To illustrate this argument, I refer the reader back to Table 2.2, and highlight that since Palma et al. (2008) introduced "CB-SMoT", their clustering-based stop/move detection algorithm based on DBSCAN (Ester et al., 1996), all subsequent approaches I reviewed also used a clustering-based approach. One might say my clustering-based grouping is too general and does not indicate the similarity of these approaches. However, I argue that many of them are indeed similar; specifically, I have found that 66% of the reviewed approaches introduced after CB-SMoT follow the same general algorithm design and directly extended DBSCAN to find stop clusters. Hwang et al. (2017) even uses the original DBSCAN algorithm, which does not consider the sequential nature of trajectories, and adds an additional minimum stop time constraint. Of course, the approaches I reviewed are not entirely the same as each other: some consider extra dimensions in the trajectory data, such as DB-SMoT considering direction (Rocha et al., 2010), while others extend the original stop/move detection task in problem-specific ways (i.e. Tran et al. (2011) introduced a hierarchy of stop types, like shared stops and personalised stops). However, there certainly appears to be a reasonable amount of similarity in terms of pure stop/move detection processes. Thus, if the user's goal is to find *only* stops and moves in the trajectory, the job of selecting the single most suitable approach from the existing works is somewhat unclear. For example, if I consider one of the use-cases in this thesis —to incorporate some stop/move approach as part of a semantic trajectory data mining application —then it is not immediately clear from algorithm features alone which approach from the literature would be the most suitable. Therefore, I formulated the following questions to evaluate the suitability of existing approaches for my purpose:

1. *What is the stop/move classification accuracy of the approach?*
2. *Will the approach find stops/moves from a raw trajectory, purely based on the data; or, does it require parameter setting? If user-specified parameters are required, how is the user to set them?*
3. *Given that a trajectory has a degree of spatial uncertainty, is there any degree of confidence attached to the stop/move labels produced (i.e. I would prefer not to have false stops causing false patterns)?*

In regard to the first question, the accuracy of existing approaches, I observed that the task of labelling trajectory entries as either stop or moves appears to be a typical unsupervised classification problem. Thus, I would expect existing approaches to quantitatively test their accuracy like other binary classifiers: that is, using ground-truth datasets. However, some of the approaches I reviewed did not: they simply discussed measurements like the number or location of stops (Alvares et al., 2007; Palma et al., 2008; Tran et al., 2011). Such measurements do little to indicate the quantitative accuracy of a stop/move approach. Those aside, however, the other approaches I reviewed did use ground-truth datasets to measure the effectiveness of their algorithms; I highlight, however, that the techniques they

used to make their ground-truth datasets were of varying quality. For example, some approaches took existing trajectories and labelled them, after recording, by using mapping applications, and carefully examining the distance and temporal gap between recordings (Xiang et al., 2016; Hwang et al., 2017; Luo et al., 2017); however, this is not ideal, as this introduces some bias and potential for confirmatory results. Other researchers generated synthetic ground-truth datasets by simulating stopping and moving entities (Thierry et al., 2013); however, these synthetic datasets surely lack the true nuances and errors of real GPS trajectories. Lastly, some existing works used GPS trajectories of people in the real-world, and those people provided a ground-truth by manually annotating whether they were stopping or moving in the moment, using an application on the GPS receiver or some other means (Zimmermann et al., 2009; Rocha et al., 2010; Gong et al., 2015; Fu et al., 2016).

The problem is, that even after a decade of research, not a single author has shared their ground-truth stop/move trajectories. I am reminded of a similar problem early in the field of map-matching that Newson and Krumm (2009) observed: "all the work on map-matching used private data sets for testing, making it impossible to objectively compare results from different algorithms". This is exactly the case here, too; due to the lack of a public ground-truth dataset, there is no way to objectively reproduce and compare the quantitative accuracy demonstrated in the experiments of existing approaches. This problem of reproducibility and comparability between approaches is further exacerbated by the distinct lack of public source code for the majority stop/move algorithms in the field (see Table 2.2). The result is that the user who wishes to implement some existing stop/move approach has to make a judgement as to which set of incomparable experiments might best represent the algorithm's suitability for their use-case. Thus, answering my first question, regarding the accuracy of existing stop/move approaches, appears to require creating a ground-truth dataset, implementing each approach, and then comparing them. This is surely out of scope for most users and certainly out of scope for this thesis.

This brings me to the second question: the requirement for parameters and how to select them. Table 2.2 reveals that all the reviewed approaches require at least one parameter, other than the input trajectory, with the majority requiring three. These parameters are used to define a stop considering some combination of features (i.e. space, time, speed, direction etc.). However, because spatial uncertainty varies as a GPS receiver moves around, selecting the optimal parameters to correctly label noisy stops across the whole dataset is a non-trivial task. In fact, without a ground-truth dataset to verify against, the task can devolve into guesswork with no indication of correctness. From my review of the literature, there does not appear to be a clear indication of which approach has the least sensitive parameters, and *very* little indication of how to select parameters for any of the approaches in an unsupervised setting. Some approaches introduce estimation functions for some, but not all, of their algorithm parameters (Zimmermann et al., 2009); however, even one user-specified parameter still requires the user to, effectively, guess when tuning the algorithm. Specifically, after reviewing the literature, I have concluded that using any of the existing approaches and tuning their parameters on a raw trajectory, by guesswork, would be fairly useless; the only indicator would be finding, more or less stops —which is hardly an effective metric to judge accuracy. As I have highlighted in Table 2.2, there is an overall lack of parameter estimation in the existing works, which is unreasonable, as the data is noisy and the parameters numerous. Thus, in answering to my

second question: all existing approaches require parameters and a user to manually (and blindly) set at least one, if not all, of those parameters.

Now, onto my third question, regarding the confidence of any given stop/move classification. All the stop/move approaches I reviewed only classify trajectory entries definitively. That is, an entry is either labelled as a stop, a move, or some other type the author was interested in. By definitively labelling entries (or clusters of entries), existing approaches do not provide any indication that some classifications may be more or less ambiguous than others, due to some factors such as spatial uncertainty. In my case, where I wish to use a stop/move approach as a step in a semantic data mining application, this is not ideal. Ideally, I would prefer each stop/move classification to have an associated confidence value; I could use this value to filter out potentially false-positive stop classifications, thereby, filtering out false patterns that may occur later in the data mining process. Thus, in answer to my third question, there is no indication of classification confidence in any existing stop/move approaches.

2.4.3 Literature Gaps

Summarising the review in Section 2.4.2 I can answer the three questions raised as follows:

1. *What is the stop/move classification accuracy of the approach?* Likely, due to a lack of publicly available implementations and a labelled ground-truth stop/move trajectory dataset, there is no objective way to compare the accuracy of the reviewed approaches without implementing each one and collecting a ground-truth dataset ourselves. In short, very few of the reviewed stop/move approaches share their source code, nor is there any ground-truth shared for evaluating approaches.

2. *Will the approach find stops/moves from a raw trajectory, purely based on the data; or, does it require parameter setting? If user-specified parameters are required, how is the user to set them?* There was no reviewed approach that can estimate all its own parameters effectively; of the reviewed approaches that provide guidance on parameter setting, that guidance is limited in an unsupervised setting.

3. *Given that a trajectory has a degree of spatial uncertainty, is there any degree of confidence attached to the stop/move labels produced (i.e. I would prefer not to have false stops causing false patterns)?* All reviewed approaches do not consider the possibility that some stop/move classifications may be more uncertain than others.

Overall, the reviewed stop/move approaches seem, in some way or another, unideal in regard to the posed questions. Thus, I propose to investigate a probabilistic stop/move approach that detects stops and moves in trajectories, and associates a value with each entry that indicates the probability that the entry was stopping. In this way, I will be able to fulfil my use-case, of filtering out ambiguous stops before I perform further data mining. Additionally, I will make the implementation and ground-truth datasets used in my quantitative experiments available to the public, as it is evident from my review that the field as whole is sorely lacking such a resource. My hope is that future researchers will be able to use the dataset to quantitatively and objectively compare the effectiveness of my approach to theirs, with no additional burden to themselves. This investigation into a probabilistic stop/move approach is detailed in Chapter 5.

2.5 Depth Analysis of Place-matching Approaches

To review the existing place-matching literature, I have identified five features that the existing place-matching approaches introduce and build upon. Considering each feature, I am able to compare the differences between existing approaches, and also identify gaps in the literature as a whole. I introduce each feature as follows:

1. **Points/Polygon Places.** When matching places to trajectories it is common that these places come from some database of known places, such as OpenStreetMap or Google Places. OpenStreetMap stores place geometries as polygons, or points, depending on the type of place; whereas, Google Places and Foursquare stores places strictly as a single point. Supporting only one geometry type or the other limits the usability of a place-matching approach; thus, ideally, both should be supported. Furthermore, perhaps even line geometry should be supported, too, for stops that occur on a street or walkway (such line geometries are also available through services like OpenStreetMap).

2. **Stop/Place Disambiguation.** Many place-matching approaches begin by extracting trajectory stops; they, then proceed, matching each stop with a single known place. However, the task of matching a stop to a single place becomes ambiguous when the stop itself is surrounded by, or intersecting, several potential places. This is a relatively common occurrence when real-world place databases are used, especially when the region being studied is dense with places. Thus, the new task becomes one of stop/place disambiguation. Some approaches ignore this problem, which simply lowers their accuracy, whilst others simply match the stop to the closest place. However, matching to the nearest place is, typically, entirely unreliable, at the scale of stops and places, because the stop geometry/location should be considered somewhat spatially uncertain; the distance measurement cannot be trusted absolutely. Other factors, such as place type, the duration of the stop, the opening hours of the candidate place, and the potential sequence of visited places, form a far more well-rounded solution to the stop/place disambiguation problem.

3. **Order of Visited Places.** Considering that the purpose of place-matching is to transform a trajectory into a sequence of visited places, one should not forget that the tracked entity does not visit places entirely randomly but, instead, visits them in some reasonable order. Assuming this, one may enforce rules that prune away seemingly unlikely visitation sequences from consideration. For example, one may invent a weighting, where it is extremely unlikely for a person to go to a restaurant and then go to another restaurant one hour later. Such rules are somewhat arbitrary and require a good understanding of the tracked entities. A far less arbitrary approach from the literature is to construct a network of all possible place visitation sequences caused by a sequence of trajectory stops (Yan et al., 2013; Lv et al., 2016). Each of these potential visitation sequences has a probability of occurring; thus, the task is simply to select the most likely sequence, and that becomes the final place-matching result. However, building such a probabilistic network introduces a new problem: inferring the probabilities of travelling from one place to another. The simple solution is to use ground-truth data to determine

such probabilities; however, the requirement for ground-truth datasets limits the usability of the approach.

4. **Place Topology.** When place geometries are represented as polygons (like in OpenStreetMap) it is not uncommon for some places to be fully contained by, or intersecting, with some other places. Consider the scenario where there is a University place polygon that fully contains a smaller café place polygon, and the geometry of the stop intersects the café and, thereby, is also contained in the University. This once again introduces the problem of stop/place disambiguation, but, this time, in a more nuanced way. The University is certainly not an incorrect match for the stop, but the café may be more specific and therefore more semantically meaningful. Thus, the task is to develop a procedure that somehow selects a single place in a logical and consistent way when presented with these topologically complicated places.

5. **Supervised or Unsupervised.** Place-matching is essentially a machine learning task. Thus, like most machine learning tasks it can be performed in a supervised or unsupervised manner, with the use of labelled data to initialise or train the model. However, an unsupervised approach is highly preferable for place-matching because ground-truth trajectories, labelled with the places the entities visited, are time-consuming and often times not possible to acquire (i.e. for privacy reasons).

2.5.1 Overview

Considering each of the aforementioned features, I present a feature comparison of some relevant existing place-matching approaches from the literature in Table 2.3, which is then followed by a brief explanation of each approach.

1. Liao et al. (2007) present an early work for extracting places from people's raw trajectories. Out of all the works I reviewed in this section, this work is the one that is most different from the others. Specifically, Liao et al.'s work infers activities, then places (whereas later approaches typically do the opposite); it does not find all places, but only those it considers significant. However, the places it finds are not semantically enriched: they are simply clusters. It is far more similar to its foundational predecessors such as Ashbrook and Starner (2003), Kang et al. (2004), and Hightower et al. (2005): all of which found visited locations (coordinates or clusters) as opposed to fully qualified semantic places (i.e. places that have names or place types). However, the main reason I include (Liao et al., 2007) in my review, is that it is one of the first approaches that probabilistically models the problem of matching semantics (activities not places in this case) to a trajectory as a whole. Particularly, it makes the important consideration that the potential order of visited places (or activities performed in this case) makes some visitation sequences more or less likely. Specifically, Liao et al. (2007) present an approach where a trajectory is spatially segmented by clustering it onto nearby street edges (a sort of clustering-based map-matching). Then, using contextual, temporal, velocity-based, and sequential feature functions, they train a conditional random field (CRF) on ground-truth GPS traces that are labelled with visited places and the activities performed at each place. The CRF is a probabilistic model that computes the likelihood of certain sequences of activities based on the aforementioned feature functions. Using their

Approach	Handles point/ polygon places:	Handles stop/ place disambiguation?	Considers order of visited place?	Considers place topology?	Unsupervised?
1. Liao et al. (2007)	✗	✗	✓	✗	✗
2. Alvares et al. (2007)	✗	✗	✗	✗	✓
3. Xie et al. (2009)	✗	✓	✗	✗	✓
4. Spinsanti et al. (2010)	✗	✓	✓	✗	✓
5. Richter et al. (2012)	✓	✗	✗	✗	✓
6. Furletti et al. (2013)	✗	✓	✗	✗	✓
7. Yan et al. (2013)	✓	✓	✓	✗	✓
8. Moreno et al. (2014)	✗	✗	✗	✓	✓
9. Boukhechba et al. (2015)	✓	✗	✗	✗	✓
10. Lv et al. (2016)	✓	✓	✓	✓	✗
11. Gu et al. (2017)	✓	✓	✗	✗	✓

TABLE 2.3: Overview of relevant place-matching literature.

trained model, they demonstrate that it can estimate the most likely sequence of activities given a raw GPS trajectory. Then, using the sequence of activities and segments, their approach finds the most frequent weekly activities and clusters those segments to formulate significant places.

2. Alvares et al. (2007) present “Stops and Moves of Trajectories" (SMoT), which is one of the first approaches to directly apply the model presented by Spaccapietra et al. (2008) where trajectories are represented as a series of stops and moves. Unlike later approaches that first find stop and move episodes and then associate them with places, SMoT constructs geographically enriched trajectory stops at any sub-trajectory that is contained in a candidate geographical region, for at least a user-specified duration. In this way, SMoT finds both stops and places in a single step; however, SMoT’s place-matching procedure does not handle places represented as points, nor spatially overlapping places, or trajectories that stop nearby (but not inside) a place (which may be caused by device inaccuracy). I highlight that SMoT is directly extended in a later work called CB-SMoT (Palma et al., 2008). The main change going from SMoT to CB-SMoT is that the algorithm is divided into two separate steps, which allows it to find stops at places with no place geometry. In CB-SMoT the first step is to discover stops by using a clustering-based procedure similar to DB-SCAN (Ester et al., 1996). Then, in the second step, the same geometry intersection from SMoT is used to match the stops to any candidate places they may be contained by. If a stop is not contained in any known place geometry, the stop is simply labelled as an unknown stop. Thus, because the place matching procedure is the same I omit CB-SMoT from the comparison in Table 2.3.

3. Xie et al. (2009) present a version of the place-matching problem, where only a point representation of places is given (a PoI dataset). Using these points they construct a Voronoi diagram where each Voronoi cell represents a single place. Place-matching is performed by associating subsequences of the trajectory with the relevant Voronoi cells that they passed through. Then, using a user-specified set of rules the duration spent in each Voronoi cell/place is used to determine the type of activity that occurred there. I highlight that because the approach presented in (Xie et al., 2009) uses a Voronoi diagram for place-matching, every entry in the trajectory will be associated with some Voronoi cell/place. One can imagine that some local sparseness in the PoI dataset may result in a sub-trajectory being associated with a PoI that is relatively distant from itself. Such issues are non-existent in later place-matching approaches that do not partition the study region.

4. Spinsanti et al. (2010) present the first approach after CB-SMoT (Palma et al., 2008) that place-matches by finding a series of trajectory stops, and matches each stop to a single geographic place. However, unlike SMoT (Alvares et al., 2007) and CB-SMoT (Palma et al., 2008) their approach handled the problem of a stop being nearby to many geographic places. Of note, is that Spinsanti et al. (2010), seemingly independent of the work of Liao et al. (2007), introduce a technique to handle this stop/place disambiguation problem using a probabilistic model based on spatial and temporal proximity. Additionally, Spinsanti et al. (2010) present a somewhat heuristic technique to incrementally update the probability of previously visited places based on newly matched places. Later approaches consider the probability of certain

place visitations as a whole sequence, rather than incrementally, as shown in this approach.

5. Richter et al. (2012) introduces the notion of reference points, which are semantic points associated with both detected trajectory stops and moves, to describe the journey of a tracked entity in a highly compressed way. In their work Richter et al. (2012) explain that reference points include junctions in the road network and places nearby to trajectory stops. In their approach places are matched to trajectory stops by simply querying an underlying contextual database (OpenStreetMap in this case) for the entity nearest to the stop. Of course, this entirely ignores the problem of stop/place disambiguation and place topology; however, if the goal is simply to compress the entity's journey, then some semantic inaccuracy may be acceptable.

6. Furletti et al. (2013), sharing some of the same authors as Spinsanti et al. (2010), present an extension of (Spinsanti et al., 2010) with an approach that probabilistically matches trajectory stops to geographic places. For each stop, candidate places are found in a contextual database (OpenStreetMap and Google Places, in this case) that are within a given search distance of the stop and have opening hours that are relevant for the time the stop occurred. Once the candidate places are found for each stop, the probability of observing each place at that stop is calculated using a so-called "gravity model". Specifically, the probability is calculated by counting the total number of places in some category and dividing that total by the squared distance between the current stop and the closest candidate place, from that category. This approach does handle the problem of stop/place disambiguation using a probabilistic model; however, it does not consider place topology or the order of visited places in its probability calculation.

7. Yan et al. (2013) present a so-called "holistic" approach where they enrich raw GPS trajectories using the stop and move model (Spaccapietra et al., 2008), and then further enrich the trajectory stops and moves using points, lines, and regions. Specifically, the stop episodes are enriched using geographic regions and points of interest, while the moves are enriched by map-matching them to an underlying road network. The step where they match stop episodes to points of interest is particularly notable, because they use a Hidden Markov Model (HMM) to probabilistically model the most likely sequence of visited points, based on place types, spatial proximity, and the order of points visited. The usability of the HMM, they propose, is somewhat limited, because they do not provide any way to estimate the transition matrix (i.e the probability between different types of places). In their work the setting of the transition matrix is done by hand and the authors state, "learning dynamic and personalized transition matrix [sic] is interesting but not the focus of this article". Additionally, by spatially intersecting regions with trajectory stop episodes they ignore the problem of place topology (i.e. what to do with overlapping places).

8. Moreno et al. (2014) present an extension of SMoT (Alvares et al., 2007) called "SMoT+", which matches stops to geometric places, even when the places are contained within one-another (i.e topologically nested). Specifically, SMoT+ finds the most-nested place each stop could have occurred. I highlight, that as far as I know, SMoT+ is the first place-matching approach to introduce the

problem of matching stops to places when the places are topologically nested (i.e shopping centres with shops inside). I consider the handling of topologically ambiguous places an important step for future approaches; however, SMoT+ only handles the simplest cases, where one place is entirely nested within another place. In real-world place datasets, such as OpenStreetMap, far more pathological scenarios are commonplace. For example, SMoT+ makes no specification as to how to handle a stop that occurs in the region where two place geometries intersect, but neither contains the other. Similarly, much like SMoT, SMoT+ does not logically handle matching a place to a stop when the trajectory stops, and intersects, with several places at the same time (SMoT/SMoT+ would produce several stops, switching between places if the duration was long enough in each, or simply no stop at all, if the durations were too short).

9. Boukhechba et al. (2015) present one of the only place-matching approaches that does place-matching using streaming trajectory data. The first part of their approach is a novel technique to classify trajectory subsequences as stopping, moving, or moving-with-activity (i.e walking on foot in a shopping centre). They perform this classification using a modified K-means clustering, that considers trajectory orientation and velocity. Once the trajectory is transformed into a series of stop/move/move-with-activity clusters, each cluster is associated with the nearest OpenStreetMap entity. This simple distance-based, nearest-entity matching is similar to the work of Richter et al. (2012) and therefore this approach, similarly, ignores the problem of stop/place disambiguation and matching to topologically complex places.

10. Lv et al. (2016) present a probabilistic approach for matching people's trajectories to the types of places they visited. Their approach begins by constructing a HMM that models the probability of observing a particular sequence of visited places, being caused by some sequence of trajectory stops. Specifically, their HMM specification considers the spatial, place-type, temporal, and sequential attributes of the input stop/places pairings. However, I highlight that the usability of this approach is somewhat limited because both the temporal and sequential modelling proposed by Lv et al. (2016) require ground-truth trajectories labelled with visited places. However, one particular advantage of their approach, is that it neatly avoids the problem of matching stops to places with complicated topologies. Basically, stop and place geometry are only considered during an initial intersection test to generate candidate places for each stop. All the other probabilistic modelling in this approach simply does not consider the spatial features of stops or places; thus, by extension, the issue of place topology is moot. I highlight that this design choice contrasts with nearly all previous works that heavily make use of spatial proximity to perform place-matching.

11. Gu et al. (2017) present an approach to fuse people's trajectory data with Foursquare check-in data, to create stop/move, matched ground-truth data. They do this by matching a check-in to a stop that is within both a spatial and temporal threshold (multiple candidates are resolved down to the one with the closest temporal similarity). Then, using this ground-truth data, they construct a probabilistic model based on the "general choice model" (Kumar et al., 2015) that calculates the probability of observing each place as a result of the input stop. In their approach, the general choice model is realised, as

calculating the place probability by considering both the distance to the place and the number of closer candidate places. I highlight that such a model is more limited than some previous approaches, as it does not consider the order of visited places, nor the place topology. Furthermore, their approach is generally more limited than some others because it requires sufficient Foursquare check-ins in some locale in order to initially construct the ground-truth data required to function effectively.

2.5.2 Critical Review

My first observation is that compared to the stop/move detection literature I reviewed in Section 2.4, the reviewed place-matching literature is fairly varied, and certainly does not tend towards extending a single approach like the reviewed stop/move approaches do. The explanation for this is likely that the problem of place-matching can be solved in many ways, so no existing approach has yet emerged as the clear choice. I speculate that this may be because, in general, the place-matching problem, unlike the stop/move detection problem, relies on additional data sources (i.e. the place databases) and, therefore, each approach is somewhat specific to the type of contextual data used. For instance, recall in Section 2.5: I highlighted that OpenStreetMap supports polygon, point, and line geometries for places, whereas Google Places and Foursquare only support points. Thus, it seems likely the problem of place-matching will evolve as additional useful sources of information become available to the public. For example, recently Zhu et al. (2016) demonstrated an approach for learning activities from tweets, and Beber et al. (2016) use that approach with geo-referenced tweets, Foursquare places, and trajectory data to classify trajectory stops into certain activities based on temporal features of the stop. Although, the place-matching is strictly done by associating the stop to the nearest Foursquare place; in this case, this example demonstrates how an additional data source can be combined with feature of the trajectory (i.e the temporal features of a stop) to make further inferences. Another example of this sort of data mash-up is presented by Gu et al. (2017), who introduce an approach for fusing Foursquare check-ins with people's stops for the purpose of place-matching. The disadvantage of many of these contextual data sources, such as Google Places, Foursquare, and Twitter, is that they impose rate limits on the number of queries a user can perform in some time-frame. This, ultimately, means that approaches based on these sources may have some issues scaling to handle truly large datasets.

My second observation is that a number of approaches have proposed probabilistic models to solve the problem of place-matching (Liao et al., 2007; Spinsanti et al., 2010; Furletti et al., 2013; Yan et al., 2013; Lv et al., 2016; Gu et al., 2017). A probabilistic model seems naturally suited to the problem because each stop is somewhat spatially uncertain; thus, any nearby places could be the cause of the stop, so each place should be given a probability that it was the true stop place (i.e. considering perhaps the spatial, temporal, or place-type features). Although some approaches (Furletti et al., 2013; Gu et al., 2017) only consider calculating the probability between a stop and its candidate places, there is also another aspect of the problem that can be modelled probabilistically. Specifically, if one stop has candidate places with associated likelihoods, and the next stop has candidate places with associated likelihoods, then there is also an opportunity to model the probability of transitioning between any place at the first stop to any place at the second stop. It seems reasonable to assume that people visit certain types of places

in order, due to personal routines, task-specific reasons, or convenience; thus, both Yan et al. (2013) and Lv et al. (2016) incorporate a so-called transition probability between places into their models. I consider this a step in the right direction for two reasons: the first is that it naturally incorporates the sequential nature of the trajectory into the problem, as opposed to considering each stop/place match in isolation; the second is that it utilises an extra source of data (the sequence of places) that was not used before, and all extra data is useful in such an uncertain problem.

My third observation is that while some of the more recent place-matching approaches address stop/move disambiguation, place topology, and the order of visited places, they still have some limitations that impact their usability for my scenario of semantic trajectory data mining. Specifically, they either require difficult-to-set, user-specified parameters (i.e. Yan et al. (2013) require a user-specified transition matrix), or require a ground-truth training dataset for their model that is often infeasible to generate (i.e (Lv et al., 2016)).

2.5.3 Literature Gaps

Analysing Table 2.3, it is apparent that none of the reviewed place-matching approaches meet all five of reviewed features. As I discussed above, even the most well rounded approaches (Yan et al., 2013; Lv et al., 2016) have some limitations. Despite those limitations, I do consider a probabilistic place-matching approach, similar to the works of Yan et al. (2013) and Lv et al. (2016), to be the most suitable in terms of considering stop/place disambiguation and the order of visited places. Thus, I propose to investigate a probabilistic place-matching approach that handles: points and polygon place geometries; stop/place disambiguation; the order of visited places; and, place topology —all in an unsupervised setting without the need for expert-level parameter setting. This investigation into a probabilistic place-matching approach is detailed in Chapter 6.

2.6 Research Hypotheses

From my literature reviews of each topic I have formulated the following research hypotheses to guide each chapter:

- **Trajectory simplification hypothesis.** *A framework that generalises the problem of trajectory simplification into a combination of significance scoring functions and processing strategies, will be able to extend existing poly-line simplification algorithms to trajectory simplification approaches. These new simplification approaches will outperform their original counterparts in regard to spatio-temporal error metrics such as SED.*

- **Vehicle trajectory pattern mining hypothesis.** *Mining map-matched vehicle trajectories, using a contiguous sequential pattern mining approach that prunes the pattern output, will result in a set of roads-driven that are unique (non-overlapping) and less redundant than the output from existing contiguous-closed or max-sequential pattern mining approaches.*

- **Probabilistic stop/move hypothesis.** *A probabilistic stop/move detection algorithm with a minimum stop probability threshold parameter will allow low probability (i.e. ambiguous) stop classifications to be filtered out by the user; this will*

ultimately result in a more controllable false-positive rate and a higher overall classification accuracy than existing approaches.

- **Probabilistic place-matching hypothesis.** *A probabilistic place-matching algorithm that considers stop/place disambiguation, place topology, and the sequence of visited places will be more accurate than place-matching approaches that form matches by using spatial intersections.*

Chapter 3

A Framework of Spatio-temporal Trajectory Simplification Methods

In this chapter I focused on the challenge of data complexity and combating it by pre-processing free-space trajectories. Specifically, I investigated reducing data complexity through simplification of trajectories in free-space. From my review of existing trajectory simplification approaches in Section 2.2, it seemed apparent that the task of trajectory simplification can be generalised as the combination of two concepts: an entry significance scoring function and a processing strategy; such a generalisation has not been explored in existing works. Thus, in this chapter I explored this generalisation and introduced a framework to create size-bounded, offline, trajectory simplification approaches by combining significance scoring functions with processing strategies. Specifically, using my framework I created and evaluated a number of trajectory simplification approaches in terms of running time, synchronised Euclidean distance, synchronised area, and preserving visitation through detected regions-of-interest. Experiment results demonstrated that using my proposed framework I was able to create a number of efficient and effective spatio-temporal trajectory simplification methods that outperformed their original spatial-only counterparts. The topics covered in this chapter are illustrated in bold in Figure 3.1.

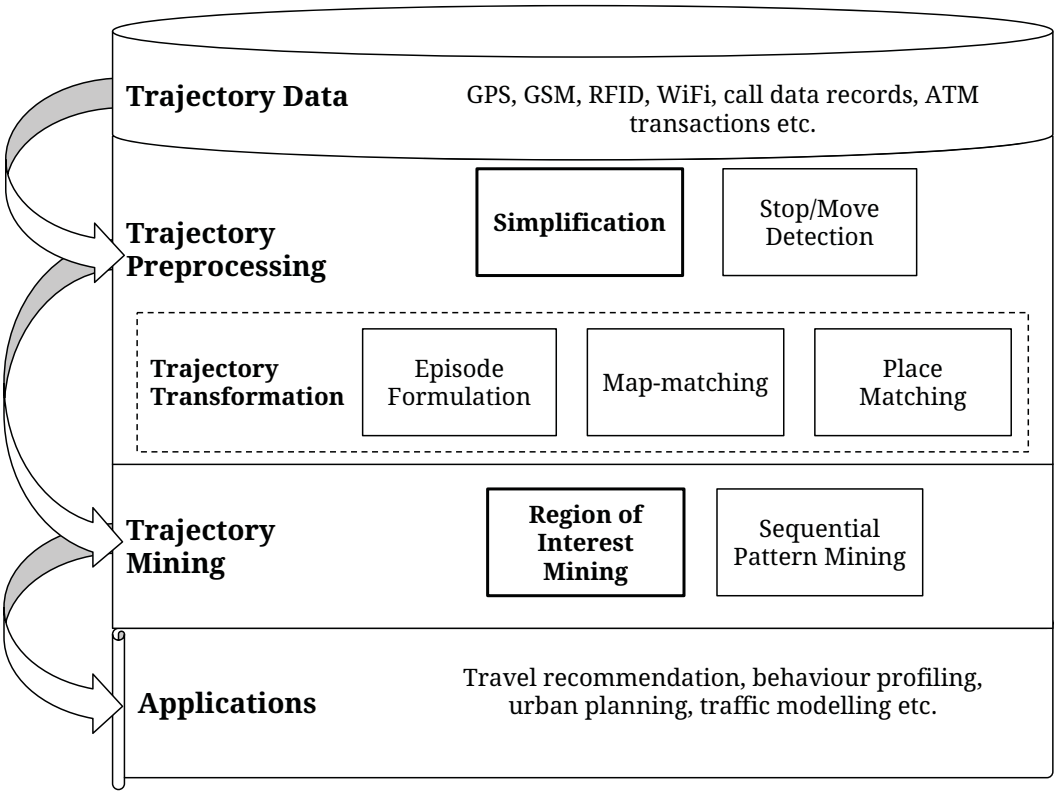


FIGURE 3.1: Overview of the trajectory data mining and knowledge discovery stages and the specific topics (the bold rectangles) covered in this chapter.

As a full disclosure, readers should be aware that the work in this chapter went on to be published as: Bermingham, L. and Lee, I. (2017). A framework of spatio-temporal trajectory simplification methods. In the: *International Journal of Geographical Information Science*, 31(6), pp.1128-1153.

3.1 Introduction

Due to the adoption and growth of sensor-enabled mobile phones and GPS tracking technology, there exists huge amounts of trajectory data (Kang and Yong, 2010). Such large trajectory datasets can be mined to uncover patterns and trends that can lead to new knowledge discoveries. However, due to being collected from sensors, this trajectory data is often complex, redundant, and noisy. Thus, in data-rich environments it is highly beneficial to pre-process trajectories to remove the unnecessary details and noise before data mining, whilst still preserving the underlying structures and patterns of the tracked entities. In this chapter I refer to this process as *trajectory simplification*.

Trajectory simplification is useful to deal with growing dataset sizes, but when one considers the ways in which trajectory data is typically gathered, trajectory simplification becomes a requirement. Specifically, common data generation platforms such as mobile phones and GPS systems are imperfect; it is an accepted pitfall that these devices suffer from spatial uncertainty. One of the common ways to combat this spatial uncertainty is to redundantly oversample in the hope that some less noisy entries will also be recorded (Long et al., 2013). The next step after this redundant oversampling, is to remove the redundancy and preserve the underlying movements as effectively as possible: this is where trajectory simplification is useful.

Recalling my findings from the trajectory simplification literature review in Section 2.2 I identified the following gaps in the state-of-the-art that motivate this chapter:

1. There are other poly-line simplification approaches, besides the Douglas-Peucker approach, that may contain unique concepts that can be extended to the problem of trajectory simplification.

2. The concept of size-bounded trajectory simplification naturally generalises the overall problem by replacing algorithm-specific parameters with a concrete simplification percentage, yet this generalisation has not been widely explored.

3. It appears that trajectory simplification approaches can neatly be generalised by two concepts: an entry significance scoring function, and a processing strategy. Yet, such a generalisation has not been exploited to quickly create various combinations of trajectory simplification algorithms.

Thus, considering these motivations I use this chapter to investigate a generalised trajectory simplification framework that can create size-bounded, offline, trajectory simplification algorithms, consisting of an entry significance scoring function that is executed by some processing strategy. Specifically, I aim to present a general framework that can combine scoring functions and processing strategies together, to quickly make new and tailored trajectory simplification approaches. Some of these scoring strategies and processing strategies come from existing poly-line simplification approaches, and others, from existing trajectory simplification approaches. Overall, in this chapter, I present the following contributions:

1. a general trajectory simplification framework that I use to create eight different spatio-temporal trajectory simplification methods;

2. four new $O(n)$ spatio-temporal trajectory simplification methods;

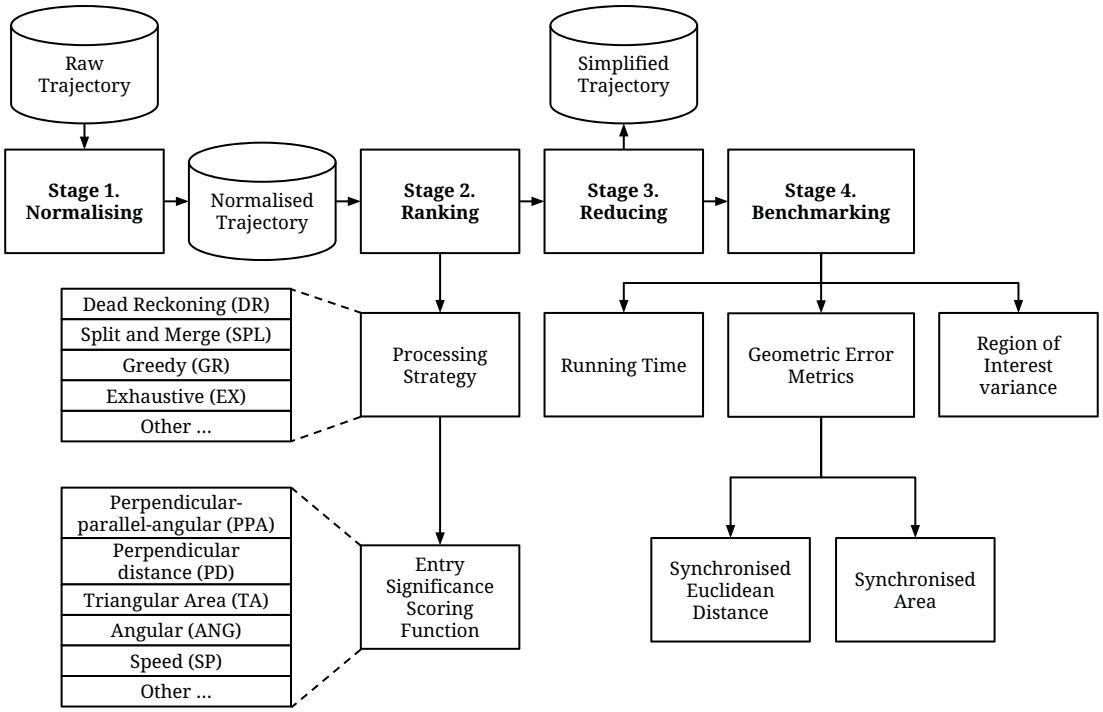


FIGURE 3.2: My framework for trajectory simplification, with each major stage in bold.

3. an empirical comparison of efficiency and effectiveness between twelve trajectory simplification methods; and

4. two novel trajectory similarity benchmarking metrics, one called synchronised area and the other a RoI visitation technique.

The rest of this chapter is organised as follows: Section 3.2 introduces my generic framework for creating spatio-temporal trajectory simplification methods, and also provides details about each stage of the framework; Section 3.3 provides various experimental results including running time efficiency and quantitative effectiveness benchmarking metrics; and lastly, Section 3.4 summarises the contribution of my framework and highlights its use-cases.

3.2 Framework

In this section I present my framework for creating and evaluating size-bounded, offline, trajectory simplification algorithms. Figure 3.2 illustrates the modules and four major stages that define the framework: (1) *normalising*; (2) *ranking*; (3) *reducing*; and (4) *benchmarking*.

In the first stage, normalising, a raw spatio-temporal trajectory is transformed into the unit cube $[0, 1]^3$ so that Euclidean calculations can be performed (see Section 3.2.1 for more details). In the second stage, ranking, a processing strategy and an entry significance scoring function are combined to calculate the significance of each entry in the normalised trajectory, with respect to some criteria. This is a crucial stage in my framework because this combination of processing strategy and significance scoring function is what defines each simplification algorithm. The purpose of a significance scoring function is to calculate a score for a

given entry, and, thus, quantitatively indicate that entry's significance in the trajectory. Significance scoring functions calculate scores using a given entry and some combination of other entries in the trajectory. For example, the Triangular Area function calculates the area of the triangle formed by a given entry and its two neighbours (see Section 3.2.4 for more details). The purpose of a processing strategy is to decide the order in which entries are processed. For example, the Dead Reckoning strategy processes entries from beginning to end, not removing any previously processed entries as it progresses; the Exhaustive processing strategy, however, processes entries by finding the most significant entry, storing and removing it, then repeating this process until all entries are processed (see Section 3.2.3 for more details).

The reason that the processing order of entries (i.e. the processing strategy) makes any difference to the results is because of the significance scoring functions. Recall, each significance scoring function calculates the significance of a given entry by using that entry and some other entries. Thus, when one considers that some processing strategies such as the Dead Reckoning do not remove entries, whilst others such as the Exhaustive strategy do, it becomes apparent that the pool of entries available during significance scoring will change throughout the ranking process.

In the third stage, reducing, the trajectory is simplified by removing a specified percentage of entries. The simplification occurs by removing entries in ascending order of significance. I highlight that if trajectory simplification is required at varying percentages, my framework is quite efficient: as long as the significance scores for each entry are stored, no additional processing time needs to be spent on re-normalising or re-ranking. In the final stage, benchmarking, the efficiency and effectiveness of trajectory simplification methods are tested. The efficiency is tested by measuring the simplification method's running time on various real world and synthetic trajectories, whilst the effectiveness is tested by measuring the Synchronised Euclidean Distance, Synchronised Area, and RoI variance. Briefly, Synchronised Euclidean Distance and Synchronised Area measure the sum of Euclidean distances and areas, respectively, between the raw entries and their projected (i.e temporally synchronised) counterparts on the simplified trajectory (see Section 3.2.7 and Section 3.2.7 for more details). These metrics measure the amount of spatio-temporal variance between the raw and simplified trajectories. However, these metrics in no way guarantee that the underlying visitations (and hence the meaning) of the trajectory are preserved after trajectory simplification occurs.

In tasks such as data mining where pre-processing via simplification is often required, it is essential the underlying meaning of the trajectory does not change. Therefore, to quantify the change in visitations between the raw and simplified trajectories I measure the RoI variance. The RoI variance partitions the study area into cells, records the density of each cell by recording the number of trajectory visitations, and then finds the RoIs by merging neighbouring cells that are above a certain visitation threshold (see Section 2.1.3 for a recap). This process occurs for both the raw and simplified trajectories, and the cellular distance between the resulting raw and simplified RoIs is the variance score that I record (see Section 3.2.7 for more details).

Overall, together, stages 1-3 represent a general methodology of trajectory simplification. I highlight that because scoring functions and processing strategies are effectively modules, this means that adding a new processing strategy or significance scoring techniques into my framework will result in a number of new

trajectory simplification combinations, available for use. One advantage of creating trajectory simplification methods in this way, is that different combinations of processing strategies and significance scoring functions can easily be combined to produce new trajectory simplification methods. Another advantage is that it is simple to recreate existing simplification methods using my framework (assuming one can identify the processing strategy and significance scoring function). In Section 3.2.3 and Section 3.2.4, I demonstrate the process of recreating some existing simplification methods by Douglas and Peucker (1973), Visvalingam and Whyatt (1993), and Lee et al. (2007), using my framework.

In the following subsections, I provide more details explaining the purpose and process for each stage in my framework.

3.2.1 Normalising Trajectory Entries

In my framework trajectory sequences are simplified by removing insignificant entries. This requires determining the significance of each entry; however, computing significance often requires calculations such as area, angle and distance between entries. These calculations operate in Euclidean space, yet my aim is to score the significance of spatio-temporal trajectory entries in the format $\{spatial, spatial, temporal\}$ (non-Euclidean). Clearly, space and time values are semantically different and cannot be treated as analogous. One way to perform Euclidean operations is to normalise the spatio-temporal data into Euclidean space.

One such example is the trajectory simplification work of Gudmundsson et al. (2009), who use a mapping parameter, μ, "that transforms time units into space units". They then use this mapping parameter for the temporal dimension, which allows them to perform Euclidean distance calculations. Equation 3.1 is the mapping approach by Gudmundsson et al. (2009).

$$P_{xyz} = \{x, \mu \cdot t, z\}, \tag{3.1}$$

where, P_{xyz} is a transformed trajectory entry (a point in Euclidean space); x, z, t are the spatial coordinates (x, z) and the time-stamp (t) of the trajectory entry; and μ is the mapping parameter between time and space (i.e. 1 time unit is μ space units).

Another example is the work of Oliveira et al. (2013), who use similar spatio-temporal mapping to perform shared nearest-neighbour clustering. Their method uses a spatio-temporal distance function that normalises, separately, both space and time into the range [0,1].

$$D_{st}(W_s, W_t, x_1, x_2, z_1, z_2, t_1, t_2) = W_s \cdot \frac{D_s(x_1, x_2, z_1, z_2)}{S_{max}} + W_t \cdot \frac{D_t(t_1, t_2)}{T_{max}}, \tag{3.2}$$

where, D_{st} is the spatio-temporal distance; D_s, D_t are the spatial and temporal distance functions, respectively; W_s, W_t are the spatial and temporal weightings, respectively (a zero weighting fully discounts a dimension); S_{max} and T_{max} are data-driven maximum values to normalise space and time, respectively; x_1, x_2, z_1, z_2 are the spatial coordinates; and t_1, t_2 are the timestamps.

Normalising of spatio-temporal data is a common technique to maintain validity of Euclidean operations. Euclidean operations are a requirement of my framework and therefore I, too, opt to normalise the data before simplification. The normalisation I adopt is inspired by Kang and Yong (2010), who use min-max normalisation to normalise the spatial and temporal dimensions of their data into

unit space $[0,1]^3$ before clustering. In the context of trajectory simplification, min-max normalisation normalises a spatio-temporal point in a given trajectory into unit space $[0,1]^3$ by using the upper and lower spatio-temporal bounds of that trajectory.

I present my equation for spatio-temporal min-max normalisation in Equation 3.3. Using min-max normalisation to transform a spatio-temporal trajectory into the unit cube is useful because it automatically applies a data-defined, but heuristic, mapping between the spatial and temporal dimensions. I use the following example to illustrate the implicit mapping between space and time units that occurs by using min-max normalisation in this way. Consider, a trajectory with a duration of 100 seconds that travels 500 meters in a straight line. Using the min-max normalisation I have outlined in Equation 3.3, it implies that 1 second equals 5 meters for this example. The parameters W_t and W_s can skew this mapping but, in general, I use a W_t and W_s of 1 and rely on this data-defined normalisation to compare spatial and temporal dimensions. The only exception I make to this is to use a W_t of zero to discount the temporal dimension when performing comparisons (i.e. comparing the spatio-temporal Douglas-Peucker to the original spatial-only method).

$$\begin{aligned} \Delta S &= \max(x_{max}, z_{max}) - \min(x_{min}, z_{min}), \\ E_i &= \{W_s \cdot \frac{x_i - x_{min}}{\Delta S}, W_t \cdot \frac{t_i - t_{min}}{t_{max} - t_{min}}, W_s \cdot \frac{z_i - z_{min}}{\Delta S}\}, \end{aligned} \tag{3.3}$$

where, x_i, t_i, z_i are the coordinates of a spatio-temporal trajectory T, such that $T = \{\langle x_1, t_1, z_1\rangle, \ldots, \langle x_n, t_n, z_n\rangle\}$; E_i is a normalised trajectory entry (a point in the unit cube $[0,1]^3$) in the normalised trajectory N, such that $N = \{E_1, \ldots, E_n\}$; ΔS is the spatial range of the trajectory (bounding rectangle of the trajectory $[x_{min}, x_{max}][y_{min}, y_{max}]$); W_s is the weighting parameter on the spatial dimension; and W_t is the weighting parameter on the temporal dimension.

3.2.2 Ranking Trajectory Entries

One commonality I observe between many simplification algorithms is that, at their core, they identify significant entries and preserve them. The difference in results between various simplification algorithms is mostly driven by two factors: the first is how entry significance is determined —the significance scoring function; and the second is how the entries are processed —the processing strategy. I use this observation as a foundation of my framework, specifically at its second. Stage 2 takes a normalised trajectory and ranks each of its entries using a processing strategy and significance scoring function. As mentioned in Section 3.2, in my framework processing strategies and scoring functions can easily be added and combined to create new trajectory simplification methods. In the following sections I introduce the details for a number of processing strategies and scoring functions.

3.2.3 Processing Strategies

Processing strategies, effectively, determine the number of passes performed over the trajectory entries, and also whether entries are removed, grouped, or left alone during significance scoring. Different processing strategies directly affect which entries are available to score the significance of a specific entry. Consider, a scoring function that computes the significance of a given entry as the perpendicular distance between that entry, and the line segment between the first and last entries in the trajectory. If one applies a single-pass processing strategy where each entry is

processed one-by-one, and no entries are removed, this perpendicular scoring function may score entries that are collinear to the first-and-last entries as insignificant, even if they are a large distance away from their neighbouring entries.

However, if one applies a split-based strategy that divides the entries into groups, and performs many passes over the entries, then this scenario is less likely. In the first pass, a split-based strategy finds the entry that has the largest perpendicular distance between itself and the line-segment formed by the first and last entries. Next, this entry and its score are stored. Following this, the entry is removed from the trajectory, thus dividing the trajectory into two groups. The first and last entries in both groups then define two new line segments that are used for scoring the significance of entries in those groups. Once again, this allows the most significant entry in both groups to be stored and removed, which, in turn, creates further subdivided groups. This process continues until the groups can be divided no further. Due to how the split-based strategy controls the grouping of entries, the significance scoring and, thus, the overall simplification result too, are both vastly different to the incremental, single-pass approach. However, I highlight that the single-pass strategy is not without merit; it is computationally far faster than the split-based strategy. Overall, the selection of a processing strategy can often be viewed as a trade-off between simplification efficiency and effectiveness.

Understandably, there are many processing strategies that already exist; I choose to implement a few of the most popular. Based on the works of Meratnia and By (2004) and Long et al. (2013), I adopt these three in my framework: *dead reckoning*, *split*, and *greedy*. I also incorporate an approach that has not yet been formalised, which I call *exhaustive*. I now present a description of each.

Dead Reckoning Strategy (DR)

The name “dead reckoning" comes from a navigation technique where the current position is computed using only the previously known position. Inspired by that navigation technique, Lange et al. (2008) and Lange et al. (2009) proposed a trajectory simplification processing strategy that they also call dead reckoning. In the context of trajectory simplification dead reckoning is an incremental processing strategy that performs a single pass over the trajectory entries. It computes the significance score of each entry, one by one, and does not remove any previously processed entries. The advantage of the dead reckoning processing strategy is efficiency: it has $O(n)$ time complexity. I prefix the dead reckoning trajectory simplification methods with the naming convention “*DR*".

Split-based Strategy (SPL)

The split-based processing strategy, or “top-down" as it is called by Meratnia and By (2004), is a divide-and-conquer approach. A number of existing trajectory simplification algorithms (Cao et al., 2006; Gudmundsson et al., 2009) are split-based because they are built on the popular split-based algorithm by Douglas and Peucker (1973). The concept of the split-based processing strategy is to perform a pass over the data and find the most significant entry, the *split point*. Using the split point, the sequence is divided and additional passes are performed on the divided subsets to find their split points. The process of dividing and locating split points is repeated until all the recordings are processed. As each split point is found it is stored and, thus, when all entries are processed, this collection of split points forms the ranked trajectory sequence. Split-based trajectory simplification

algorithms generally have a time complexity of $O(n^2)$. However, in comparison to dead reckoning-based algorithms, they typically produce a simplification that represents the original data more accurately. I prefix the split-based trajectory simplification methods with the naming convention "*SPL*".

Greedy Strategy (GR)

The greedy processing strategy is based on the notion of accumulating insignificant entries until an entry surpasses a significance threshold, and has to be preserved. Examples of greedy trajectory simplification in the literature include the works of Meratnia and By (2004), Lee et al. (2007), and Kolesnikov (2011). For example, Lee et al. (2007) introduce their greedy trajectory simplification algorithm that stores recordings based on a trade-off between "preciseness and conciseness". They define preciseness as the sum of the $\log_2$ value of the perpendicular, parallel, and angular distance between the previously stored entry (or the first entry if one is yet to be stored) and the current entry. Additionally, they define conciseness as the sum of the $\log_2$ value of Euclidean distances between each of the entries between the previously stored entry and the current entry. When preciseness, plus a user-specified threshold, is greater than conciseness the current entry is considered significant and is stored. However, this threshold does not map simply to a percentage of entries to preserve and this, therefore, makes it difficult to select a threshold because it is context and case dependent. Thus, I make a modification to their approach in order to automate the threshold selection. I achieve this by initialising the significance threshold to its maximum value, performing a pass over the sequence, removing and storing significant entries, decreasing the significance threshold and, then, repeating the process until the required number of entries remain. The trade-off for this parameter automation is that the time complexity becomes $O(n^2)$. I prefix greedy-based simplification methods with the naming convention "*GR*".

Exhaustive Strategy (EX)

The exhaustive processing strategy finds the most significant entry in the trajectory, removes and stores it along with its score, and, then, repeats this process until all entries have been removed and stored. Note, when an entry is removed and stored, only its neighbouring entries, that have not yet been stored, need to have their significance scores updated. To the best of my knowledge, this processing strategy has not yet been formalised in the field of trajectory simplification. The only approach I discovered in the literature that implemented this strategy was the line simplification algorithm introduced by Visvalingam and Whyatt (1993). In their approach, Visvalingam and Whyatt (1993) use the triangular area formed by an entry and its neighbours to calculate significance. As significant entries are preserved and removed, their algorithm then recalculates the triangular areas of the neighbouring entries. This process is repeated until every recording has been stored. The time complexity of this process is $O(n \log n)$. I prefix the exhaustive trajectory simplification methods with the naming convention "*EX*". As mentioned, the Exhaustive processing strategy has not yet been formalised in trajectory simplification literature, therefore, in Algorithm 1 I introduce it.

Readers, please note that the other processing strategies I discuss in this paper, SPL, GR, and DR, all have the same input and output formats as Algorithm 1; that is, they also take a normalised trajectory and significance scoring function as input

Algorithm 1 EX

Input: N, a normalised trajectory, $\{E_1, \cdots, E_n\}$, *Significance(N,I,E)*, a scoring function.
Output: I, a list of trajectory entries inserted in ascending order of significance.
Assign I to $\emptyset$. *//List of entries ordered by significance.*
Assign R to $\emptyset$. *//Map sorted in ascending order of entry significance, stored as $\{Entry, Score\}$.*
Assign T' to $N\{E_2, \cdots, E_{n-1}\}$. *//Entries to process, all but the start and end.*
while $T' \neq \{NULL, NULL\}$ **do**
    for Entry E in T' **do**
        if $E \neq NULL$ **then**
            Assign S to **SIGNIFICANCE**(N,I,E). *//Find the significance of the current entry using one of the significance functions.*
            Put entry-score, $\{E, S\}$, into R.
        end if
    end for
    if $R \neq \emptyset$ **then** *//If there is still entries to transfer to I.*
        Assign E_{min} by removing head of R. *//Remove least significant entry.*
        Put entry E_{min} into I.
        //Get the next subsequent entries, E_{next} and E_{prev}, affected by removing E_{min}, so long as the neighbours have not been processed already and are not the trajectory's start or end entry.
        Assign E_{next} to entry $\succ E_{min} \in T | E_{next} \notin I$.
        Assign E_{prev} to entry $\prec E_{min} \in T | E_{prev} \notin I$.
        if $E_{next} = E_n$ **then**
            Assign E_{next} to $NULL$
        end if
        if $E_{prev} = E_1$ **then**
            Assign E_{prev} to $NULL$
        end if
        Assign T' to $\{E_{prev}, E_{next}\}$. *//Neighbour entries need their scores recalculated.*
    else
        Assign T' to $\{NULL, NULL\}$
    end if
end while
Put E_1 and E_n into I. *//Preserve the start and end entries last, they are the most significant.*
return I.

and output. However, as their general formulation has already been presented in the literature, please refer to Meratnia and By (2004), Lee et al. (2007), and Long et al. (2013) for more details.

3.2.4 Computing Significance

At some stage in their algorithm, all of the processing strategies used in my framework require the ability to determine the significance of the current trajectory entry (for example, see Algorithm 1, Line 9). I now introduce a number of entry significance scoring functions that I incorporate into my framework —some from the literature, and others new. All of the scoring functions I introduce are illustrated in Figure 3.3.

Specifically I present three significance scoring functions from the literature: Perpendicular Distance (PD) (Douglas and Peucker, 1973), Triangular Area (TA) (Visvalingam and Whyatt, 1993), and Perpendicular-Parallel-Angular (PPA) (Lee et al., 2007). In addition to those, I also present two scoring functions that, to my knowledge, have not yet been formalised: ANGular (ANG) and SPeed (SP).

Perpendicular Distance (PD) Scoring Function

The PD scoring function was popularised by the 2d poly-line simplification algorithm created by Douglas and Peucker (1973). In their algorithm, Douglas and Peucker (1973) compute the significance of a given entry as the perpendicular distance between the current entry and the line formed by connecting the poly-line's first and last entries. I make two modifications to their scoring function for my purposes. First, because my data is spatio-temporal, I compute perpendicular distance in 3d space. Second, to facilitate the dead reckoning processing strategy, my version is passed a boolean flag. When the flag is true, significance calculation is performed using the perpendicular distance between the current entry and the line formed by connecting that entry's two neighbours (rather than the line formed by the trajectory's first and last entries). I present my modified version in Equation 3.4, and a visual illustration in Figure 3.3(a). All of my trajectory simplification methods that use this significance scoring function are suffixed with "*PD*".

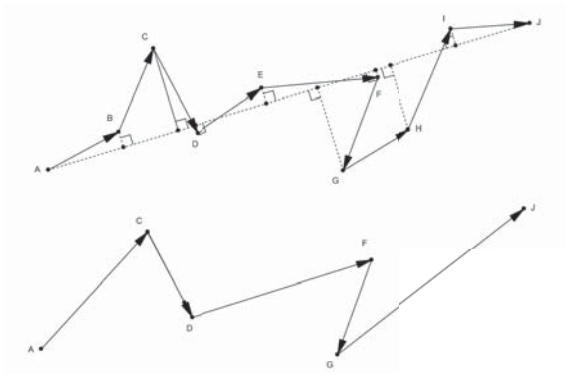
$$PD(E_i) = d_{\perp}(E_s, E_e, E_i) \tag{3.4}$$

Borrowing the terms defined in Equation 3.3, the remaining terms in Equation 3.4 are as follows:

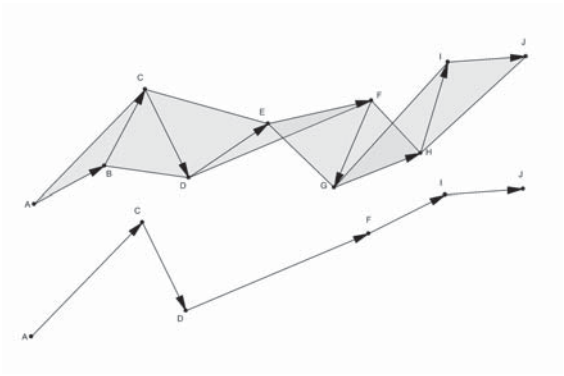
- $d_{\perp}(E_s, E_e, E_i)$, the perpendicular distance between the line $\overrightarrow{E_s, E_e}$ and the point E_i.
- E_s, either E_{i-1} or E_1 (the former case is used for dead-reckoning).
- E_e, either E_{i+1} or E_n (the former case is used for dead-reckoning).

Triangular Area (TA) Scoring Function

The TA scoring function originates from the work of Visvalingam and Whyatt (1993), who use it in their poly-line simplification algorithm. The concept of this function is to score the significance of a given entry as the area of the triangle formed by the current entry and its two neighbours. The underlying reasoning is

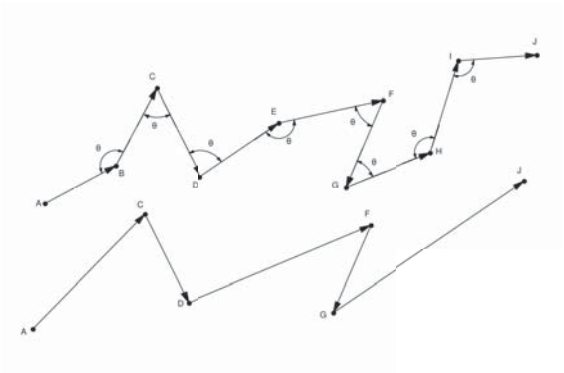


(A) PD

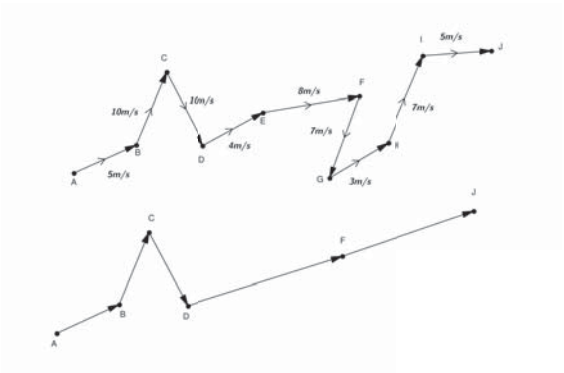


(B) TA

(C) PPA



(D) ANG



(E) SP

FIGURE 3.3: Visualisation of each trajectory significance scoring functions. The top half of each shows the inner workings, whilst the bottom-half shows the simplifications.

that larger areas represent larger changes in the sequence and should be preserved, whereas a small triangular area, such as that formed by an entry with two collinear neighbours, will be regarded as insignificant. The only change I make to the function presented by Visvalingam and Whyatt (1993) is that I calculate triangular area in 3d space. I present my modified version in Equation 3.5 and a visual illustration in Figure 3.3(b). All of my trajectory simplification methods that use this significance scoring function are suffixed with "*TA*".

$$TA(E_i) = \triangle(E_{i-1}, E_{i+1}, E_i) \tag{3.5}$$

Borrowing the terms defined in Equation 3.3, the remaining term in Equation 3.5, $\triangle(E_{i-1}, E_{i+1}, E_i)$, is the area of the triangle formed by the normalised points E_{i-1}, E_{i+1}, and E_i.

Perpendicular, Parallel, and Angular (PPA) Distances Scoring Function

PPA is a multi-metric significance scoring function: it uses perpendicular, parallel, and angular distances to score entries. This function is introduced by Lee et al. (2007) in a simplification algorithm in their trajectory clustering framework, TraClus. Their algorithm computes the significance of entries as a trade-off between "preciseness and conciseness". Preciseness is represented by the accumulated PPA distance between sequential entries, processed thus far. Whereas conciseness is represented as the accumulated total of the $\log_2$ value of the Euclidean distances between each sequential entry processed thus far. Once the accumulated total of preciseness, plus the current entry's conciseness, outweighs the accumulated conciseness, that entry is considered significant and the accumulations are reset for continued processing. The only modification I make to their preciseness and conciseness measures, is that I extend any distance calculations to 3d space. As their significance scoring function contains two parts, for brevity's sake, I formally show only the details for the more complex part: preciseness (i.e. the PPA distance measure). I present the PPA distance measure in Equation 3.6, and an illustration in Figure 3.3(c)

$$PPA(E_i) = \log_2 |d(E_i, P_i)| + \log_2 |d(E_1, P_i)| + \log_2 |\measuredangle(\overrightarrow{E_n, E_1}, \overrightarrow{E_{i+1}E_i})| \tag{3.6}$$

Borrowing the terms defined in Equation 3.3, the remaining terms in Equation 3.6 are as follows:

- P_i, the perpendicular projection of E_i onto $\overrightarrow{E_1E_n}$.
- $d(E_i, P_i)$, the Euclidean distance between E_i and P_i.
- $d(E_1, P_i)$, the Euclidean distance between E_1 and P_i.
- $\measuredangle(\overrightarrow{E_n, E_1}, \overrightarrow{E_{i+1}E_i})$ the angle between the two vectors.

Readers, please note that in my implementation of Lee et al.'s 2007 algorithm, I also use the conciseness measure which, as mentioned, is the accumulated total of the $\log_2$ value of the Euclidean distances between entries. My methods that use the PPA scoring function are suffixed with "*PPA*".

Angular (ANG) Scoring Function

ANG is the first of my own scoring functions; however, it is quite similar to the scoring functions presented by Chen et al. (2009) and Long et al. (2013). The concept of ANG is that large directional changes in trajectory movement should be preserved. After normalisation, my data is in 3d space, therefore, I must compute the directional changes against each dimension, X, Y, and Z. I present the inner workings of ANG in Equation 3.7 and an illustration in Figure 3.3(d). All of my trajectory simplification methods that use this scoring function are suffixed with "*ANG*".

$$ANG(E_i) = 180 - \theta_{xy} + 180 - \theta_{yz} + 180 - \theta_{xz} \tag{3.7}$$

Borrowing the terms defined in Equation 3.3, the remaining terms in Equation 3.7 are as follows:

- θ_{xy}, the change in angle in the x,y dimensions around E_i, specifically, $\measuredangle(\overrightarrow{E_{i-1}, E_i} \times \{1,1,0\}, \overrightarrow{E_{i+1}, E_i} \times \{1,1,0\}) \bmod 180$.
- θ_{yz}, the change in angle in the y,z dimensions around E_i, specifically, $\measuredangle(\overrightarrow{E_{i-1}, E_i} \times \{0,1,1\}, \overrightarrow{E_{i+1}, E_i} \times \{0,1,1\}) \bmod 180$.
- θ_{xz}, the change in angle in the x,z dimensions around E_i, specifically, $\measuredangle(\overrightarrow{E_{i-1}, E_i} \times \{1,0,1\}, \overrightarrow{E_{i+1}, E_i} \times \{1,0,1\}) \bmod 180$.

Speed (SP) Scoring Function

The concept of the SP scoring function is that large changes in speed, between trajectory entries, indicate a significant movement and should, therefore, be preserved. Entry speed is an attractive heuristic because it naturally incorporates the spatio-temporal properties of trajectory data. Speed, however, does not consider changes in direction to be significant; therefore, I also augment the speed-score based on the magnitude of the directional variance. I present the speed-based scoring function in Equation 3.8, and an illustration in Figure 3.3(e). All of my trajectory simplification methods that use this scoring function are suffixed with "*SP*".

$$SP(E_i) = \frac{d(E_i, E_{i+1})}{\Delta time} \times \frac{\Delta_\theta}{180} \tag{3.8}$$

Borrowing the terms defined in Equation 3.3, the remaining terms in Equation 3.8 are as follows:

- $d(E_i, E_{i+1})$ the Euclidean distance between the two points.
- $\Delta time$, the change in time around E_i, specifically, $|E_{i+1}.y - E_i.y|$.
- Δ_θ, the change in x, z direction around E_i, specifically, $|\measuredangle(\overrightarrow{E_{i-1}, E_i} \times \{1,0,1\}), \{1,0,1\}) - \measuredangle(\overrightarrow{E_{i+1}, E_i} \times \{1,0,1\}), \{1,0,1\})| \bmod 180$.

3.2.5 Creating Trajectory Simplification Methods

Processing strategies and significance scoring functions have now been introduced, so, in this section I move onto combining them to create trajectory simplification methods. As mentioned previously, the naming convention for the created

simplification methods is to use the processing strategy as a prefix and the scoring function as a suffix. For brevity's sake, I try only a selection of possible processing-scoring combinations. The trajectory simplification methods I investigate are shown in Table 3.1.

Three of the processing-scoring approaches from Table 3.1 were chosen because they represent a spatio-temporal version of a method from the literature. First, SPLDP is the spatio-temporal version of the Douglas and Peucker (1973) algorithm. Second, EXTA is the spatio-temporal version of the Visvalingam and Whyatt (1993) algorithm. Last, GRPPA is the spatio-temporal version of the Lee et al. (2007) algorithm (with minor modifications to allow parameter tuning as discussed in Section 3.2.3). Readers, note that, during my testing I created 2d (spatial-only) versions of these algorithms (suffixed with "(2d)") so I could perform as close a comparison as possible to the original simplification methods. The 2d algorithms were created by assigning a temporal weighting of zero during the normalisation stage of my framework (see Section 3.2.1 for details).

All the additional combinations I investigated were either chosen because they may exhibit interesting properties, or because they provide a comparison against another combination. For example, the TA and PD significance scoring functions are only paired with non-linear processing strategies in the literature. However, these functions may prove to be effective when paired with a linear processing strategy such as dead reckoning, hence I chose to investigate DRDP and DRTA. Likewise for PPA, it is only paired with a greedy processing strategy in the literature, thus, to evaluate whether PPA is a more effective metric than just plain PD, I chose to investigate SPLPPA. Additionally, DRANG and DRSP are introduced for comparison and experimentation purposes against the other DR approaches, even though their experiment results are not promising, as is shown in Table 3.3.

3.2.6 Reducing Trajectory Entries

I now introduce the third stage of my framework, reducing, where the trajectory is simplified. Given that the trajectory entries have been ranked in order of significance in the previous stage, it is now straightforward to remove a user-specified percentage of the entries. Specifically, I simplify the trajectory by removing entries one by one in ascending order of significance. In other words, the the most insignificant entry is removed first, and then the next most insignificant entry, and so on until the specified percentage of entries has been removed.

Simplifying trajectories in this size-bounded way has several advantages over traditional error-bounded approaches. Instead of relying on user-supplied threshold parameters, such as in the work of Douglas and Peucker (1973), one now have full

	DR	**SPL**	**GR**	**EX**
PD	DRPD†	*SPLPD*‡		
TA	DRTA†			*EXTA*‡
PPA		SPLPPA‡	*GRPPA*‡	
ANG	DRANG†			
SP	DRSP†			

TABLE 3.1: Combinations of processing strategies and significance scoring functions investigated. Italic combinations are inspired from the literature. † $= O(n)$, ‡ $= O(n \log n)$

control of how many entries will be removed. Furthermore, if the significance scores of each entry are stored, one can perform simplification on the trajectory at varying percentages without spending any additional computation on significance scoring. However, the obvious drawback of requiring each trajectory entry to be ranked, is that unlike traditional approaches, such as that of Douglas and Peucker (1973), there is no opportunity to short-circuit computation without processing every entry at least once. In other words, all algorithms produced by my framework are bounded to never have a running time better than $O(n)$.

3.2.7 Benchmarking Trajectory Simplification Methods

In this section I introduce the final stage of my framework, the ability to evaluate trajectory simplification methods using benchmarking metrics. The purpose of these benchmarking metrics is to quantify the total error between the raw trajectory and its simplification. Generally, as simplification increases more error is introduced. Likewise, as simplification is reduced, error decreases. Thus, it is valuable to benchmark simplification methods to determine if they have a favourable simplification-error trade-off. Therefore, I perform benchmarking using three different simplification benchmarking metrics: Synchronised Euclidean Distance (SED), Synchronised Area (SA), and density-based RoIs. The reason for using three different benchmarking metrics to measure the effectiveness, is that certain simplification approaches can have a favourable trade-off in one benchmarking metric, but not another. For example, simplification approaches that use the PD scoring functions are expected to perform better in the SED metric, and, likewise, approaches that use the TA scoring function are expected to perform better in the SA metric.

Synchronised Euclidean Distance

In the literature, SED has been widely used for the purpose of calculating spatio-temporal trajectory displacement (Meratnia and By, 2004; Muckell et al., 2011; Chen et al., 2012; Liu et al., 2013). The concept is to calculate the sum of Euclidean distances between entries in the raw and simplified trajectories. However, the raw trajectory will inherently have multiple entries that are no longer in the simplified trajectory; the problem arises as to how one might measure the Euclidean distance between the simplified trajectory and the entries in the raw trajectory that have been removed. The solution is to project these removed entries back onto the simplified trajectory using their temporal dimension to linearly interpolate them to the correct position along the relevant line segment. An illustration of this process is given in Figure 3.4.

Formally, I define a raw trajectory as T, and a simplified trajectory as T', both consisting of a list of spatio-temporal entries, $\{E_1\langle x_1, z_1, t_1\rangle, \ldots, E_n\langle x_n, z_n, t_n\rangle\}$ where E_i is an entry and x, z are its spatial coordinates and t is its temporal value such as a time-stamp. Note that, $T' \subseteq T$. Therefore, if there exists an entry, $E_a \in T \wedge E_a \notin T'$, then such an entry must be *sychronised* back onto T' in order to calculate its contribution to the SED. Given E_a, I define $E_p = E \prec E_a | E \in T \wedge E \in T'$ and $E_s = E \succ E_a | E \in T \wedge E \in T'$. I define $L = \frac{E_a.t - E_p.t}{E_s.t - E_p.t}$, and thus, the synchronisation of E_a onto T' as, $E'_a = \langle E_p.x + L(E_s.x - E_p.x), E_p.z + L(E_s.z - E_p.z), E_a.t\rangle$. Given E_a and its synchronised E'_a, I formally define SED in Equation 3.9. Readers, please note that $d()$ is the Euclidean distance between the spatial coordinates of the two entries.

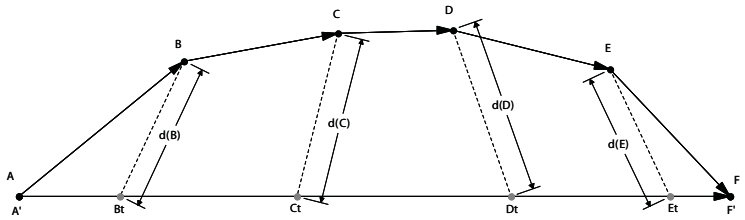
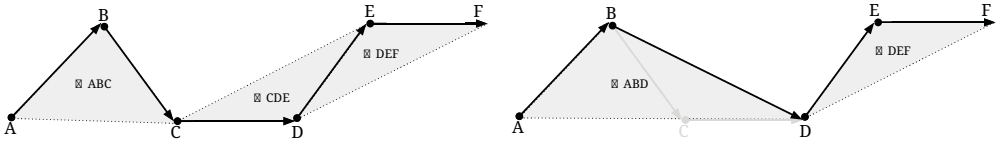
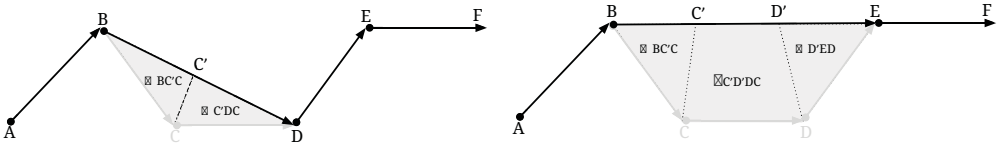


FIGURE 3.4: SED calculation between the raw trajectory $\{A, B, C, D, E, F\}$ and its simplfiied counterpart $\{A', F'\}$. Note: how the removed entries $\{B, C, D, E\}$ are projected back onto the simplified trajectory, $\{B_t, C_t, D_t, E_t\}$, using a linear interpolation between the temporal component of the preserved entries $\{A', F'\}$.



(A) The enclosed area of the raw trajectory $\{A, B, C, D, E, F\}$.

(B) The enclosed area of the simplified trajectory $\{A, B, D, E, F\}$.



(C) The SA between trajectory $\{A, B, D, E, F\}$ and the raw trajectory.

(D) The SA between trajectory $\{A, B, E, F\}$ and the raw trajectory.

FIGURE 3.5: The enclosed area metric compared to SA.

$$SED = \sum_{a=1}^{n} d(E_a, E'_a) | E_a \in T \wedge E_a \notin T'. \tag{3.9}$$

Synchronised Area

Similarly to SED, SA measures the total spatio-temporal displacement between raw and simplified trajectories. However, unlike SED, SA uses the *synchronised* area between entries as its metric to measure displacement. To the best of my knowledge, SA has not been introduced before. Similar approaches have been formalised by Ekdemir (2011) and Liu et al. (2013), who both use a metric where the difference between the enclosed area of the raw and simplified trajectories are compared. However, a flaw of the enclosed area metric is that the raw trajectory and its simplified counterpart can have the same enclosed area, which would incorrectly indicate no displacement. This flaw is visually illustrated using a raw trajectory and its simplified counterpart in Figures 3.5a and 3.5b. These figures demonstrate that even after simplification the enclosed area metric reports the same area. This is in contrast to Figure 3.5c that shows SA being computed between the same simplified trajectory and its original trajectory —one can see the change in area is represented more accurately. Figure 3.5d illustrates that, as the trajectory is simplified further, SA still computes an accurate change in area.

Recall, in Section 3.2.7 I showed the process to synchronise an entry back onto the simplified trajectory T' using its two surrounding entries, E_p and E_s. I now define an entry $E_r \in T$ and $E_{r-1} = E \prec E_r | E \in T$ (note, this is potentially different to E_p which I defined in Section 3.2.7). Additionally, I define E'_{r-1} as the result of synchronising E_{r-1} onto T'. Given E_r, E'_r, E_{r-1}, and E'_{r-1}, I formally define SA in Equation 3.10. Readers, please note that $A()$ is the area between the spatial coordinates of the four entries.

$$SA = \sum_{r=1}^{n} A(E_r, E'_r, E_{r-1}, E'_{r-1}) | E_r \in T. \tag{3.10}$$

Region-of-Interest Discovery

Both SED and SA calculate displacement geometrically. I now introduce my final simplification benchmarking metric, which calculates displacement using a different metric, RoI discovery. RoI discovery is a useful metric to evaluate trajectory simplification because it identifies whether or not the simplification preserves the underlying visitations originally made by the entity. Preserving underlying meaning during simplification is essential for some tasks, such as data mining. For example, in data mining, simplification is a common preprocessor, but, if the underlying meaning of a trajectory is changed during simplification, the mining result may be incorrect.

The specific RoI discovery algorithm I choose is one of my previous works (Bermingham and Lee, 2014). The first stage of this approach is to define a study region, G, which is a three dimensional space that contains the normalised trajectory, $T = E_1, \ldots, E_n$. G is divided into a user-specified number of partitions in each dimension. The number of partitions in each dimension is controlled by the parameters $j, k,$ and l. The result is a grid of cells, A, which I describe using matrix notation, like so, $A^{j \times k \times l}$. I define C as a cell in A, where I refer to each as C_i when dealing sequentially with cells, and also $C_{j,k,l}$ when accessing them using the index of each dimension. Note, that each cell's visitation count, $C.V = 0$ when it is initialised. The next step in the algorithm is to trace the trajectory, T, through A.

I define $\vec{E_{a,a+1}}$ as the line segment formed by each pair of subsequent entries in T. Each cell, C_i, that contains or intersects $\vec{E_{a,a+1}}$ has its value incremented by one, $C_i.V + 1$. Note, a cell cannot have its value incremented more than once until the trajectory is traced through a different cell (this prevents saturation, but preserves visitation). Once the whole trajectory has been processed, all cells are put into a list, L, and sorted in descending order of their values.

Now the algorithm begins forming a set of RoIs, $R = \{O_1, \ldots, O_n\}$, where $O_i = \{C_1, \ldots, C_n\}$. Note, that all cells are checked and added to RoIs if they meet the *candidate criteria*, that is, $C \notin O \wedge C.V \geq D_{min}$, where D_{min} is a user-specified minimum value, a cell must have to be considered part of a RoI. The process begins by repeatedly popping the head off L until a cell $C_{j,k,l}$, meeting the candidate criteria is found. This cell $C_{j,k,l}$ is added to an empty RoI, O_{cur}. All neighbouring cells of $C_{j,k,l}$, that is, $C_{j-1,k,l}, C_{j+1,k,l}, C_{j,k-1,l}, C_{j,k+1,l}, C_{j,k,l-1}$, and $C_{j,k,l+1}$, are also added to O_{cur} if they meet the candidate criteria. This process of adding the neighbouring cells of newly added cells continues are until there is no neighbouring cell that satisfies the candidate criteria. When O_{cur} cannot be expanded any further it is considered a finished RoI and is added to the set R. This process of RoI formulation continues until L is exhausted. It is important to note that any cell can only be a member of a single RoI.

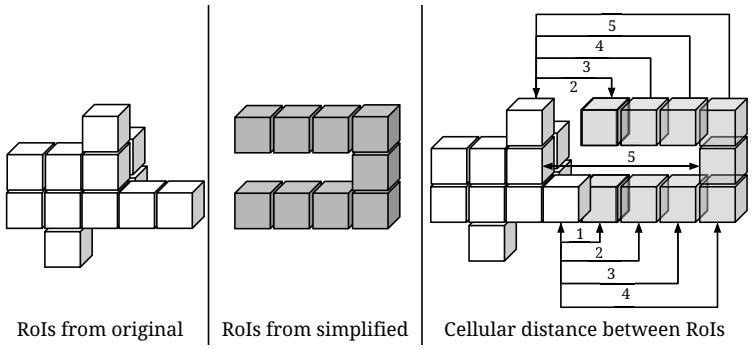


FIGURE 3.6: The cellular distances between the raw trajectory's RoI cells and the simplified trajectory's RoI cells.

The reason I use this RoI discovery approach to benchmark my simplification methods is that its results rely on preserving the underlying visitation. The simplification approaches I have created will likely remove much of the noise (and hence density) from the trajectory, but an ideal simplification would still preserve the trajectory visitations. Thus, to determine if my simplification algorithms preserve visitation, I perform RoI discovery on the both the raw, and simplified trajectories. The result is two sets of RoIs, which I use to compute the total displacement between the raw trajectory's RoIs and the simplified trajectory's RoIs.

Given two cells, C_r and C_s, recall, that C_r can also be accessed through the indices j, k, l. Additionally, to allow for a more convenient definition of distance between cells, assume C_s can be accessed through its equivalent indices a, b, c. Therefore, the cellular distance between any two cells is given in Equation 3.11.

$$D_{cell}(C_r, C_s) = |j - a| + |k - b| + |l - c|. \tag{3.11}$$

A visual illustration of measuring the cellular distance between two RoIs is given in Figure 3.6.

In order to measure the total displacement between the raw and simplified trajectories I formalise my notion of total cellular variance. Let the set of RoIs from the raw trajectory be R_r, and the set of RoIs from the simplified trajectory be R_s. Given an RoI from R_r, there is an RoI in R_s that minimises the sum of cellular distance between the cells of the two RoIs. I call such a pair of RoIs, a minimum distance RoI pair. Additionally, I call the sum of cellular distances between a minimum distance RoI pair the *minimum cellular distance*. Finally, the total cellular variance is defined as the sum of minimum cellular distances between all minimum distance RoI pairs.

3.3 Experiment Results

In order to quantitatively evaluate my simplification methods, I conducted a range of efficiency and effectiveness experiments. To measure the efficiency I recorded the running time of each simplification method, and to measure effectiveness I conducted each of benchmarks outlined in Section 3.2.7. I performed all of the experiments over a range of simplification strengths (i.e. removing $50\%, 70\%, 90\%, 95\%$ of points from the trajectories).

All experiments were performed using my various simplification methods and their 2d counterparts from the literature: SPLDP(2d) (Douglas and Peucker, 1973),

Name	Entries (millions)	Trajectories	Range (km)	Span (days)	Notes
Trucks	0.112	50	49	40	Constrained.
TDrive	0.814	1275	758	7	Constrained.
Imis	3.097	928	651	3	Unconstrained.
Synthetic	0.35-3.5	1	10	1	Generated.

TABLE 3.2: Trajectory datasets used for experiments. Note, the numbers for TDrive are for subset #13 only.

EXTA(2d) (Visvalingam and Whyatt, 1993), and GRPPA(2d) (Lee et al., 2007). In Table 3.2 I introduce the three real-world datasets I used for the experiments: Trucks (Pelekis et al., 2009), TDrive (Yuan et al., 2010; Yuan et al., 2011), and Imis (Patroumpas, 2013). Additionally, I also used a single, large, synthetic trajectory, generated to test the running time of the simplification methods at scale.

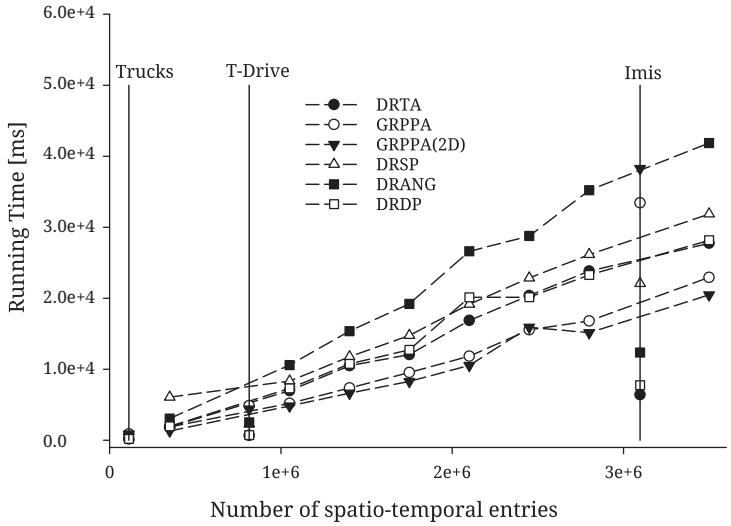
The synthetic trajectory was generated by selecting a random starting location and heading, then subsequent entries were added travelling at varying speeds along the random heading. The trajectory was confined to a certain geographic study region and, if it intersected the boundary, the heading was randomly adjusted to keep it inside the perimeter. Additionally, each entry was perturbed by an amount of spatial noise normally distributed between zero and a user-specified value (20 meters in this case).

3.3.1 Simplification Running Time Efficiency

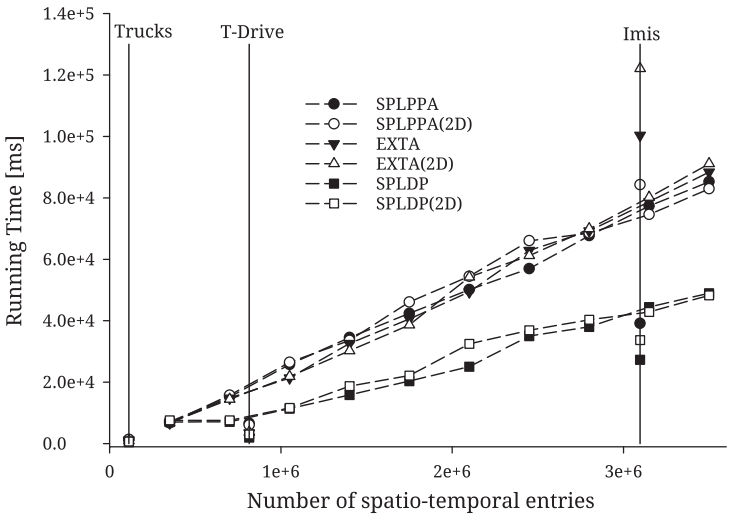
The first experiment I performed was to measure the running time of each simplification methods while I varied the dataset types and sizes. In addition to the real world datasets specified above, I also gradually simplified a single, large and noisy synthetic trajectory. I started by generating a synthetic trajectory with 350,000 entries and gradually increased its size up to 3.5 million entries. Because my test suite is written in Java, the JVM was warmed up, prior to testing each method, by running training sets. Additionally, previous results were cleared from memory to ensure garbage collection did not skew the results. In this experiment, I ran each simplification method five times, and averaged the running times to obtain the final time used in the results. The results of this experiment are shown in Figure 3.7. Please note, that Figure 3.7 depicts the running time achieved by combining various processing strategies and significance scoring functions; thus, it does not necessarily reflect the time complexity of processing strategies discussed in Section 3.2.3.

In general, I expect that the separation in running time between the simplification methodologies, shown in Figure 3.7, will become more exaggerated as dataset size increases. However, I also predict that the ordering of fast and slow simplification methodologies, shown in the results, will remain roughly the same. The reason for this prediction is that simplification running time is ultimately governed by the processing strategy used. To clarify: I expect that simplification methods using the $O(n)$ dead reckoning processing strategy will always outperform simplification methods using the more expensive split-based and exhaustive processing strategies.

I observe from the results that all the simplification approaches achieved reasonable running times on both the synthetic and real datasets sizes of 3 million



(A) Methods using DR and GR processing strategies.



(B) Methods using SPL and EX processing strategies.

FIGURE 3.7: Running time of my trajectory simplification methods when applied to multiple dataset types and sizes. The series with the dashed lines are for the growing synthetic trajectory dataset, and the vertical lines are each of the real-world datasets.

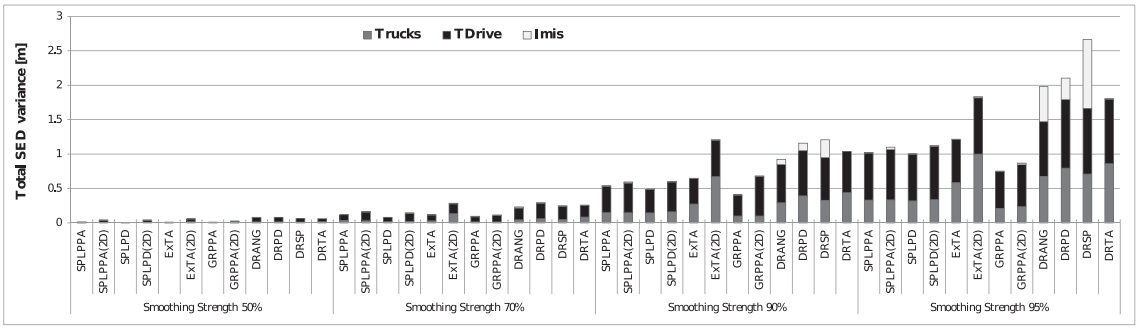


FIGURE 3.8: SED displacement of each simplification method at varying strengths and applied to multiple datasets (a lower score is better).

entries and greater. Particularly, the simplification methods DRTA and DRDP, which both achieved running times of less than 10 seconds on the Imis dataset. Another observation is that the spatio-temporal simplification methods that were extended versions of traditional approaches - SPLDP, EXTA, and GRPPA - only incurred a marginal increase in running time compared to their traditional, spatial-only, counterparts. In general, I expect the traditional 2d simplification approaches to be faster than the spatio-temporal approaches produced by my framework because they do not have to compute significance scores or rank entries. However, by gaining this running time advantage the traditional approaches lose the ability to re-simplify the trajectory at varying strengths without processing the entries again.

The final observation I make is that, as the dataset size increases, there is a reasonable time difference between some methods for the synthetic and real datasets. This is particularly evident for all the methods that use the greedy and exhaustive processing strategies, as they all run much faster on the synthetic trajectory compared to real-world Imis dataset. Investigation reveals that this result is caused by the nature of the noise in the synthetic trajectory. The trajectory is dense with noisy sections, then continues relatively unperturbed. This results in quite long runs of entries appearing in nearly sorted significance order, which results in the sorted map of $\langle entry, score \rangle$ having very low cost add operations (recall as shown in Algorithm 1 the exhaustive processing strategy builds a sorted map of entries). However, in real trajectory data, such as Imis, the opportunity for such fortunate ordering of entries is not necessarily present.

3.3.2 Geometry-based Simplification Effectiveness

The aim of this experiment was to demonstrate how effective each of the simplification methods is at minimising SED and SA displacement as the simplification strength was increased. The results of this experiment are illustrated in Figures 3.8 and 3.9.

The results indicate that simplification strengths play a strong role in SED and SA displacement. I note that the strongest performers, at the extreme simplification strength of 95%, tend to be strongest performers in that metric overall. This is the case for GRPPA in SED and DRTA in SA. This is noteworthy because both methods outperform the much more thorough simplification methods that use the split-based and exhaustive processing strategies. Another observation I highlight is that both GRPPA and DRTA do not fair as well in the other synchronised metric; GRPPA, in

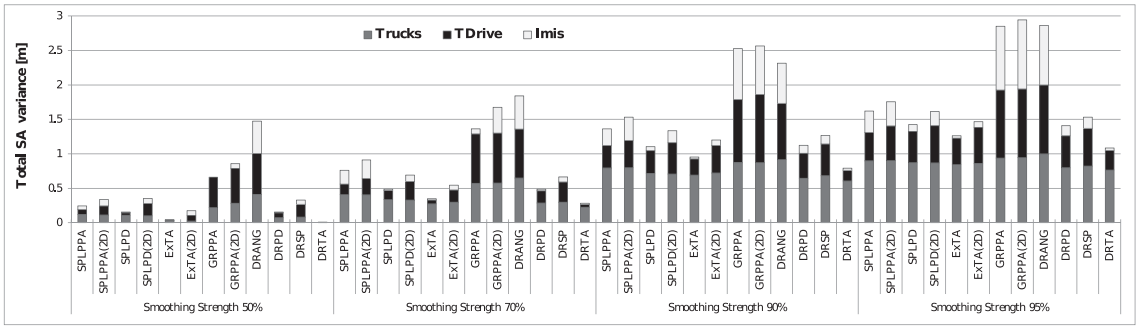


FIGURE 3.9: SA displacement of each simplification method at varying strengths and applied to multiple datasets (a lower score is better).

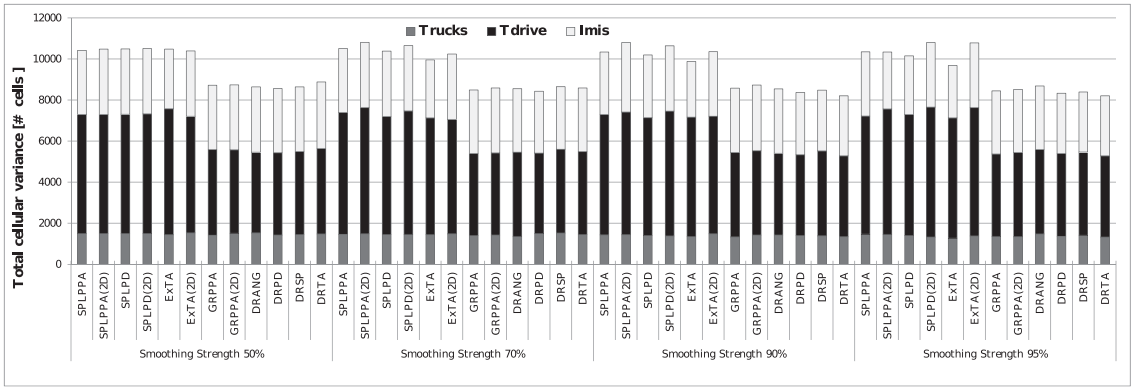


FIGURE 3.10: Total RoI displacement score for each of my trajectory simplification methods across multiple datasets (lower scores are better).

particular, scores poorly in the SA metric. I speculate this is because these methods favour a certain scoring function too heavily (i.e. perpendicular distance or area). Lastly, I observe that the spatio-temporal versions of the traditional simplification algorithms consistently achieve better SED and SA scores than their 2d counterparts.

3.3.3 RoI-Based Simplification Effectiveness

The aim of this experiment was to gauge how effective each simplification method was at preserving region visitation as trajectory simplification strength was increased. The trajectory RoI mining method I used in this experiment is one of my previous works (Bermingham and Lee, 2014), and it takes two parameters: $nCells$ and $minDensity$. The specific parameters I used for the Trucks, TDrive, and Imis datasets, respectively, are as follows $\{30, 12\}, \{20, 50\}, \{30, 50\}$. The results of this experiment are illustrated in Figure 3.10.

Figure 3.10, illustrates that compared to the other approaches, the split-based and exhaustive methods are less effective at preserving region visitation. I suggest this is because the split-based and exhaustive processing strategies enforce their geometric significance scoring more thoroughly, thus, pruning away geometrically uninteresting region visits, more often. The other observation I make is that, irrespective of simplification strength, the RoI visitation effectiveness remains approximately unchanged. This is because RoIs are formed through trajectory visitation. That is, as long as the simplified trajectory retains certain entries that

define its major movements, then removing collinear entries, and entries with small movements, will have little to no effect on the RoI cells visited in the study region. In other words, if the significant entries are preserved, then approximately the same RoI cells in the study region will be visited. Ultimately, this is a seemingly significant result because this means that even at very high simplification levels (in this case up to 95% reduction) my simplification methods managed to approximately preserve the underlying visitations of the trajectory.

3.3.4 Simplification Overall Ranking

In this section I present a summary of the overall performance of each of the simplification methods across the various experiments. To do this I used a combined score that highlights which methods performed best in terms of both efficiency and effectiveness. Specifically, I took the actual values obtained in each of the experiments and used z-score normalisation to transform the raw results of each experiment into a comparable range. Having comparable values for each experiment, a combined score was then calculated to indicate how well a particular algorithm did overall in terms of both efficiency and effectiveness. However, due to the uneven number of efficiency and effectiveness experiments, I weighted the z-scores from each experiment when calculating the combined score. The equation I used for calculating the combined score of each algorithm is shown in Equation 3.12. It is a linear combination of results from each experiment and their respective weightings.

$$S = W_t * T + W_r * R + W_e * E + W_a * A, \tag{3.12}$$

Where, S is the overall score for the simplification method; $T, R, E,$ and A are the z-scores that the simplification method received in the running time, RoI visitation, SED, and SA respectively; and $W_t, W_r, W_e,$ and W_a are weightings for each of the experiments.

The goal of the overall score is to rank each method in terms of both efficiency and effectiveness. Given that I conducted one efficiency experiment, one RoI visitation effectiveness experiment, and two geometric effectiveness experiments, I use the following weightings when calculating the combined score: $\{W_t = 0.5, W_r = 0.25, W_e = 0.125, W_a = 0.125\}$. Note, that, I allocated the same weight to efficiency (running time) and effectiveness (SED, SA, and RoI variance). Furthermore, for the effectiveness weightings, I allocated equal weightings to measuring geometric displacement (SED and SA) and topological displacement (RoI variance). The z-scores and the overall rankings for each of the simplification methods are shown in Table 3.3.

3.4 Conclusion

Overall, I have quantitatively evaluated each of my trajectory simplification methods using a variety of efficiency and effectiveness benchmarks. The results from the experiments have indicated the methods I extended from 2d, to simplify spatio-temporal trajectories, have improved both SA and SED displacement and only suffer a marginal increase in running time.

Additionally, I used my framework to combine several processing strategies and scoring functions together to create new trajectory simplification methods. Results have shown that several of the new $O(n)$ trajectory simplification methods

	Time	RoIs	SED	SA	Overall
DRTA	-0.86	-2.09	1.94	-4.00	-1.21
DRPD	-0.84	-1.50	3.15	-2.43	-0.70
GRPPA	-1.03	-0.78	-2.77	3.40	-0.63
GRPPA(2d)	-1.12	-0.89	-2.71	4.30	-0.58
ExTA	1.33	-3.29	-1.35	-1.92	-0.57
SPLPD	-0.09	0.46	-1.96	-1.23	-0.33
SPLPD(2d)	-0.12	1.33	-1.19	-1.17	-0.02
DRSP	-0.71	-0.65	4.72	0.73	0.17
SPLPPA(2d)	1.13	1.10	-1.44	-0.24	0.63
DRANG	-0.35	1.30	1.85	4.66	0.97
SPLPPA	1.22	2.90	-1.87	-0.60	1.03
ExTA(2d)	1.43	2.10	1.64	-1.52	1.25

TABLE 3.3: Trajectory simplification method z-scores and their overall weighted ranking (a lower score is better).

(particularly DRTA) have efficient running times whilst maintaining reasonable SED and SA displacement scores. Furthermore, results from the region visitation experiment indicate that my simplification methods approximately preserve the regions a trajectory visits —even after heavy simplification.

Overall, I argue that my framework is simplistic enough to be incorporated into many problem spaces requiring tunable, spatio-temporal trajectory simplification. One problem-space I plan to investigate is the incorporation of my framework as pre-processor into existing trajectory data mining approaches. In addition to the eight simplification methods I have created using my framework, I suggest my framework is a useful foundational tool-kit for others to explore new and application-specific trajectory simplification methods.

Chapter 4

Mining Distinct and Contiguous Sequential Patterns From Large Vehicle Trajectories

In this chapter I focused on the challenge of trajectory pattern complexity in network-space, and combating it through reducing the pattern output of sequential pattern mining. Specifically, I investigated reducing pattern complexity by mining a so-called distinct set of contiguous sequential patterns from highly redundant map-matched vehicle trajectories. From the review in Section 2.3 of existing sequential pattern mining approaches and relevant pattern mining approaches for vehicle trajectories, it seemed to me that there does not exist any specific approach that can mine a redundancy controlled set of patterns from large and redundant sequence databases, such as map-matched vehicle trajectories. Thus, in this chapter I introduced an approach that allows the user to specify a maximum-redundancy parameter and, then, mine a set of redundancy controlled contiguous sequential patterns. I evaluated the redundancy, compression, lossiness, and running time of my new approach and compared it to relevant closed- and max-contiguous pattern mining algorithms. Experiment results demonstrated that my proposed distinct-contiguous pattern mining algorithm is able to achieve user-controllable redundancy levels with negligible running time increases over existing approaches. Additionally, I demonstrated that due to my approach, the pattern output of the once highly redundant vehicle trajectories is now succinct enough it can visualised on a map for a user to interpret —a task that would previously have been meaningless using traditional, pattern-dense, sequential pattern mining outputs of map-matched vehicle trajectories. The topics covered in this chapter are illustrated in bold in Figure 4.1.

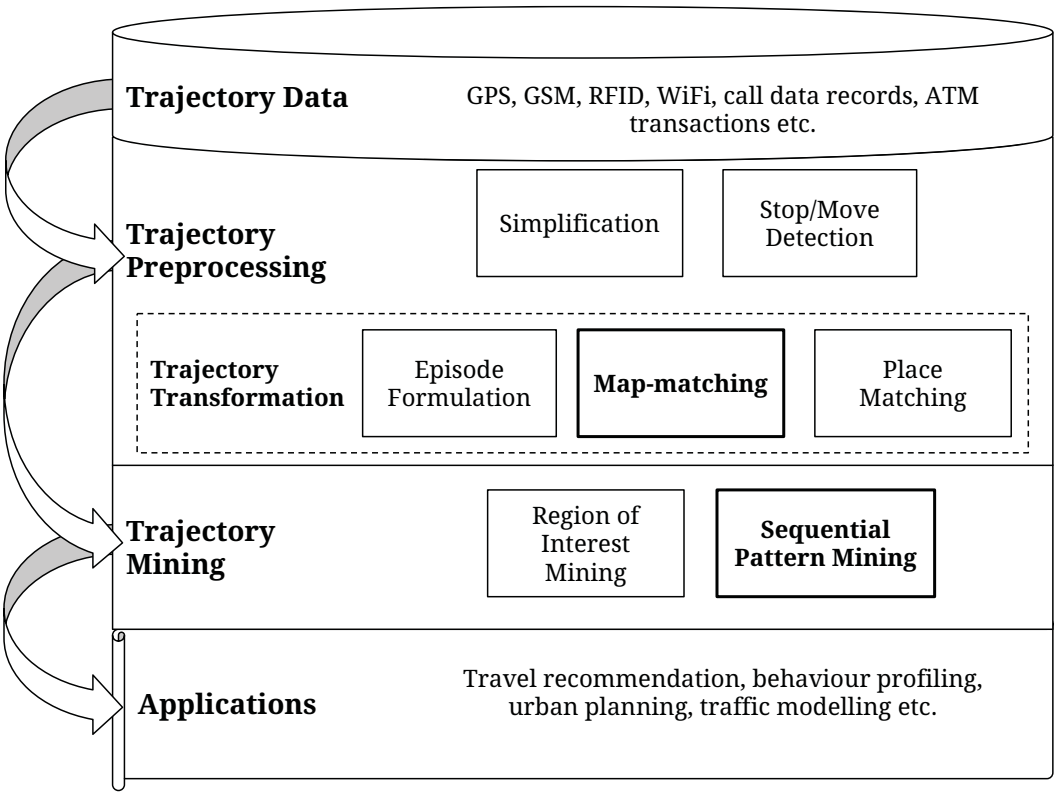


FIGURE 4.1: Overview of the trajectory data mining and knowledge discovery stages and the specific topics (the bold rectangles) covered in this chapter.

4.1 Introduction

Due to the affordability and widespread availability of GPS technology, the generation and collection of vehicle trajectories is relatively straightforward and cost effective. These vehicle trajectories present a valuable opportunity to extract knowledge in domains such as urban planning (Zheng et al., 2014b), route planning (Chen et al., 2011), and traffic congestion (Kong et al., 2016). In this chapter I focus on extracting this knowledge by mining sequential patterns from these vehicle trajectories. However, sequential pattern mining of vehicle trajectories is difficult for two reasons: (1) it requires sequences of discrete items, however, trajectory coordinates are too granular to be easily comparable; (2) vehicle trajectories commonly contain hundreds of thousands, if not millions, of recordings —which cause many existing SPM approaches to have massive, redundant and, therefore, incomprehensible pattern outputs (Atev et al., 2010; Chen et al., 2011).

The first problem can be easily solved by map-matching the vehicle trajectories to the appropriate road network as a pre-processing step (Song et al., 2014; Wang et al., 2014). Doing so removes spatial uncertainty and converts the trajectories into discrete sequences of road-node visitations suitable for sequential pattern mining. However, the second problem of mining a smaller, less redundant, set of sequential patterns from the vehicle trajectories still remains. Of course there is the so-called maximal (max) patterns (recall Section 2.1.3 Definition 13), which mine a set of patterns such that no pattern in the output is permitted to be a direct sub-pattern of another. This will reduce the size of pattern output by reducing the number of patterns; however, it will not necessarily control the redundancy (i.e partially overlapping subsequences are still permitted). Additionally, there is also the contiguous constraint (recall Section 2.1.3 Definition 14) that requires the items in the candidate patterns to exist contiguously in the underlying sequence database. Once again, this will reduce the size of the output because there will be less patterns; it is also well suited to vehicle trajectories because it guarantees that the resulting vehicle patterns will always travel along real-world routes without gaps; it will not, however, necessarily control the redundancy of the pattern output.

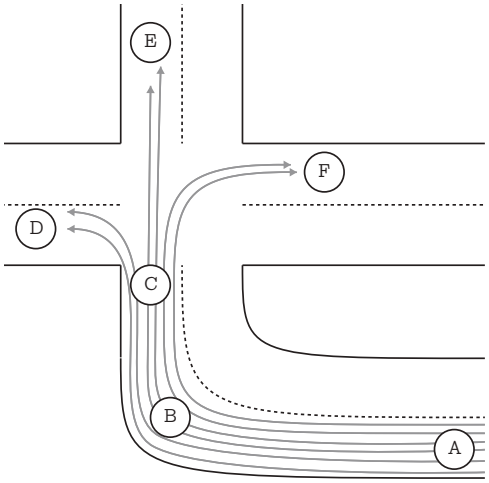


FIGURE 4.2: An example of simplified vehicle trajectories scenario.

In fact, even applying both maximal and contiguous constraints can still result in a highly redundant pattern output if the input sequences are sufficiently homogeneous. To illustrate this problem I present an example in Figure 4.2, which

is a simplified scenario containing six vehicles moving through an intersection. In Table 4.1, I present the result of mining the set contiguous max patterns from this example using a support of two (i.e $minSup = 2$).

TABLE 4.1: The maximal contiguous sequential patterns present in Figure 4.2.

Sequences	Max Contiguous Patterns
$\langle A, B, C, D\rangle$ $\langle A, B, C, D\rangle$	$\langle A, B, C, D\rangle[SUP : 2]$
$\langle A, B, C, E\rangle$ $\langle A, B, C, E\rangle$	$\langle A, B, C, E\rangle[SUP : 2]$
$\langle A, B, C, F\rangle$ $\langle A, B, C, F\rangle$	$\langle A, B, C, F\rangle[SUP : 2]$

From Table 4.1, observe that even the max contiguous sequential patterns can become quite redundant. For example, the sequence $\langle A, B, C\rangle$ is found in every discovered pattern. Specifically, this means that 75% of the pattern output is repeated redundantly. I highlight that this scenario is not an exceptional or contrived case; rather, it demonstrates an issue that is even further exacerbated when mining sequential patterns from large real-world vehicle trajectories. Recalling Section 2.3, it appears there is no existing approach that can mine a redundancy-controlled set of contiguous sequential patterns. Therefore, to solve this problem, I present the key contribution of this chapter: an algorithm to mine a set of **D**istinct **C**ontiguous **S**equential **PA**tter**N**s (DC-SPAN).

The remainder of the chapter is organised as follows: Section 4.2 introduces several formal definitions for key data-structures, and the concepts required to formally define the problem of distinct contiguous sequential pattern mining; Section 4.3 introduces the specific process of my algorithm DC-SPAN; Section 4.4 introduces the running time, compression, lossiness, and redundancy experiments, presents the results, and provides a discussion and visualisation of the experiment results; and, lastly, Section 4.5 is my conclusion, where I discuss the overall applicability of DC-SPAN to map-matched vehicle trajectories and, also, the trade-offs between redundancy and lossiness that are present.

4.2 Problem Statement

In this section I introduce some key concepts and data-structures used throughout. Then, I formally define the problem of mining distinct contiguous sequential patterns.

4.2.1 Preliminaries

All of definitions below are preliminaries to formally establish the problem of distinct pattern mining, and, also to aid in explaining DC-SPAN. Readers please note that the definitions directly extend the sequential pattern mining definitions outlined in Section 2.1.3 of Chapter 2; I encourage a recap of that section as necessary before proceeding to read the below definitions.

Definition 15 (Pair). Given a sequence $S_a = \langle a_1, a_2, \ldots, a_n\rangle$, a pair p is a tuple of any two adjacent items in S_a. That is, $p = \langle a_i, a_{i+1}\rangle$ for $1 \leq i < n$.

A pair is the next unit up from an item,; unlike an item, a pair conveys the underlying sequential nature of the data. In other words, a pair represents sequential information, whereas an item cannot, which is of importance in spatio-temporal trajectory data mining.

Definition 16 (Pair Set). Given a sequence database, $SDB = \{S_1, S_2, \ldots, S_m\}$, a pair set PS, is the set of all possible pairs that occur within it. That is, $PS = \{p \mid p$ is a pair in S_k for $\forall\ S_k \in SDB\}$.

Notations for a pair set are $PS(S)$ and $PS(SDB)$ for a sequence S and a sequence database SDB, respectively. Additionally, the number of pairs within the pair set is denoted as $|PS|$.

Definition 17 (Cover Map). Given a sequence database, SDB, a cover map, CM, is a key-value map where each key is a pair in $PS(SDB)$ and each associated value is the frequency that the pair occurs in SDB. Note, creating a cover map is denoted $CM(SDB)$.

I call the frequency associated with each pair the *cover* of the pair. This is because it represents how much of the original sequence database is covered by that particular pair. Additionally, please note that in practice this and all the other maps I define in this section are implemented as hash-maps so that they have $O(1)$ lookup and all the standard map operations, such as *get(p), contains(p), put(p,frequency), and remove(p)* for a pair p.

Definition 18 (Sequence Cover). Given a cover map, CM, and the pair set of a sequence, $PS(S) = \{p_1, p_2, \ldots, p_n\}$, the *cover* of the sequence is $\sum_{i=1}^{n} CM.get(p_i)$. Computing the cover of a sequence is denoted as $cover(S, CM)$, and for a sequential pattern that already has its cover stored, it is denoted as $cover(S)$.

By breaking a given sequence into its pair set, one can determine how much of the sequence database is covered by that sequence. The support value computed by existing approaches does not provide this kind of representative information. I provide a small example to illustrate the extra information provided by computing the sequence cover. Consider two patterns found by a traditional SPM approach, $\{a, b, c\}$ [SUP:10] and $\{b, a, c\}$ [SUP:10]. With only the support information available, one is left to assume these two patterns represent equal portions of the underlying sequence database. However, if one computes the cover for these two sequential patterns the result may become, $\{a, b, c\}$ [SUP:10 COVER:50] and $\{b, a, c\}$ [SUP:10 COVER:150]. Although, both of these patterns are found in 10 sequences, the cover reveals that the second pattern has sub-sequences that occur in three times more of the sequence database. In other words, cover provides one with a quantitative metric to judge how representative a sequence is with regard to the underlying sequence database.

Definition 19 (Sequence Map). Given a set of sequences, $\mathbb{S}$, a sequence map, SM, is a key-value map where each key is a unique id and each value is a sequence in $\mathbb{S}$. Creating a sequence map is denoted by $SM(\mathbb{S})$.

The sequence map is used for fast lookup of sequences (or sequential patterns) by their id.

Definition 20 (Pair to Sequence Ids Map). Given a sequence map, SM, a pair to sequence ids map, $P2SID$, is a key-value map where each key is a pair in

$PS(SM.values)$ and each associated value is a set of sequences ids indicating which sequences contain that pair. Notation for creating a pair to sequence id map is $P2SID(SM)$.

The pair to sequence ids map is a data-structure I use to track which pairs have appeared in the distinct sequential pattern mining output. Once a pair appears in the pattern output, it is removed from the map to indicate it has been marked as *redundant*. Controlling the pair redundancy within the pattern output is one of the unique features of DC-SPAN.

Definition 21 (**Sequence Redundancy**). Given a pair to sequence map, $P2SID$, and the pair set of a sequence, $PS(S) = \{p_1, p_2, \ldots, p_n\}$, the *redundancy* of the sequence is

$$\frac{\sum_{i=1}^{n} \begin{cases} 0, & \text{if } P2SID.contains(p_i), \\ 1, & \text{otherwise.} \end{cases}}{n}$$

Computing the redundancy of a sequence is denoted $redund(S, P2SID)$.

I highlight that the redundancy of all sub-sequences and sequential patterns found in a sequence database will initially be zero, because $P2SID$ contains every relevant pair to begin with. However, as pairs in the map are removed, sequences that contain those pairs will have their redundancy increased.

Definition 22 (**Distinct Pattern**). Given the set of all sequential patterns for a database AS, a user-specified maximum-redundancy threshold, $maxRedund$, and a pair to sequence ids map, $P2SID$, a sequential pattern S_a is *distinct* iff $S_a \in AS \wedge redund(S_a, P2SID) \leq maxRedund \wedge \nexists S_b \in AS$ such that $cover(S_b) > cover(S_a)$. Note, the set of all distinct patterns is denoted DS.

This definition implies that there can only be one distinct pattern in the set of all sequential patterns, however; as I explain in more detail in Section 4.3 once a distinct pattern has been found, it is removed from AS, and its relevant pairs are also removed from the $P2SID$. This allows one to find a set of distinct patterns each with maximal cover w.r.t the already found distinct patterns.

Additionally, by repeatedly finding the distinct pattern with the maximum cover, I produce a concise redundancy-controlled set of sequential patterns that is representative of as large a portion of the underlying sequential database as possible.

4.2.2 Problem Definition

Based on the definitions in Section 4.2.1, I now introduce the problem that I aim to solve in this chapter.

Definition 23 (**Distinct Contiguous Sequential Pattern Mining**). Given a sequence database, SDB, a user-specified minimum-support threshold, $minSup$, and a user-specified maximum-redundancy, $maxRedund$, find a set of all distinct contiguous sequential patterns.

4.3 Methodology

In Figure 4.3, I present my framework for mining distinct contiguous sequential patterns from vehicle trajectories. Briefly, each stage in Figure 4.3 of my framework is as follows:

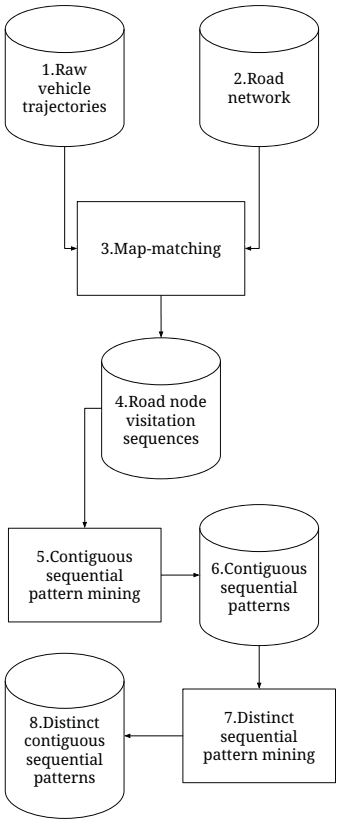


FIGURE 4.3: My framework for mining distinct contiguous sequential patterns from vehicle trajectories.

1. **Raw vehicle trajectories.** The purpose of my framework is to extract a set of patterns that represent frequent routes that vehicles have taken within their relevant road networks. The vehicle trajectories I consider in this chapter are all recorded using GPS and are stored as plain-text files as sequences of timestamped geographic coordinates. More details on the specific datasets I used are provided in Section 4.4.1.

2. **Road network.** In order to mine the vehicle trajectories using sequential pattern mining, I assume that the vehicles are constrained to travelling along road networks. I obtained the relevant road networks for each of the datasets from MapZen's Open Street Maps metro extracts repository [1].

3. **Map-matching.** It is well established that GPS recordings can be noisy and inaccurate. Thus, it is not an easy task to match each recording in the vehicle's trajectory with the correct road segment from the underlying road network. For this task of map-matching the vehicle trajectories, I used the Hidden Markov Model based approach proposed by Newson and Krumm (2009) (recall Section 2.1.2). For my purposes, I find that it yields logical matches for all of the tested datasets.

4. **Road node visitation sequences.** Using the trajectories and relevant road network as input, the map-matching produces a plain-text file containing the sequence of road nodes that each trajectory visited. This is the input sequence database that is used for sequential pattern mining.

[1] `https://mapzen.com/data/metro-extracts/`

5. **Contiguous sequential pattern mining.** The sequence database produced in the previous stage is now mined for contiguous sequential patterns. However, as discussed in Section 1.4.3 of Chapter 1 the set of patterns produced is often quite redundant and in the case of vehicle trajectories, overlapping patterns, basically, show the same section of road being visited with minor detours. To refine these redundant patterns later, I store them as a sequence database.

6. **Contiguous sequential patterns.** This is a plain-text database of contiguous sequential patterns mined from the previous stage.

7. **Distinct sequential pattern mining.** I run my algorithm on the sequence database of contiguous sequential patterns and refine it down to a set of patterns that does not surpass a user-specified maximum-redundancy parameter. This so-called distinct set of patterns is, once again, stored as a sequence database.

8. **Distinct contiguous sequential patterns.** This is the output of my algorithm, and I use it in the experiments to measure pattern output lossiness, compression, and redundancy. Additionally, this output is what I use to interpret the vehicle patterns, as it is much more succinct than the output from stage 5. Additionally, this increased succinctness allows for visualisation of the uncovered vehicle patterns, which I demonstrate in Section 4.4.6.

The main contribution of this chapter is DC-SPAN: an algorithm for discovering distinct contiguous sequential patterns. I present the details of DC-SPAN in Algorithm 2.

Firstly, I highlight that DC-SPAN does not compute contiguous sequential patterns itself and, instead, refines the output of an existing contiguous sequential pattern mining algorithm. Thus, DC-SPAN inherits the performance characteristics and bottlenecks of whichever algorithm is chosen. For example, in my implementation, the function call to '**MineACSP(**$SDB, minSup$**)**' on Line 2, is replaced with a call to a modified CC-SPAN (Zhang et al., 2015) algorithm that mines the set of all contiguous sequential patterns. My explanation of this choice, and the resulting performance characteristics, are reported in Section 4.3. I highlight that, in practice, there is no explicit need to compute the set of all contiguous patterns inside the algorithm, and if desired, these patterns can be precomputed by other means and passed in as a parameter.

After obtaining the set of all contiguous patterns, DC-SPAN initialises all the required data-structures that are used to refine the patterns down to the set of so-called *distinct* patterns (see Definition 22). Inside DC-SPAN's main loop, Line 10, the most covered pattern is found and stored in the set of distinct patterns (see Definition 18 for an explanation of sequence/pattern cover). Then, all the pairs from the distinct pattern are removed from the pair to sequence ids map, effectively marking them redundant, because they will now appear in the distinct pattern output. Finally, to ensure the user-specified maximum-redundancy is not surpassed, all patterns in the set of all contiguous sequential patterns that contain too many redundant pairs are removed. This whole process repeats, until there are no remaining patterns to refine.

Algorithm 2 DC-SPAN algorithm.

Input:
(1) SDB, a sequence database;
(2) $maxRedund$, the maximum allowed redundancy;
(3) $minSup$, the minimum allowed support;
Output: DS, the set of all distinct contiguous sequential patterns;

1: **function** DCSPAN($SDB, maxRedund, minSup$)
// Use a relevant contiguous SPM algorithm.
2: Assign AS to **MineACSP(**$SDB, minSup$**)**;
3: Assign DS to $\emptyset$;
// Make a cover map, see Definition 17.
4: Assign CM to $CM(SDB)$;
// Make a sequence map of patterns, see Definition 19.
5: Assign SM to $SM(AS)$;
// Make a pair to sequence id map, see Definition 20.
6: Assign $P2SID$ to $P2SID(SM)$;
// Find the cover of each sequence.
7: **for** (Sequential pattern S in SM) **do**
8: Assign $S.cover$ to $cover(S, CM)$;
9: **end for**
10: **while** SM is not empty **do**
// Find pattern with max cover.
11: Assign S_{max} to $\underset{\{S \in SM\}}{\operatorname{argmax}}\ cover(S)$;
12: Remove S_{max} from SM;
// Write pattern into memory or disk.
13: Save S_{max} to DS;
// Remove the relevant pairs.
14: Assign PS_{max} to $PS(S_{max})$;
15: Assign IDs to $\emptyset$;
16: **for** (Each pair p in PS_{max}) **do**
17: Add all ids from $P2SID.get(p)$ to IDs;
18: Remove p from $P2SID$;
19: **end for**
// Remove patterns that have become redundant.
20: **for** (Sequence id i in IDs) **do**
21: **if** $redund(SM.get(i), P2SID) \geq maxRedund$ **then**
22: Remove i from SM;
23: **end if**
24: **end for**
25: **end while**
26: **return** DS;
27: **end function**

4.4 Experiment Results

To gauge the efficiency and effectiveness of DC-SPAN at mining distinct-contiguous sequential patterns from large vehicle trajectories, I conducted experiments measuring running time, compression, lossiness, and redundancy. Where appropriate, I compared DC-SPAN against other contiguous sequential pattern mining algorithms that mined all-, closed-, and max-patterns. One problem I faced is that I planned to use CM-SPAM (Fournier-Viger et al., 2014a) to mine the set of all contiguous patterns, and VMSP (Fournier-Viger et al., 2014c) to mine the set of all max-contiguous patterns. However, both of these algorithms ran out of memory mining the large trajectory datasets, thus, they turned out to be unsuitable for this study. Fortunately, I was able to mine the set of all closed-contiguous sequential patterns using CC-SPAN (Zhang et al., 2015). Using CC-SPAN as a base, I slightly modified the source code to produce two algorithms: one to mine the set of all contiguous sequential patterns (AC-SPAN), and the other, to mine the set of all max-contiguous sequential patterns (MC-SPAN). Thus, for most of the experiments I compared the output produced by DC-SPAN against AC-SPAN, CC-SPAN, and MC-SPAN.

Additionally, I highlight that all of the experiments were run on a machine with an i5-520M processor and 5GB of unallocated memory. Furthermore, all of the algorithms were implemented in Java and an adequate JVM warm-up was used before all experiments.

4.4.1 Experiment Datasets

Across all of the experiments, I used the same three vehicular trajectory datasets as input. I highlight that prior to experimentation, these vehicular trajectory datasets were transformed from a series of geographic coordinates and timestamps, into sequences of visited road network nodes (see Section 4.3 for an explanation of this process). “TDrive" is the first and biggest dataset I used and is publicly available [2] from Microsoft Research Asia. The TDrive dataset contains six day’s worth of taxi trajectories in the Beijing area (Yuan et al., 2010; Yuan et al., 2011).

“Buses" is the second dataset I used, and is publicly available [3] from the Dublin city council’s “Insight" project. The Buses dataset contains approximately a month of bus trajectories moving around Dublin. The entire dataset is too large to fit into the test machine’s memory, so I used just a subset which, itself, contains 7,806 tracked buses.

“Trucks" is the last and smallest dataset I used and is a well researched trajectory dataset (Pelekis et al., 2009; Pelekis et al., 2011; Abul et al., 2008; Herrera et al., 2010; Panagiotakis et al., 2012) that is publicly available from the Chorochronos archive [4]. The trucks dataset is so-called because it contains various cement trucks making daily deliveries around the Athens region. In Table 4.2, I present some specific information about each of the datasets after they underwent map-matching.

[2] https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/

[3] https://data.dublinked.ie/dataset/dublin-bus-gps-sample-data-from-dublin-city-council-i

[4] http://chorochronos.datastories.org/

TABLE 4.2: Transformed Trajectory Datasets

Name	Sequences	Items	Avg Seq Length	Distinct Items
TDrive	7,806	5,400,239	692	50,933
Buses	782	702,384	898	14,329
Trucks	50	204,662	4,093	13,279

4.4.2 Running Time

The aim of this experiment was to quantitatively measure the efficiency of DC-SPAN against the existing all- (AC-SPAN), closed- (CC-SPAN), and max- (MC-SPAN) contiguous sequential pattern mining algorithms, when mining large real-world vehicular sequence databases. The results of this experiment are shown in Figure 4.4.

Analysing Figure 4.4, I observe that DC-SPAN has a negligible running time compared to the other approaches I measured; though, it is misleading not to restate that DC-SPAN requires the computation of the set of all contiguous sequential patterns during its routine (i.e it runs AC-SPAN internally). Therefore, the overall running time for DC-SPAN to produce a result, can be thought of as DC-SPAN + AC-SPAN. Therefore, the running time of DC-SPAN is always tied to whichever algorithm is used to mine the set of all contiguous sequential patterns (AC-SPAN in this case). Additionally, it follows that the overall running time of DC-SPAN is longer than mining the set of closed or max contiguous sequential patterns. Another observation I make is that the running time of DC-SPAN appears to be mostly insensitive to the changing support levels, which cannot be said for the other algorithms. Overall, these results suggest to me that mining the set of distinct-contiguous sequential patterns imposes little performance overhead if the set of all contiguous sequential patterns is already computed.

4.4.3 Compression

The aim of this experiment was to measure the relative compression achieved by each algorithm's pattern output. The compression value I computed is the size of the pattern output relative to the set of all contiguous sequential patterns. Specifically, I computed compression in this experiment using Equation 4.1.

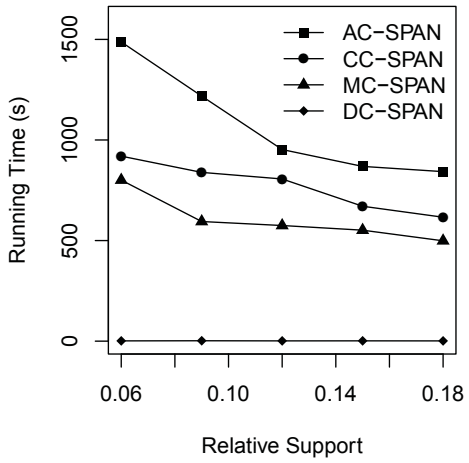
$$Compression = 1 - \frac{I_{XS}}{I_{AS}}, \tag{4.1}$$

where each term is as follows:

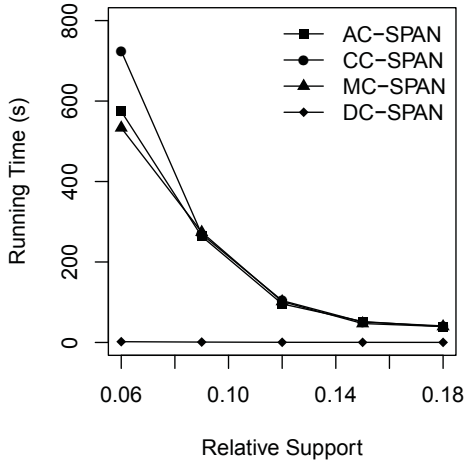
- I_{XS}: the total number of items in a given algorithm's contiguous sequential pattern output;
- I_{AS}: the total number of items in the set of all contiguous sequential patterns.

I clarify that computing the compression produced by a specific algorithm at a given minimum support requires computing the set of all contiguous sequential patterns at that same support level. With the preliminaries defined, I present the results from this experiment in Figure 4.5.

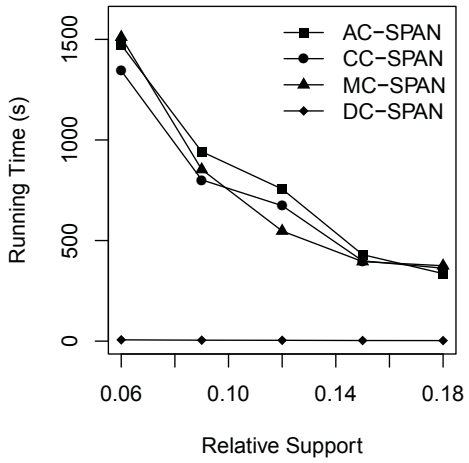
The results in Figure 4.5 indicate to me, that for all the tested datasets, DC-SPAN produces a smaller pattern output than the approaches that mined the set of closed- or max-contiguous sequential patterns. Figures 4.5b and 4.5c indicate



(A) TDrive

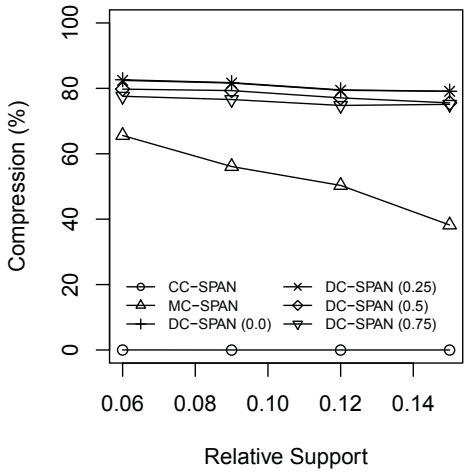


(B) Buses

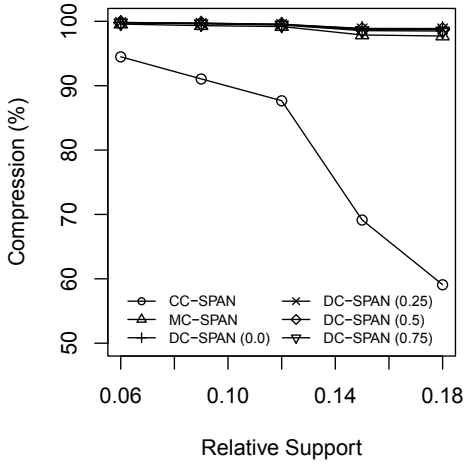


(C) Trucks

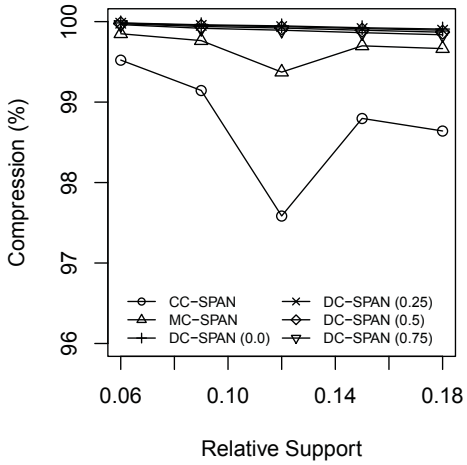
FIGURE 4.4: Running time analysis (lower is better).



(A) TDrive.



(B) Buses.



(C) Trucks.

FIGURE 4.5: The pattern output compression achieved by each algorithm (higher is better). DC-SPAN was tested at varying maximum redundancies.

DC-SPAN achieves approximately a 99% compression for the respective datasets. I highlight that for these datasets the set of all closed- and max-patterns also achieves very high compressions, around 98%-99%. This result is explained by the fact that the set of all contiguous patterns for these datasets is huge, containing many redundant sub-patterns. Thus, when these sub-patterns are removed huge portions of the patterns are compressed away. Due to the pattern output being far smaller, the results from Figure 4.5a are perhaps more telling of the overall compression abilities of each algorithm. Specifically, I highlight that in Figure 4.5a it is more clearly indicated that DC-SPAN achieves a better overall compression than the other algorithms. Additionally, I also observe that increasing the maximum-redundancy of DC-SPAN shifts its compression closer towards the set of all max-contiguous sequential patterns.

4.4.4 Lossiness

The aim of this experiment was to measure the percentage of patterns that were lost by DC-SPAN. In this experiment, I calculated the lossiness of an algorithm at a given support by counting the number of patterns, from the set of all contiguous sequential patterns, that are not contiguously contained (See Definition 14) in the given pattern output. Specifically, I calculated the lossiness of a given pattern output by using Equation 4.2.

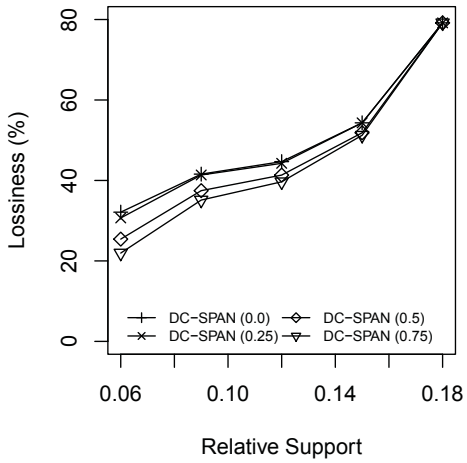
$$Lossiness = 1 - \frac{\sum_{S \in AS} \begin{cases} 1, & \text{if } S \sqsubseteq XS, \\ 0, & \text{otherwise.} \end{cases}}{|AS|} \tag{4.2}$$

where each term is as follows:

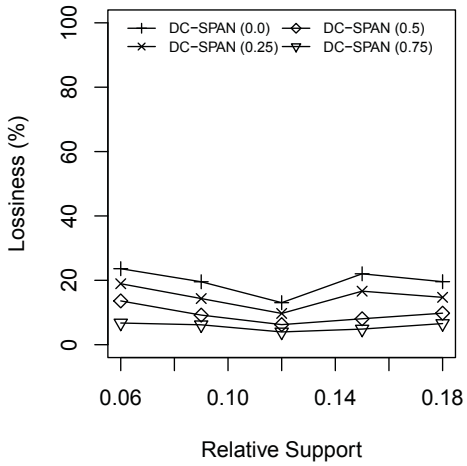
- S is $\in SP$
- XS: the set of contiguous sequential patterns produced by a given algorithm
- AS: the set of all contiguous sequential patterns.

I highlight that this definition for lossiness gives all-, closed-, and max-contiguous sequential pattern mining algorithms a perfect lossiness of zero, because by their very definitions (See Definitions 12 and 13), every pattern from the set of all contiguous sequential patterns will be contiguously *contained* by some pattern in their output. This is not the case for DC-SPAN, which discards sequential patterns based on their redundancy. Therefore, in this experiment I do not test AC-SPAN, CC-SPAN, or MC-SPAN, but instead, test DC-SPAN at varying maximum redundancy levels of 0%, 25%, 50%, and 75%. The results of this experiment are provided in Figure 4.6.

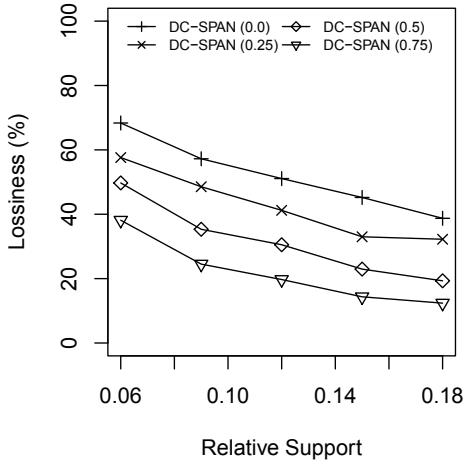
Figure 4.6 shows some varied results in terms of the lossiness DC-SPAN achieves for each of the three datasets. Specifically, for the Tdrive dataset, the output gets more lossy as support is increased, whilst for the Buses dataset the lossiness is mostly steady, and finally for the Trucks dataset the lossiness gradually declines as the support is increased. My investigation into these results reveals that for each case, the result is explained by the homogeneity or heterogeneity of the pattern output, produced by mining the set of all contiguous sequential patterns. In this context, I say that the pattern output is homogeneous if the sequential patterns discovered have a high number of pairs shared among them; heterogeneity occurs in the opposite case.



(A) TDrive.



(B) Buses.



(C) Trucks.

FIGURE 4.6: The percentage of all sequential patterns lost by each algorithm (lower is better). DC-SPAN was tested at varying maximum redundancies.

In the results for the TDrive dataset, the set of all contiguous patterns becomes increasingly homogeneous because the total number of patterns discovered is so small. At a relative support of 0.18 (i.e minimum absolute support of 1,405 sequences), only 48 sequences are found, with 4 pairs among them. Many of the sequences consist only of single items and are therefore discarded by DC-SPAN, as they have a pair cover of zero. DC-SPAN is designed for mining long sequences and setting the support close to its maximum, for the dataset will produce small patterns that it readily discards.

For the Trucks dataset, the set of all contiguous sequential patterns is very homogeneous. For example, at a relative support of 0.06, the output contains 1,008,755 sequences, with a total of 58,003,641 items. This indicates that a huge number of sub-patterns makes up the pattern output, meaning it is very homogeneous. With so many patterns sharing pairs, huge chunks of the output become redundant as the distinct patterns are mined. Thus, in this case, the main reason lossiness decreases, as support is raised, is simply because the set of all sequential patterns is pruned and becomes more heterogeneous.

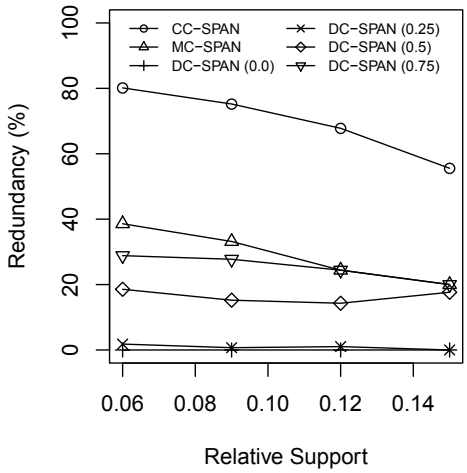
For the Buses dataset, the lossiness scores are fairly consistent across varying support levels. My investigation into the results revealed that the pattern output for the Buses dataset is the most heterogeneous, of the three datasets tested. Specifically, I found that the number of distinct pairs in the pattern output was very close to the total number of pairs. This means that most of the patterns in the output already have quite a low number of shared pairs and, therefore, mining the set of distinct contiguous patterns causes less patterns to become redundant.

From Figure 4.6 I observe, that for all of the datasets I tested, an increase in the maximum redundancy parameter correlates with a decrease in lossiness. In other words, the higher the maximum redundancy parameter, the closer the output becomes to the set of all max contiguous patterns.

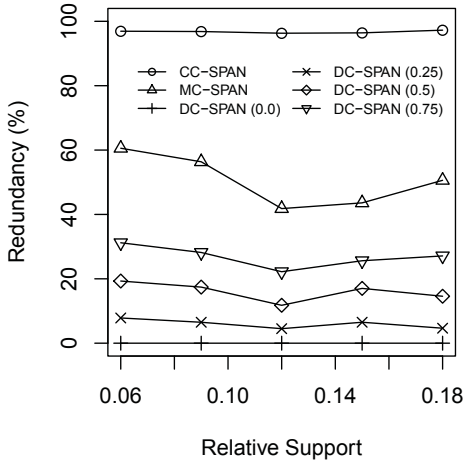
4.4.5 Redundancy

The aim of this experiment was to measure the number of redundant pairs in the pattern output of each algorithm tested. I ask readers to refer to Definition 20 and Definition 21 for an explanation of pair redundancy. The redundancy score I computed in this experiment is the number of redundant pairs divided by the total number of pairs, giving a percentage to describe the overall pattern output redundancy. The results of this experiment are provided in Figure 4.7.

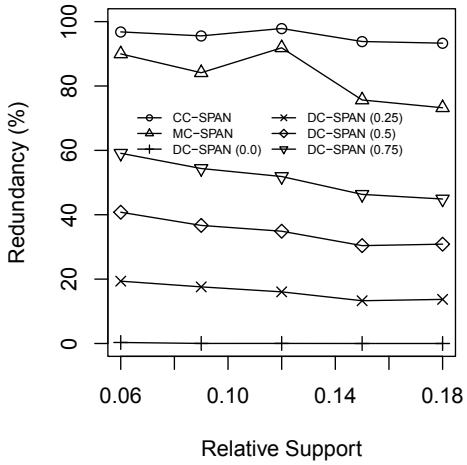
From Figure 4.7, I observe that DC-SPAN achieves its intended purpose of controlling the redundancy of the pattern output. Specifically, when DC-SPAN's maximum redundancy is set to 0%, a redundancy of 0% is obtained, and as the maximum redundancy is increased, the pattern output, as expected, becomes more redundant. I highlight that when DC-SPAN's maximum redundancy was set to 0%, the redundancy of the pattern output was approximately 30 to 60% less redundant compared to MC-SPAN, and appropriately 60 to 95% less redundant compared to CC-SPAN, at the same support levels. Additionally, even at the highest maximum redundancy tested, of 75%, DC-SPAN achieves an overall redundancy that is at least equal to, if not substantially lower than, both the closed- and max-pattern outputs across all datasets and support levels.



(A) TDrive.



(B) Buses.



(C) Trucks.

FIGURE 4.7: The percentage of redundant pairs produced by each algorithm (lower is better). DC-SPAN was tested at varying maximum redundancies.

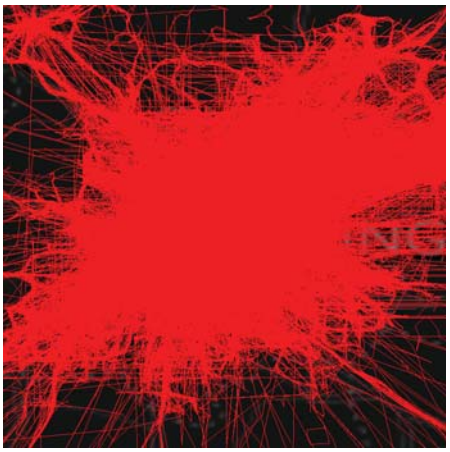
4.4.6 Visualisation

Large vehicle trajectory datasets are often huge and repetitive, making them difficult to visually inspect, but ideal to mine. In my experience, using traditional sequential pattern mining approaches on vehicle trajectory datasets often produces pattern outputs that are still too dense to visually interpret because of the large number of repeated and redundant patterns. However, by using DC-SPAN the redundancy can be controlled thus, meaning, visualisation now becomes a relevant process for interpreting the pattern output. In Figure 4.8 I present visualisations for each of the raw trajectory datasets I used in the experiments, and an accompanying pattern output mined by DC-SPAN.

Noisiness of GPS technology is a well known difficulty and is frequent in these vehicular datasets, particularly the TDrive and Buses datasets (see Figure 4.8a and 4.8c). Despite this, my combination of map-matching and distinct contiguous sequential pattern mining has uncovered a succinct and clean set of sequential patterns which map very accurately to the underlying road network topology. Analysis of the results in Figure 4.8 reveals that many of the sequential patterns found are highways and major roadways, which hints at the validity of the uncovered patterns. To the best of my knowledge, this combination of map-matching and contiguous sequential pattern mining has not been used in this way before. I speculate that, previously, sequential pattern mining output was too large and redundant to visualise meaningfully.

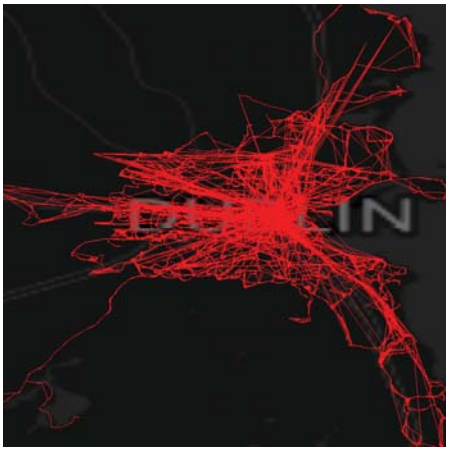
4.5 Conclusion

Although there exist many efficient and effective sequential pattern mining algorithms, none of them is particularly well suited for mining large vehicle trajectories where the pattern should be, ideally, contiguous *and* redundancy-controlled. In this work, I have presented my approach, DC-SPAN, to solve this problem. Through quantitative experiments I have shown that DC-SPAN is able to mine distinct, redundancy-controllable, contiguous sequential patterns from large and varied real-world vehicle trajectories, with very little additional overhead, compared to existing approaches. Additionally, the experiment results also revealed that the set of distinct patterns mined by DC-SPAN is more succinct than traditional approaches, but, at the cost of pattern lossiness. Specifically, the experiment results indicated there exists a trade-off between increased redundancy and decreased compression and lossiness. The more skewed this trade-off is towards decreased compression and lossiness, the more DC-SPAN resembles a max-contiguous sequential pattern mining algorithm. Overall, I conclude that the main usefulness of DC-SPAN is the tunable redundancy of the pattern output and, therefore, in cases like vehicular datasets, where a redundancy-controlled pattern output makes sense, DC-SPAN is an effective solution.

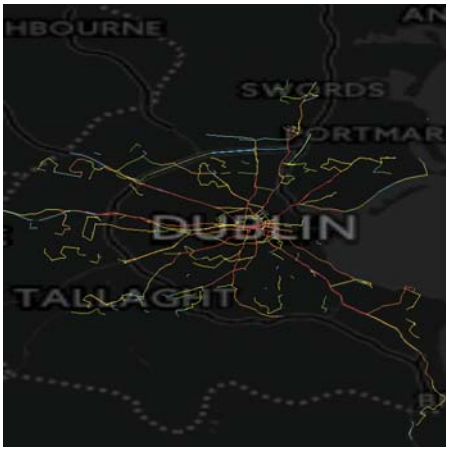


(A) TDrive trajectories.

(B) TDrive distinct patterns (MinSup = 0.185, MaxRedund = 0).



(C) Buses trajectories.



(D) Buses distinct patterns (MinSup = 0.025, MaxRedund = 0).

(E) Trucks trajectories.

(F) Trucks distinct patterns (MinSup = 0.2, MaxRedund = 0).

FIGURE 4.8: The raw trajectory datasets (left) and the distinct contiguous patterns mined from them (right).

Chapter 5

A Probabilistic Stop and Move Classifier for Noisy GPS Trajectories

In this chapter I focused on addressing the challenges of spatial uncertainty and adding semantic meaning to free-space trajectories. Specifically, I investigated semantically enriching free-space trajectory entries by classifying them as either stopping or moving. However, from my review of existing free-space stop/move detection approaches in Section 2.4, it seemed to me there is no existing approach that can detect stops/moves and attach a degree of confidence to the classification (i.e. the classification may be relatively uncertain if many of the points were spatially noisy). Additionally, there does not appear to be any existing approach that can perform its stop/move detection in a unsupervised way, and estimate all its own parameters. Thus, in this chapter, I introduced an unsupervised, probabilistic, stop/move classification algorithm that attaches a probability to each classification and allows the user to easily filter out stop and move classifications that are not probable enough for their use-case. Additionally, I also provided estimation schemes for all of my algorithm's parameters, provided source code for my algorithm, and supplied several real-world, ground-truth, stop/move annotated trajectories for evaluation of classification effectiveness.

In the experiments I used these ground-truth trajectories alongside some synthetic trajectories to compare the classification effectiveness, parameter sensitivity, and running time of my approach to two well-known existing approaches: SMoT and CB-SMoT. Experiment results demonstrated the efficiency, effectiveness, sampling-rate robustness, and spatial parameter insensitivity of my approach compared to the existing approaches. Specifically, I conducted experiments using ground-truth data and was able to demonstrate that a user tweaking my algorithm can increase the minimum stop probability parameter to reduce the number of false-positive stop classifications. Furthermore, the results demonstrated that even when all of my algorithm's parameters were estimated, the classification effectiveness of my algorithm was higher than existing approaches across a range of sampling-rates. The topics covered in this chapter are illustrated in bold in Figure 5.1.

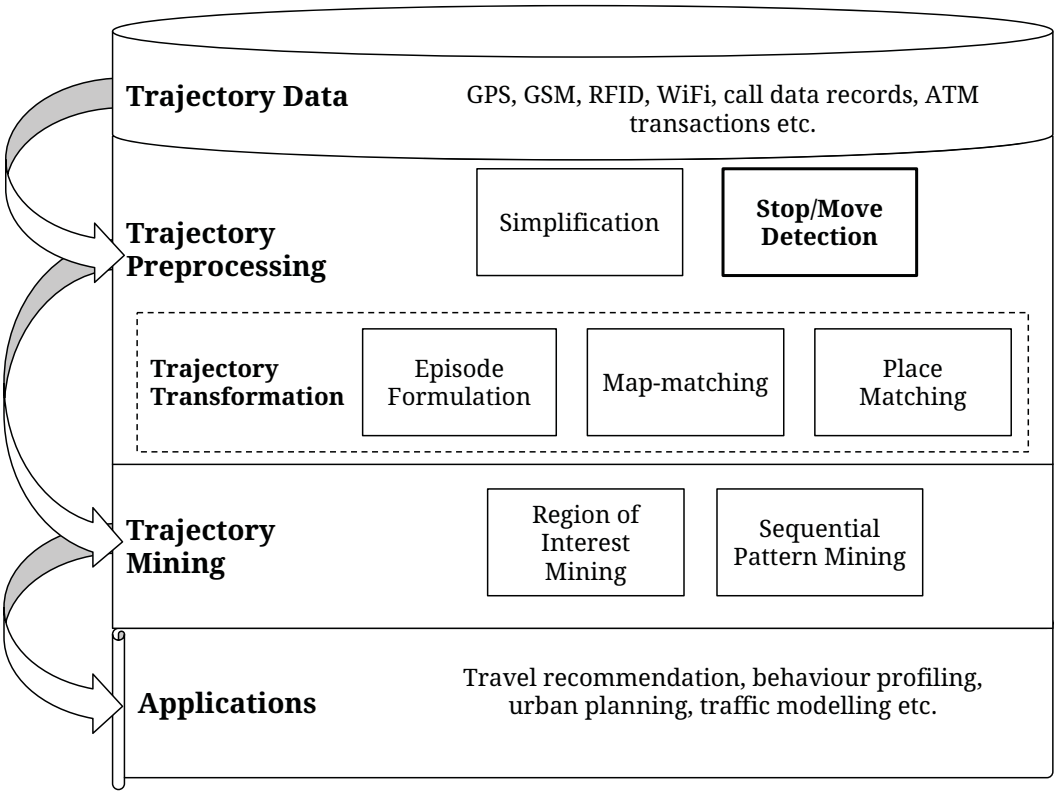


FIGURE 5.1: Overview of the trajectory data mining and knowledge discovery stages and the specific topics (the bold rectangles) covered in this chapter.

As a full disclosure readers should be aware that the work in this chapter went on to be accepted as: Bermingham, L. and Lee, I. (2017). A Probabilistic Stop and Move Classifier for Noisy GPS Trajectories. *Data Mining and Knowledge Discovery*.

5.1 Introduction

Without semantic enrichment, raw trajectories lack the necessary contextual information required to infer useful higher-level knowledge such as: what the entity was doing; what kind of place the entity was travelling to; or, what activity the entity might do next. A common technique, to add some context to these raw trajectories, is to semantically enrich them with an additional extra dimension describing if the entity is stopping or moving (Alvares et al., 2007; Gong et al., 2015; Tran et al., 2011; Xiang et al., 2016). The assumption is that when an entity stops it is performing some activity (Xiang et al., 2016), such as a person eating lunch or a bird roosting. The raw stops and moves are not inherently contextual, but when paired with additional contextual data sources, such as the underlying geography, they can become meaningful. Thus, semantically enriching a GPS trajectory with stops and moves is an effective way to add context. Though, it has become a fundamental technique in many semantic knowledge discovery approaches (Gong et al., 2015; Tran et al., 2011), the problem remains in accurately and efficiently obtaining these stops and moves. One approach is getting the entity to keep a travel diary, but this is time-consuming and not applicable to non-human entities. Another approach is using a fusion of device sensors (i.e accelerometer, gyroscope, camera, and GPS) to detect moves and stops; however, for long-term installations, such as animal tracking, this extra battery drain is not feasible. Therefore, the most commonly used approach is to use the data that is already available, the spatio-temporal recordings, and infer, from them, whether the entity is stopping or moving. The advantage of this approach is that it is applicable to all spatio-temporal GPS trajectories and does not require any special external data-sources or data pre-processing steps.

It is not necessarily straightforward to detect stops and moves using only the spatio-temporal recordings. The main challenge faced is due to the noisiness, uncertainties, and unreliability of the recorded locations. Even standing still under clear skies, the recorded location will vary somewhat, and the problem is only worsened when large buildings, weather conditions, and other various sources of interference are introduced (Trajcevski, 2011). Thus, the task of separating stops from moves, using only noisy GPS recordings as input, is a challenging task, which I call stop/move detection. Recalling Section 2.4, I identified the following gaps present in the state-of-the-art stop/move approaches that motivate this chapter:

1. It is difficult to objectively compare the reviewed stop/move algorithms, in terms of effectiveness, because there is no public ground-truth dataset available to the field. This difficulty in comparing approaches is further exacerbated given that very few approaches share their source code.

2. In the case of unsupervised stop/move detection algorithms, all those I reviewed require at least one, if not multiple, user-specified parameters. Thus, the user is either required to have detailed knowledge of the dataset/algorithm or blindly conduct trial and error to select algorithm specific parameters.

3. All reviewed approaches make no consideration in their results that some stop/move classifications may be more uncertain than others (i.e due to spatial uncertainty and recording errors). That is, all entries are classified as either definitely-stops or definitely-moves, with no indication in the result that some classifications can be made with more certainty than others.

These literature gaps motivates me to develop an unsupervised stop/move classification algorithm with the following design considerations: (1) accounts for spatial uncertainty and error in the result by calculating the probability of a stop classification at a given entry; (2) has estimation schemes for all user-specified parameters, that the user can use if desired; and, (3) provides some means for the user to filter the stop/move classifications that are of an unacceptable probability for their use-case. Summarising, I propose an algorithm that calculates the stop probability of each entry in the trajectory, and, then, lets the user decide on some threshold stop probability that is acceptable for their purposes. Entries with stop probabilities below the threshold are classified as moves, while those equal-to or above the threshold are classified as stops. Additionally, I also present estimation schemes for all my algorithm's user-specified parameters.

The contributions of this work as follows:

1. A novel stop/move classifier that calculates the probability of each entry being a stop. This probabilistic approach to stop/move classification allows the user to set a threshold probability for stopping entries; thus, giving the user direct control over filtering out classified stops that are of an unacceptable likelihood for the user's purpose;

2. Heuristic schemes for estimating the parameters of my stop/move classifier. Additionally, experiments testing the parameter sensitivity of my classifier and the effectiveness of the parameter estimation heuristics;

3. Experiments that quantitatively compare the running time and classification effectiveness of my approach to the popular SMoT (Alvares et al., 2007) and CB-SMoT (Palma et al., 2008) approaches; and

4. A means for others to quantitatively compare algorithm effectiveness by sharing my real-world, ground-truth, stop/move annotated trajectory datasets; my data collection application; my synthetic data generation source code; and my algorithm source code.

The subsequent sections in this chapter are as follows: in Section 5.2 I present my stop/move classification algorithm, then, in Section 5.3 I present experiment results that quantitatively compare the efficiency and effectiveness of my stop/move classifier against two existing approaches. Lastly, in Section 5.4, I provide some discussion and concluding remarks about my approach and the field, overall.

5.2 My Stop/Move Classification Approach

In this section I present the details of my stop/move classification methodology which, specifically, includes: my stop/move classification algorithm, parameter explanations, and parameter estimation approaches. Before defining my algorithm's pseudo-code I must first formally define the concept a stop/move annotated trajectory, which is the data-structure I aim to produce from my stop/move classification. Thus, I present Definition 24 below.

Definition 24. A stop/move annotated trajectory, T_{sm}, is a list of spatio-temporal, stop/move annotated entries, $(\langle x_1, y_1, t_1, a_1\rangle, \langle x_2, y_2, t_2, a_2\rangle, \ldots, \langle x_n, y_n, t_n, a_n\rangle)$, where $x_i, y_i \in \boldsymbol{R^2}$, $t_i \in \boldsymbol{R^+}$, and $a_i \in \{STOP, MOVE\}$, for $i = \{1, 2, \ldots, n\}$ and $t_1 < t_2 < \ldots < t_n$.

5.2.1 POSMIT Algorithm

Unlike many of the reviewed stop/move approaches that use density, or time spent in geographic regions, to definitively label each entry in the trajectory as either a stop or a move, my algorithm, "Probability of Stops and Moves in Trajectories" (POSMIT), calculates the probability that a given entry is truly stopping. Given the stop probability of each entry, the user then specifies a minimum stop probability parameter, ϵ, and, using this parameter all entries with stop probabilities equal-to or above ϵ are classified as stops; all other entries are classified as moves. This parameter directly gives the user control over how strict they wish to be during stop classification, or, in other words, it allows the user to filter out entries whose classification labels are too ambiguous for the application. Filtering ambiguous classifications is particularly useful in domains such as data mining, where misclassified stops lead to false patterns. I highlight that the ability for the user to filter out ambiguous stops is not a possibility directly afforded by existing stop/move detection algorithms. In general, existing stop/move detection algorithms offer no indication in the result regarding the goodness-of-classification for produced stop/move labels. This means, that without a ground-truth, the user is left to manually inspect the validity of produced stops/moves, for example, visually, overlaying them on a map and analysing them by eye. This is both time-consuming and highly subjective. Thus, I argue that some indication of the goodness-of-classification in the result would be most useful, but is unfortunately lacking in the reviewed approaches.

Algorithm 3 POSMIT algorithm.

Input:
(1) T, a trajectory of entries $\langle x_i, y_i, t_i \rangle$.
(2) h_i, index search bandwidth.
(3) h_d, stop variance.
(4) ϵ, minimum stop probability.
Output: T_{sm}, a trajectory of entries $\langle x_i, y_i, t_i, a_i \rangle$.

1: **function** POSMIT(T, h_i, h_d, ϵ)
$T_{sm} = ()$
// Find stop probability of each entry.
2: **for** $(i = 0; j < T.length; i{+}{+})$ **do**
3: $a_i = MOVE$
4: **if** CALCSTOPPR(T, i, h_i, h_d) $\geq \epsilon$ **then**
5: $a_i = STOP$
6: **end if**
7: Add entry $\langle x_i, y_i, t_i, a_i \rangle$ to T_{sm}
8: **end for**
9: **return** T_{sm}
10: **end function**

Thus, I now present POSMIT in Algorithm 3 and also highlight that POSMIT's Java source code is publicly available for interested readers [1]. Briefly, the algorithm iterates each entry and calculates the stop probability. How this calculation is made is explained in Section 5.2.2.

[1] `https://github.com/lukehb/137-stopmove`

5.2.2 Stop Probabilities

In order to calculate an entry's stop probability, POSMIT samples the spatial displacement of surrounding entries. An entry with a high stop probability will have surrounding entries that don't move very far (low spatial displacement). Many approaches from the literature also define a stop as occurring for some minimum duration (Alvares et al., 2007; Tran et al., 2011; Gong et al., 2015; Xiang et al., 2016). POSMIT does not make such assumptions when classifying stop/moves and can therefore find very granular stops.[2]

In Algorithm 3, line 4, *CalcStopPr*(T, i, h_i, h_d) is called to calculate the stop probability of the entry at index i. This is equivalent to the formal definition, $Pr(Stop|x_i, y_i)$ (based on the Gaussian kernel smoothing function (Hastie et al., 2001; Nadaraya, 1964)), which I present in Equation 5.1.

$$Pr(Stop|x_i, y_i) = \frac{\sum_{j=l}^{u}\{K(\omega_{i,j})K(\Delta_{i,j})\}}{\sum_{j=l}^{u} K(\Delta_{i,j})}. \tag{5.1}$$

Each term in Equation 5.1 is as follows:

- $K(z)$, a Gaussian function, see Equation 5.2.
- $\Delta_{i,j}$, the normalised index displacement between the entries i and j, see Equation 5.3.
- l, u, the lower and upper bounds of the search window, Equations 5.4 and 5.5.
- $\omega_{i,j}$, the normalised spatial displacement between the coordinates of entries i and j, see Equation 5.6.

The function $K(z)$, shown in Equation 5.2, is a kernel function (recall POSMIT is based on kernel smoothing). It is a Gaussian function with a mean of zero, a height of one, and a standard deviation of one. It serves two purposes: firstly, it provides a mapping, $z \hookrightarrow [0, 1]$, meaning the result is in the range $[0, 1]$; secondly, because it is a zero-mean Gaussian function, values decay as they move further away from zero. This is ideal for my purposes, because a small displacement between entries will produce a high stop probability, whilst a large displacement will produce a low stop probability.

$$K(z) = e^{-0.5z^2}. \tag{5.2}$$

The normalised index displacement, shown in Equation 5.3, ensures that the closer entries are to i, the more they contribute to the stop probability calculation (similar to the distance decay effect in spatial analysis (M. J. Smith, 2015)). Therefore, entries that are many indices away from i have a negligible effect on its stop probability calculation. It is important to understand that because $\Delta_{i,j}$ is passed into the Gaussian function $K(\Delta_{i,j})$ the h_i parameter effectively controls the width of that Gaussian function, much like a bandwidth parameter in Gaussian kernel density estimation. In the context of POSMIT, this means that h_i directly controls the weighting of surrounding entries that are sampled during an entry's stop probability calculation, which, as readers will see in Equations 5.4 and 5.5,

[2] In the case where stops must occur for some minimum amount of time it is straightforward to enforce this constraint on POSMIT's stop/move classification result. Firstly, all contiguous entries that are classified as stops are merged into groups, each of these groups then has their combined durations calculated, and finally groups whose durations are too low become moves.

decides how large the sampling window is when performing stop probability calculation.

$$\Delta_{i,j} = \frac{|i-j|}{h_i}. \tag{5.3}$$

The lower and upper index search-bounds, l and u (respectively), define the sampling window of indices that are considered during the stop probability calculation of the entry at index i. Calculation of these bounds is shown in Equations 5.4 and 5.5. The concept behind these bounds is to ensure that when the normalised index displacement, $\Delta_{i,j}$, reaches a small enough value its contribution to the stop probability calculation is disregarded. Given the Gaussian function $K()$ has a mean of zero and a height of one, the value where I deem the contribution low enough is $K(\pm 3) \approx 0.01$. This choice is reflected in Equations 5.4 and 5.5, where the sampling window indices are defined. The reason h_i is used as multiplier in those equations is that, as mentioned above, it effectively controls the width of Gaussian $K(\Delta_{i,j})$ that determines neighbouring entry weights. Note, I could set the lower and upper bounds to the start and end of the trajectory for a comprehensive search; however, this would result in POSMIT having a quadratic running time. The fact I wish to avoid this should highlight that h_i largely dictates POSMIT's running time by controlling the size of the sampling window for each stop probability calculation. Additionally, note that in practice, the lower and upper bounds are clamped between 0 and n to avoid index out of bounds issues (where n is the maximum index in the trajectory).

$$l = max(0, i - 3 * h_i). \tag{5.4}$$

$$u = min(n, i + 3 * h_i). \tag{5.5}$$

The normalised spatial displacement, $\omega_{i,j}$ (Equation 5.6), is the Euclidean distance[3] between the coordinates of entry i and its nearby entry j, divided by the user specified stop variance parameter, h_d. Much like h_i, it is a bandwidth parameter to control the width the Gaussian function $K(\omega_{i,j})$. Equation 5.6 is the key step in the stop probability calculation because it spatially defines the notion of a stop. That is, if the nearby entries, j, are spatially close to the entry i, then i is more likely to be a stop. Note, Equation 5.2, Equation 5.3, and Equation 5.6 model spatial dependence and spatial autocorrelation that are consistent with the Tobler's First Law of Geography (TFLG): "[e]verything is related to everything else, but near things are more related than distant things" (Haining, 2003; Tobler, 1970). Note, Equation 5.1 is based on the Gaussian kernel smoothing function (Hastie et al., 2001; Nadaraya, 1964), and Equation 5.3 reflects the distance decay effect in spatial analysis (M. J. Smith, 2015); thus, Equation 5.1 and Equation 5.3 are coherent with the TFLG (M. M. Fischer, 2010).

$$\omega_{i,j} = \frac{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{h_d}. \tag{5.6}$$

[3]Entries with spatial coordinates in a non-Cartesian geographic projection will need to be unprojected to calculate a suitable Euclidean distance. Also, Euclidean distance was chosen over great-circle distance for this problem because it is most widely used in spatial analysis (M. J. Smith, 2015), and it is faster to compute and intra-point distance between points in a candidate stop are intrinsically small; thus, factoring in the curvature of Earth in this case would be negligible.

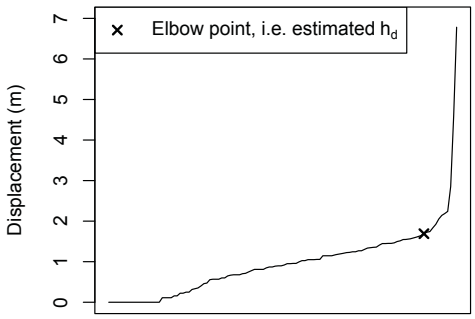


FIGURE 5.2: Spatial displacement between trajectory entries and the resulting elbow point for a trajectory of a short stop/move walk (x: trajectory nodes; y displacement (m)).

If GPS noise was not a factor, finding stops would be trivial. It would simply be a matter of finding contiguous entries with no spatial displacement between them. However, due to the noisiness and unreliability of GPS technology in reporting the same position when a user is truly stopped, the stop variance parameter, h_d, is required so that the user can control how strict they wish to be when calculating stop probabilities. For example, a h_d value close to zero means that a high stop probability could only occur if the GPS reported nearly identical geographic positions in contiguous entries, which, in practice, is unlikely.

5.2.3 Spatial Stop Variance Parameter

Selecting the ideal h_d parameter involves analysing the noise that is present in the dataset. One approach, to isolate the noisiness of a GPS recording, is to compare the recorded positions to the true, known positions. However, the true trajectory is not always known, and for such cases I use a heuristic that calculates a starting value for the parameter —from there the user can adjust it accordingly.

In the case of estimating h_d, I want to find the amount of spatial variance (or movement) that is acceptable within a true stop, but not so large that I start classifying small moves as stops. The first step towards estimating h_d is to measure the spatial displacement between each contiguous entry in the trajectory. I then sort the spatial displacements in ascending order. Using this spatial displacement data, I assume that, if there are stops and moves within the data, there exists some point of change where the value of displacement between entries differentiates stops from moves. To find this point of change, I use the concept of finding *elbow points* within the data. To illustrate this concept of measuring displacements and identifying elbow points I present Figure 5.2.

The elbow point is the point of maximum curvature in a function. In their work on detecting elbow and knee points in data, Satopaa et al. (2011) state that for a continuous function f, the curvature is given by the closed-form, $K_f(x)$, which defines the curvature of f at any point, x, as a function of its first ($f'(x)$) and second derivative ($f''(x)$) (see Equation 5.7).

$$K_f(x) = \frac{f''(x)}{(1 + f'(x)^2)^{1.5}}. \tag{5.7}$$

In this case, the displacement data is not a continuous function, and fitting a curve to it is not an ideal solution, as the displacements can be very noisy. Therefore, I adapt the approach described by Satopaa et al. (2011) called "Kneedle". Kneedle is suitable for this case because it is able to find elbow points in discrete

and noisy datasets. It begins by smoothing the data to prune out misleading jitter that could cause a false-positive elbow point. In my implementation, this smoothing is performed using a Gaussian kernel smoother. Kneedle's next step is to normalise the data into the unit-square $[0, 1]^2$ using min-max normalisation. Then, the data is transformed into the set of differences between x and y values by performing $(x, y - x)$ on each pair of values. The purpose of this difference transformation is to highlight when the data transitions from horizontal to rapidly changing (as this is likely an elbow point). From this transformed data one can extract the elbow point as the x value with the maximum absolute y value.

One can also reduce processing time and prune out unlikely h_d values by specifying a maximum reasonable displacement cut-off. The assumption here is that some displacement values between entries are so large that one can be certain they could never be true stops; thus, they can easily be pruned away. In the experiments, I set the displacement cut-off to 20 metres, thereby, pruning all displacements above 20 metres for consideration during h_d estimation. The effectiveness of this h_d estimation is evaluated in Section 5.3.2.

5.2.4 Index Search Bandwidth Parameter

Recall, from Section 5.2.2, that the greater the number of indices between the current entry and some neighbour entry, the less weighting it has on the stop probability calculation; likewise, the smaller the number of indices away, the more weighting the entry has. In POSMIT this range of considered values is directly controlled by the index search bandwidth parameter, h_i (see Equation 5.4 and Equation 5.5). I highlight that it is by design that the sampling neighbourhood is defined by an index-based parameter, and not by a user-specified duration. Even though a temporal bandwidth parameter would be more intuitive for the user to select, it does not translate well to POSMIT. This is due to the varying sampling-rates of recorded entries caused by device errors, device battery life conservation, and GPS signal unavailability.

Consider a modification to POSMIT, where a temporal bandwidth, say, h_t, did define the sampling neighbourhood, and entry weighting was based on the time difference between subsequent entries. In some cases, this modification may perform well; but I speculate, however, that such a modification would make POSMIT largely ineffective on real-world trajectories. This is because real-world trajectories are often recorded with varying sampling-rates (as mentioned above), which would mean that sparsely sampled entries may be missed if h_t were too small. Additionally, using h_t would require the user to have some knowledge regarding the average or modal sampling-rate of the input trajectory. Comparatively, using the h_i version of POSMIT, where the sampling neighbourhood and neighbour entry weightings are based on the number of indices between entries, the issue of sampling-rate between entries is mitigated; by extension this means that the user is not required to know anything about the sampling-rate of the trajectory. Additionally, in practice, I assume trajectories are recorded as time-ordered sequences of entries: this means that the index search bandwidth, h_i, still indirectly incorporates the important temporal dimension of the data too. Lastly, I remind the reader that for applications where a minimum stop duration is required, this can be applied as a post-processing step after POSMIT has labelled each entry as a stop or a move (recall the footnote from Section 5.2.2).

The issue with h_i, is that unlike a temporal parameter the user may have no intuition about how many entries would be a reasonable amount to sample for stop

probability calculation. Thus, I propose a heuristic to estimate h_i for the user. The design behind this heuristic is to find groups of easily identifiable stops (i.e stopping entries that are not perturbed by much spatial noise) and find a sampling size that captures most of those groups. The heuristic begins by iterating the input trajectory and constructing a subsequence of contiguous entries such that each entry in the subsequence is $\leq h_d$ metres away from the previous entry. When this condition no longer holds, the size of the subsequence is stored (if it is ≥ 2) and the process begins again at the index that failed the condition. Once the whole trajectory has been processed, the stored sizes are used to calculate the median stored subsequence size. The estimated h_i value is then assigned to half the median subsequence size (half, because h_i establishes a sampling neighbourhood in both directions). The limitations of this h_i estimation heuristic are: (1) that the h_d parameter must be supplied or estimated; and (2) the input trajectory must have subsequences of entries that satisfy this simple, distance-based heuristic (which may not be the case if the whole trajectory is more spatially noisy than the specified h_d). I evaluate the effectiveness of this heuristic in Section 5.3.3.

5.2.5 Minimum Stop Probability Parameter

The minimum stop probability parameter controls which entries are classified as stops. However, without knowing the noisiness of the recorded trajectory, it can take some trial and error to determine a suitable ϵ value, that provides a reasonable separation between the stop and move classification labels. One approach is to calculate the stop probability for each entry, then visualise the result. For example, in Figure 5.3 I show the calculated stop probabilities for a short walk, whose ground-truth is: $moving(0-28s) \rightarrow stopping(29-43s) \rightarrow moving(44-75s) \rightarrow stopping(76-97s) \rightarrow moving(98-122s)$.

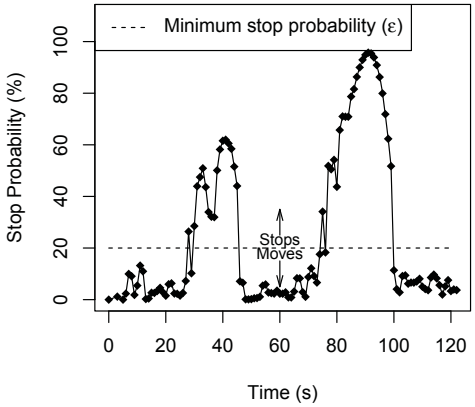


FIGURE 5.3: Stop probabilities for a trajectory of a short walk. POSMIT calculated these stop probabilities using $h_i = 4$ and $h_d = 0.6m$.

Inspecting the stop probabilities in Figure 5.3, I observe that the two ground-truth stops (29-43s and 76-97s) align well with the two spikes in stop probabilities. Visually, one can see that a minimum stop probability of 20% ($\epsilon = 0.2$) horizontally bisects the entries into two distinct groups; that is, the entries in the two spikes are classified as stops, and the jitter below that becomes moves. Specifically, $\epsilon = 0.2$ produces the following stop classifications: 37 true-positives, 79 true-negatives, 4 false-positives, and 0 false-negatives.

The visual selection of ϵ is quite effective, but somewhat subjective and time consuming; I provide a heuristic to estimate ϵ from the calculated stop probabilities.

The heuristic begins by computing the stop probabilities for each entry in a trajectory by using Equation 5.1. I then use k-means (MacQueen, 1967) ($k = 2$) to cluster these entry stop probabilities into two groups. I set $k = 2$ because I wish to find two distinct groups within the entry's stop probabilities (i.e. stops and moves). Once I have the two clusters, I take the maximum stop probability in the stop cluster, and minimum stop probability in the move cluster, and then find the value in-between these two boundary probabilities. This in-between probability value becomes an initial estimated minimum stop probability parameter (ϵ) value for POSMIT. Applying this heuristic to the short walk example, shown in Figure 5.3, yields $\epsilon = 0.39$, which produces 33 true-positives, 81 true-negatives, 2 false-positives, and 4 false-negatives. I evaluate the effectiveness of this heuristic in Section 5.3.4.

5.3 Experiments and Results

In the experiments I compared the classification effectiveness (Section 5.3.2, 5.3.3, 5.3.4, and 5.3.5) and running time efficiency (Section 5.3.6) of POSMIT against the clustering-based CB-SMoT (Palma et al., 2008) and a modified version of the geography-based SMoT (Alvares et al., 2007). CB-SMoT was chosen because I deem it to be a good representation of the many other density-based stop/move approaches; in no meaningful way is it superseded by these other approaches, either (recall my overview in Section 2.4.1). Similarly, SMoT was chosen because it represents a different paradigm for detecting stops/moves: using geography intersections. Recall, that in the original SMoT algorithm (Alvares et al., 2007), stops are found when a series of contiguous entries stay within a pre-defined geographic region for a minimum duration; however, in this work I assume stop regions are not known *a priori*. I provide the candidate stops to SMoT by partitioning the bounding-box of the study area into uniform grid cells of size R. In order to reduce redundant processing, I then remove any grid cells that the trajectory never passes through. To differentiate my modification from the original SMoT algorithm, I call this approach GB-SMoT (i.e grid-based SMoT). The effect of this modification is that GB-SMoT has a spatial parameter, like POSMIT and CB-SMoT, that I can vary in the experiments. In general, all of the experiments are directly designed to vary algorithm parameters and measure the results; thus, I present Table 5.1 —it may be useful to prepare readers for the following sections, where I refer to each algorithm's parameters frequently.

Algorithm	Parameter	Description
POSMIT	h_d	The stop variance (see Section 5.2.3).
	h_i	The index search bandwidth (see Section 5.2.4).
	ϵ	The minimum stop probability threshold (see Section 5.2.5).
CB-SMoT	Eps	The maximum sequential distance allowed in a stop cluster.
	t_{min}	The minimum duration of a stop cluster.
GB-SMoT	R	The grid-cell size used for candidate stop regions.
	t_{min}	The minimum duration of a stop visitation in a stop region.

TABLE 5.1: Parameters (besides input trajectories) for the algorithms used in the experiments.

In the following Sections, 5.3.2, 5.3.3, 5.3.4, and 5.3.5, I varied parameters and measured the resulting classification effectiveness of the tested algorithms. The classification effectiveness was calculated by using ground-truth input trajectories that already had their entries labelled as stops or moves (I discuss how these ground-truth trajectories were obtained in Section 5.3.1). These input trajectories were first passed into a stop/move classification algorithm that ignored the known labels and performed stop/move classification. Then, by comparing the computed stop/move labels to the known ground-truth labels, a series of true-positive, true-negative, false-positive, and false-negative observations were recorded. Finally, using these true and false observations, a binary classification effectiveness was calculated. The specific metric I calculated to measure the binary classification accuracy was Matthew's correlation coefficient (MCC). MCC was chosen because it represents the overall quality of a binary classification in a single value (unlike precision and recall) and captures all quadrants of the confusion matrix (unlike the various F-measures). Computation of MCC is shown in Equation 5.8, where the variables TP (True-Positive), TN (True-Negative), FP (False-Positive), and FN (False-Negative) are the quadrants of the confusion matrix.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{5.8}$$

5.3.1 GPS Trajectory Datasets

In the experiments, I used several real-world GPS trajectories with varying characteristics. Half of the real-world datasets I tested were stop/move annotated bus trajectories provided by Dublin city council's open data "Insight" project (DATA.GOV.IE, 2013). The other half of the real-world stop/move annotated trajectories I used were collected myself, for the experiments. Specifically, I recorded myself moving and stopping on foot and in vehicles using a custom built Android application [4]. The user interface of the Android application is shown in Figure 5.4. Unlike a typical GPS recorder, this application allowed me to manually annotate whether I was stopping or moving, thus, allowing me to create a ground-truth sequence of stop/move annotated spatio-temporal entries.

In addition to the real-world GPS trajectories, I also implemented a synthetic GPS trajectory generator to create sequences with vastly more entries than I was able to record. The synthetic trajectory generator begins by selecting a random starting location and heading; entries are, then, recorded travelling at a varying speed along the random heading until a quota of moving entries has been met. The generator then toggles and begins recording the entry at a stopped position until a quota of stopping entries has been met. This process continues until a specified number of entries has been recorded. Note, that each recording, moving or stopping, is displaced by an amount of spatial noise normally distributed between zero and a user-specified value. Real trajectories are far more nuanced and irregular with respect to when they stop or move; thus, the synthetic trajectories were not an ideal candidate to test the classification effectiveness of the algorithms. Therefore, in this work I only used the synthetic trajectories to gauge the running time of the algorithms tested.

In Table 5.2, I present the characteristics of each real-world trajectory I used in the experiments. I highlight that the trajectories I collected myself [5] and the bus

[4] https://github.com/lukehb/137-GPS-Tracker

[5] http://doi.org/10.13140/RG.2.2.29896.01281

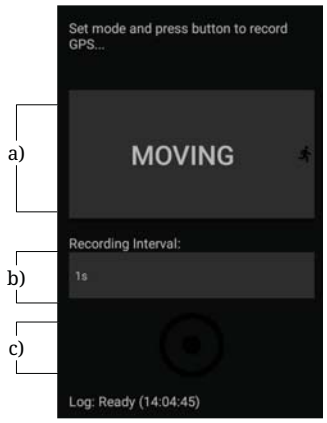


FIGURE 5.4: The Android application I made to collect GPS trajectories, the main functions of the application are as follows: a) the user can toggle whether they are moving or stopping; b) setting the target sampling interval, i.e. how often to try to record an entry; and c) beginning or ending a recording.

Name	Duration (s)	Mo (s)	Stops	Moves	Mode
Shopping	4211	1	195(11)	1773(11)	Foot, car
Hike	9531	1	932(10)	8592(9)	Foot, car
Ferry	638018	1	2651(44)	4961(43)	Foot, ferry
Bus A	42262	20	302(86)	742(86)	Bus
Bus B	42246	20	337(58)	801(58)	Bus
Bus C	42262	20	553(40)	601(40)	Bus

TABLE 5.2: Trajectories I used in the experiments. The format of stops and moves columns are *#observations (#episodes)* and the Mo column is the modal sampling-rate of each trajectory.

trajectories (DATA.GOV.IE, 2013) are publicly available. Additionally, the generator I used to create the synthetic trajectories is hosted in the same repository as my POSMIT algorithm [6]. Lastly, please note that the Ferry trajectory described in Table 5.2 was recorded over several days, making daily trips on a river ferry and then turning the tracking application off after walking to a nearby destination.

5.3.2 Varying Spatial Parameters

In this experiment, I compared the classification effectiveness and spatial parameter sensitivity of POSMIT algorithm against CB-SMoT and GB-SMoT. Additionally, I tested the effectiveness of the heuristic, presented in Section 5.2.3, at estimating POSMIT's h_d parameter. To effectively isolate the spatial parameters of each algorithm (h_d for POSMIT, Eps for CB-SMoT, R for SMoT) I had to set each algorithm's other parameters to reasonable constants. For example, POSMIT's search bandwidth, h_i, was set to a constant by using the estimation heuristic presented in Section 5.2.4 (note that estimating h_i in this way requires an input h_d value, which was itself estimated by the heuristic presented in Section 5.2.3). Likewise, CB-SMoT and GB-SMoT also required their t_{min} parameters to be set to reasonable constants. Recalling that the h_i value represents the range of values on

[6] https://github.com/lukehb/137-stopmove

either side of an entry, I realised I could use this as an approximate estimator for t_{min}. Specifically, I set the t_{min} value used for each trajectory to the calculated h_i value, multiplied by the modal sampling-rate of that trajectory (i.e $h_i * Mo$, where Mo is taken from Table 5.2). POSMIT's minimum stop probability parameter, ϵ, was set to multiple fixed values ($0.25, 0.5,$ and 0.75) and also estimated by the heuristic in Section 5.2.5. Overall, these parameter settings allowed me to vary each algorithm's spatial parameter as an independent variable, while recording MCC as the dependent variable. The results from this experiment are shown in Figure 5.5.

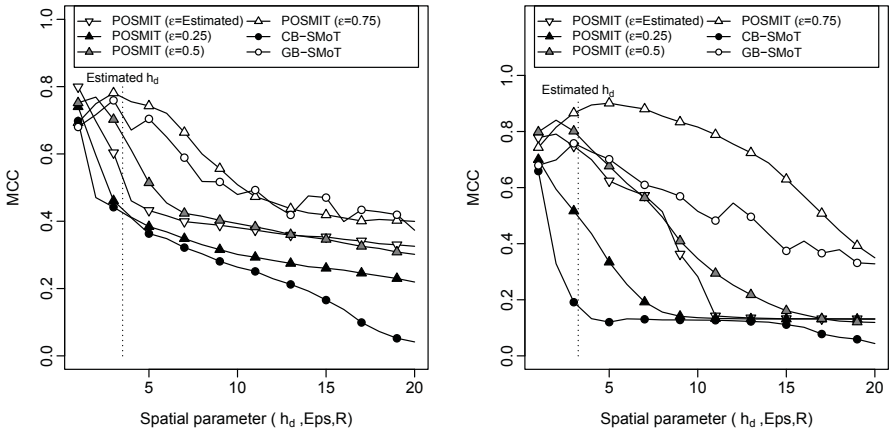
Analysis of Figure 5.5 reveals a number of observations that merit discussion. My first observation is that POSMIT achieves a higher maximum MCC than CB-SMoT and GB-SMoT for all of the tested trajectories. However, this result is not found across every POSMIT ϵ value used in the experiment. In general, this suggests that POSMIT may be more effective than CB-SMoT and GB-SMoT, but only when a suitable ϵ value is chosen. My second observation is that the range of spatial parameters that produce high MCC values is larger in POSMIT than it is in CB-SMoT and GB-SMoT. For the range of values I tested in Figure 5.5, I can determine that the best performing version of POSMIT has a spatial parameter range larger than CB-SMoT's and GB-SMoT's (respectively) by: 105% and 3% in Figure 5.5a; 373% and 35% in Figure 5.5b; 337% and 10% in Figure 5.5c; 12% and 16% in Figure 5.5d; 5% and 17% in Figure 5.5e; and, 8% and 27% in Figure 5.5f. This observation suggests to me that when POSMIT's ϵ is set to a suitable value, POSMIT is less sensitive than both CB-SMoT and GB-SMoT in regards to selecting the respective spatial parameter.

My third observation, from Figure 5.5, is that the estimated h_d produces a classification result, in the best performing version of POSMIT, that is above the best classifications produced by CB-SMoT and GB-SMoT. This, empirically, demonstrates the effectiveness and robustness of my estimation approach (recall Section 5.2.3) at selecting a reasonable h_d value for classifying real-world trajectories. My final observation is in regards to POSMIT's minimum stop probability parameter, ϵ. Specifically, in Figures 5.5a, 5.5b, and 5.5c, a higher ϵ value of 0.75 produces better MCC results; in Figures 5.5d, 5.5e, 5.5f, a lower ϵ value of 0.25 produces better MCC results. This is explained by the fact that the trajectories in Figures 5.5a, 5.5b, and 5.5c are more regularly sampled than those in Figures 5.5d, 5.5e, 5.5f (1s compared to 20s); thus, I would expect higher probability stop classifications from them. However, this does highlight that the manual selection of ϵ by the user should likely take the sampling-rate and noisiness (if possible) of the recorded trajectory into account. However, in the case, where making such assessments beforehand is not viable, the results show that the version of POSMIT that used the estimated ϵ values is a reasonable baseline: outperforming the poorly scoring ϵ values across all the tested trajectories.

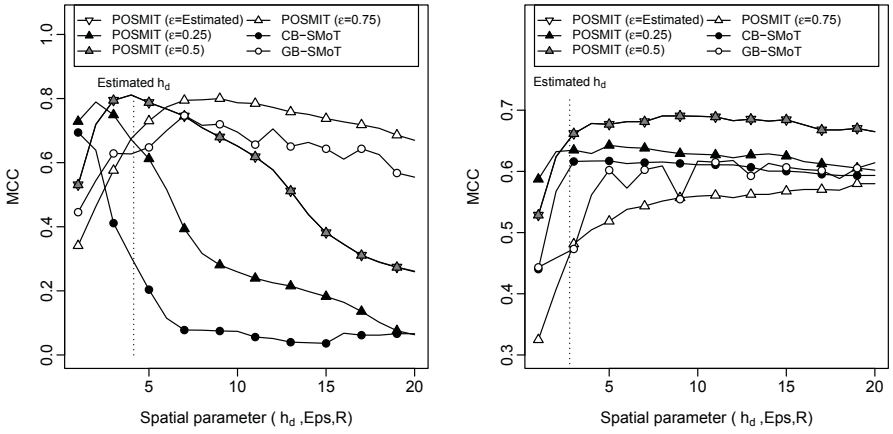
For the statistical significance test, I use one-tail pared t-test for average MCC values for the six datasets under study. POSMIT significantly improves CB-SMoT with 95% confidence, whilst POSMIT significantly improves GB-SMoT with 99% confidence.

5.3.3 Effects of POSMIT's Search Bandwidth

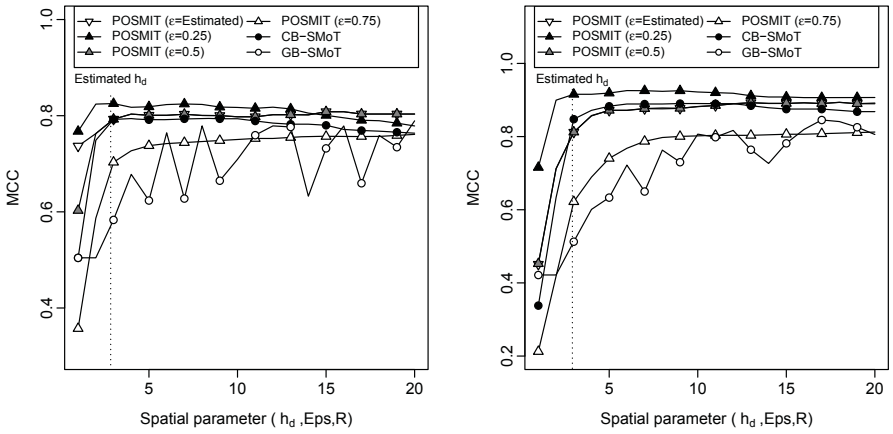
In this experiment I measured the classification effectiveness of POSMIT while its search bandwidth parameter, h_i, was varied. Additionally, I measured the classification effectiveness achieved when setting POSMIT's h_i value according to the estimation heuristic presented in Section 5.2.4. Furthermore, I also measured



(A) Shopping ($h_i = 6$, $t_{min} = 6s$). (B) Hike ($h_i = 20, t_{min} = 20s$).



(C) Ferry ($h_i = 15, t_{min} = 15s$). (D) Bus A ($h_i = 2, t_{min} = 40s$).



(E) Bus B ($h_i = 3, t_{min} = 60s$). (F) Bus C ($h_i = 5, t_{min} = 100s$).

FIGURE 5.5: POSMIT's, CB-SMoT's, and GB-SMoT's classification effectiveness (MCC) computed for several real world trajectories as their respective spatial parameters (h_d, Eps, R) are varied (a higher MCC is better).

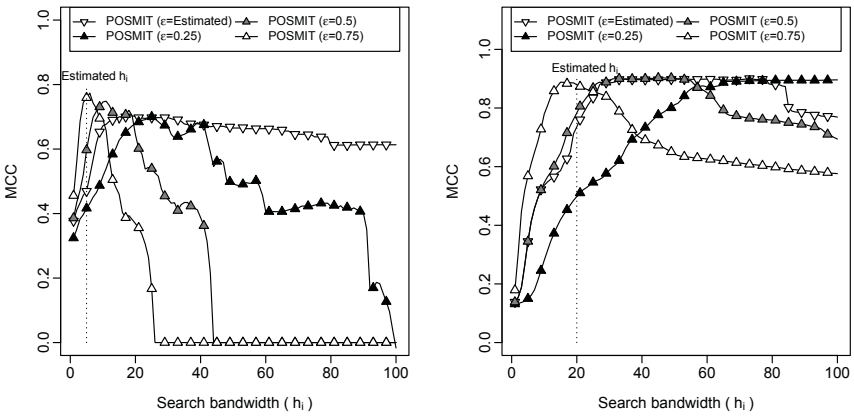
the relationship between h_i and ϵ by testing POSMIT using multiple fixed ϵ values ($0.25, 0.5,$ and 0.75) and also by setting ϵ to estimated values using the heuristic presented in Section 5.2.5. In addition to setting POSMIT's ϵ to constant values, I also set h_d to a constant, estimated by the heuristic from Section 5.2.3. These parameter settings allowed me to make h_i the independent variable, and MCC the dependent variable. The results of this experiment are presented in Figure 5.6. Readers, please note that CB-SMoT's and GB-SMoT's t_{min} are not analogous to POSMIT's h_i; therefore, CB-SMoT and GB-SMoT were not included in this experiment.

The first observation I make from Figure 5.6 is that, for each tested trajectory, there appears to exist an optimal range of h_i values that produce high MCC values. Additionally, h_i values outside that range produce worse MCC values the further away they are. The ideal, then, is to select h_i values that fall within this optimal range. The results indicate that the estimator I proposed for selecting such a h_i value (recall Section 5.2.4) did produce near optimal MCC values across all the tested trajectories. However, much like the h_d estimator I evaluated in the previous experiment, the h_i estimator was only able to achieve these results if POSMIT's ϵ parameter was set to a suitable value. Another observation I make, that has parity with the previous experiment, is that a higher ϵ value is more effective for the more frequently sampled trajectories, and a lower ϵ is more effective for the more sparsely sampled trajectories. In general, I observe that across all the test trajectories the version of POSMIT that used estimated ϵ values was more robust to h_i variances than its fixed-ϵ counterparts. Specifically, if one considers the results across all tested trajectories, the POSMIT version that used estimated ϵ values has a greater sum of MCC values than $\epsilon = 0.25$ by 13%, $\epsilon = 0.5$ by 59%, and $\epsilon = 0.75$ by 161%. This suggests to me that if the user is unsure of the minimum stop probability threshold, appropriate for their application, the ϵ estimator (see Section 5.2.5) provides a value for them that is consistently far from the worst manually selected value.

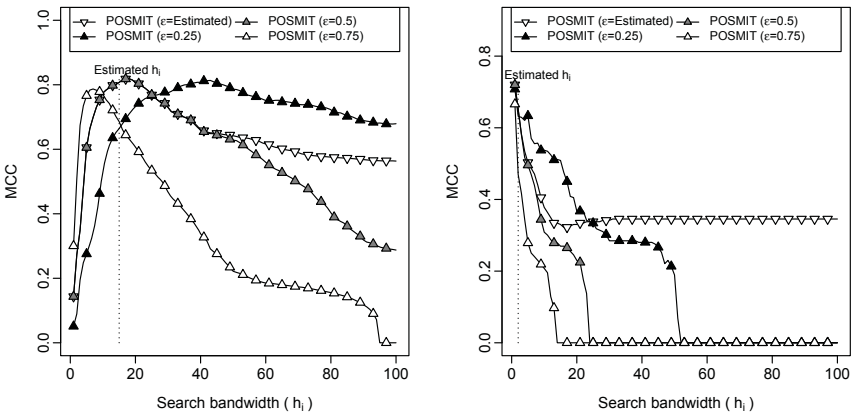
5.3.4 Effects of POSMIT's Minimum Stop Probability

In this experiment I measured POSMIT's classification accuracy while varying its minimum stop probability parameter, ϵ. Additionally, I measured the classification effectiveness achieved when setting POSMIT's ϵ value according to the estimation heuristic presented in Section 5.2.5. To isolate ϵ, POSMIT's other parameters, h_d and h_i, were set to constants using their respective estimation heuristics (recall Section 5.2.3 and 5.2.4). In contrast to the previous experiments, I chose to measure the raw values from the confusion matrix in this experiment because they reflect the variance in classification quality more granularly. Overall, these parameter settings allowed me to make POSMIT's minimum stop probability parameter, ϵ, the independent variable, and the quadrants of the confusion matrix (TP, TN, FP, FN), the dependent variables. Readers, note that CB-SMoT and GB-SMoT have no equivalent probability parameter to adjust; therefore, they were excluded from this experiment. The results of this experiment are shown in Figure 5.7.

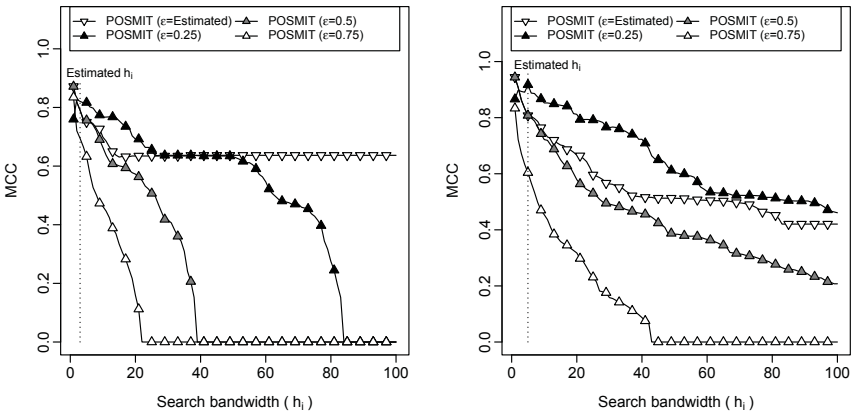
Analysis of Figure 5.7 reveals that as ϵ increases the false-positives become true-negatives, and likewise, as ϵ approaches the extreme values around one, the true-positives become false-negatives. This trend is consistent throughout all the tested datasets and provides valuable insight into tuning ϵ. For example, I observe that in all tested trajectories there exists an optimal range of ϵ values where the total



(A) Shopping ($h_d = 3.5m$). (B) Hike ($h_d = 3.25m$).

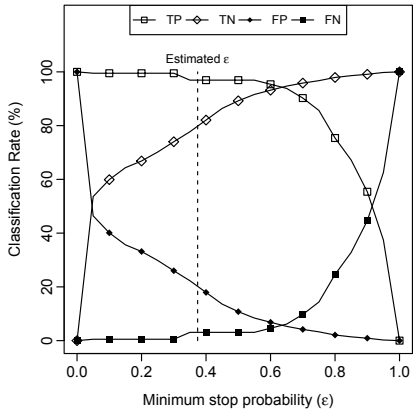


(C) Ferry ($h_d = 4.13m$). (D) Bus A ($h_d = 2.76m$).

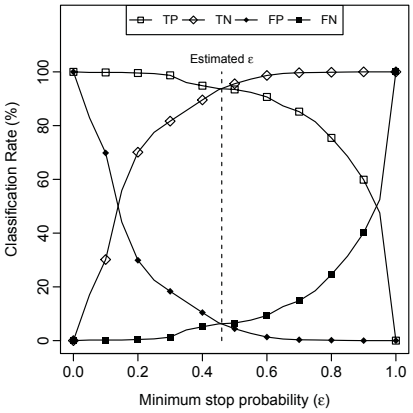


(E) Bus B ($h_d = 2.83m$). (F) Bus C ($h_d = 2.92m$).

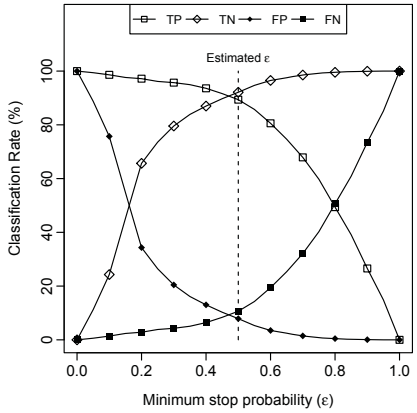
FIGURE 5.6: POSMIT's classification effectiveness (MCC) computed for several real world trajectories while its h_i parameter is varied (a higher MCC is better).



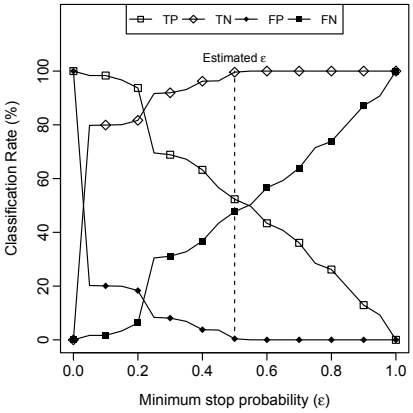
(A) Shopping ($h_i = 6, h_d = 3.5m$).



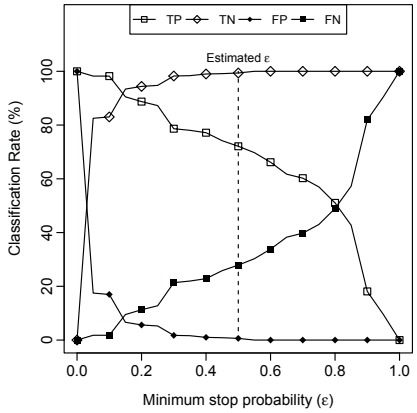
(B) Hike ($h_i = 20, h_d = 3.25m$).



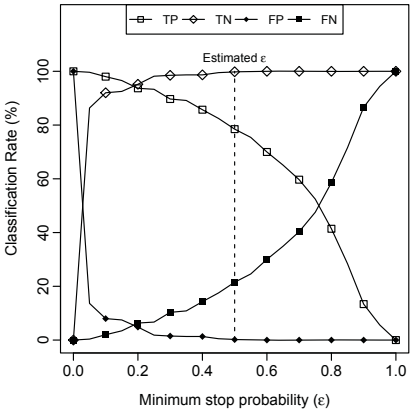
(C) Ferry ($h_i = 15, h_d = 4.13m$).



(D) Bus A ($h_i = 2, h_d = 2.76m$).



(E) Bus B ($h_i = 3, h_d = 2.84m$).



(F) Bus C ($h_i = 5, h_d = 2.92m$).

FIGURE 5.7: A comparison of each quadrant in the confusion matrix as POSMIT's minimum stop probability parameter, ϵ, is varied. Note, the dashed vertical line indicates the ϵ value estimated by the heuristic from Section 5.2.5.

number of false classifications (false-positives and false-negatives) is minimised and the total number of true classifications (true-positive and true-negatives) is maximised. This is an important finding because it shows that, even with noisy, real-world GPS trajectories, if a user controls the ϵ parameter it is possible to achieve accurate stop/move classifications. The challenge is selecting an ϵ within this optimal parameter range. One can see from the results that my heuristic for estimating ϵ is quite effective across all the tested trajectories. In fact, Figures 5.7b and 5.7c show the estimated ϵ is extremely close the optimal value. Even the worst performers, shown in Figures 5.7a and 5.7d, select ϵ values quite close to optimal range and are, certainly, far from selecting the poorly performing parameter values (i.e. extremely high or low ϵ values).

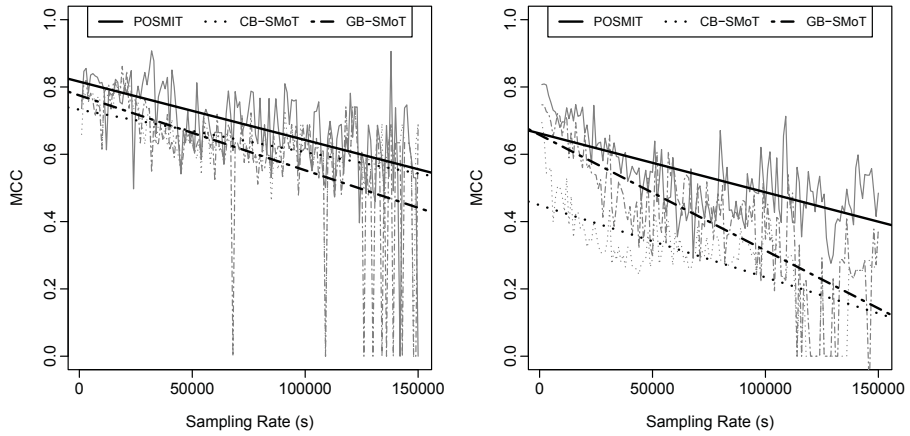
The last finding I highlight is that increasing the ϵ parameter very effectively reduces the occurrence of false-positive classifications. In fact, the results consistently show that sufficiently increasing the ϵ parameter reduces the number of false-positives to zero. Of course, the results also show that increasing the ϵ parameter too high reduces the number of true-positives. However, in an application where only finding the most likely stops (at the cost of losing some ambiguous stops) is preferred the results indicate that raising the ϵ value is an effective technique to do this.

5.3.5 Varying Sampling-Rate

In real trajectories, the sampling-rate, often, cannot be guaranteed or frequent because of battery life and GPS availability limitations. Thus, to gauge the performance of POSMIT, CB-SMoT, and GB-SMoT under such conditions, I measured the classification accuracy of each algorithm, while varying the sampling-rate of the input trajectories. All POSMIT's parameters were estimated by the various estimators I have discussed. Likewise, CB-SMoT and GB-SMoT had their parameters set to their best performing values from the experiments in Section 5.3.2. These parameter settings allowed me to make the sampling-rate the independent variable and MCC the dependent variable.

Regarding the sampling-rate, I varied it from one second to 150 seconds. This was achieved by taking the raw trajectory and storing its first entry in a new trajectory. Then, I iterated over the original entries, in order, until the total duration between the previously stored entry and the current entry was greater-than or equal-to the sampling-rate. Once this condition was reached, I stored the current entry and repeated the process until all the original trajectory entries were processed. The result was a new trajectory with, approximately, the desired sampling-rate. The limitation of this approach is the original trajectories had to be sufficiently large and frequently sampled. The only trajectories I had that met these conditions were the Hike and Ferry trajectories; thus, this experiment is limited to only those trajectories. The results of this experiment are shown in Figure 5.8.

Analysis of the results in Figure 5.8 confirms my expectation: that as the sampling-rate increases, the classification effectiveness of all the tested algorithms decreases. However, the gradient of each algorithm's performance is different, which suggests that some of the tested approaches are more sensitive to sampling-rate variances than others. Specifically, the overlaid trend lines show that, in general, POSMIT outperforms GB-SMoT and CB-SMoT across all sampling-rates; generalising, this suggests POSMIT may be more robust than CB-SMoT and GB-SMoT under various sampling-rates. Additionally, if one considers that all of POSMIT's numeric parameters are estimated in this experiment, then the results in



(A) Hike. (B) Ferry.

FIGURE 5.8: A comparison of each algorithm's classification effectiveness as the sampling-rate was made increasingly sparse (a higher MCC is better). Trend lines are overlaid on the raw results to indicate the general tendency of each algorithm.

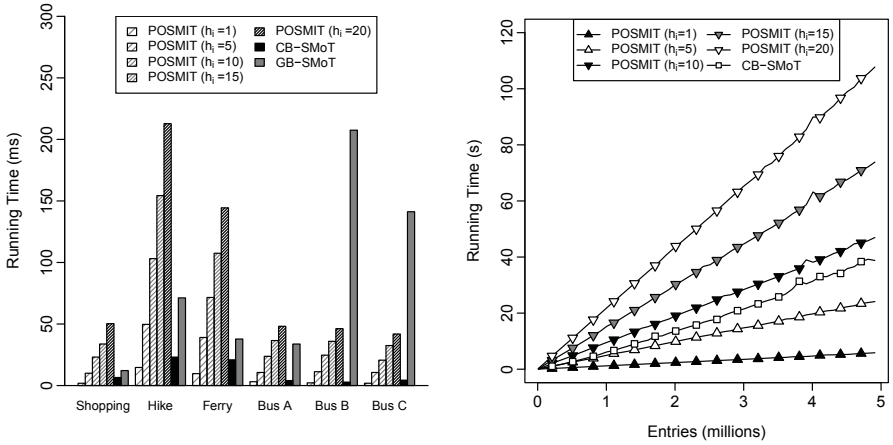
Figure 5.8 also indicate to me that fully automating POSMIT's parameter selection produces effective classifications, even under varying sampling-rates.

5.3.6 Running Time Efficiency

In this experiment, I measured the performance characteristics of CB-SMoT, GB-SMoT, and POSMIT by measuring their running time when processing various GPS trajectories. Each algorithm had all of its parameter set to the best performing values used in Section 5.3.2. The only exception to this was POSMIT's h_i parameter, which I set to multiple fixed values (specifically: 1, 5, 10, 15, and 20) to demonstrate its impact on running time. The first set of trajectories I used as input was the six real-world, noisy GPS trajectories I used in the previous experiments. In addition to these, I also used a second set of much larger trajectories that I generated using the generator described in Section 5.3.1. All the algorithms were implemented in Java; thus, the experiments underwent sufficient JVM warm-up runs and had an adequate maximum memory pool specified beforehand. Additionally, all timings from the experiments were averaged over the course of 10 runs. Finally, the test machine had an i5-M520 CPU and 5gb of freed working memory available prior to all tests.

Readers, please note that because my implementation of GB-SMoT partitions the study-space into a number of uniformly sized regions, its running time grows exponentially as either the user-specified region-size decreases or the study-space increases. The running time for my implementation of GB-SMoT was so large for the synthetic datasets tested in Figure 5.9b that, including it alongside the results for CB-SMoT and POSMIT would have eclipsed the difference between the two algorithms. Thus, for clarity, I have decided to exclude GB-SMoT's running time from Figure 5.9b.

Analysis of Figure 5.9a reveals that all three of the algorithms classified the six real-world datasets very efficiently: they all finished in less than one second, on all the real world trajectories. Additionally, the results in Figure 5.9a indicate that POSMIT's running time is mostly controlled by its h_i parameter. However, the results shown in Figure 5.9a, alone, are not sufficient to gauge the performance



(A) Real trajectories. (B) Synthetic trajectories.

FIGURE 5.9: POSMIT, CB-SMoT, and GB-SMoT's running times for classifying stops/moves in real and synthetic GPS trajectories (a lower running time is better). Note, the y-axis in Figure 5.9a is measuring milliseconds, whereas Figure 5.9b is in seconds.

characteristics of each algorithm as the input size increases. Therefore, to adequately test the performance characteristics of each algorithm, I generated much larger trajectories (millions of entries, as opposed to thousands) and benchmarked the running time of each algorithm (see Figure 5.9b for details).

The results shown in Figure 5.9b indicate that when $h_i = 1$ or $h_i = 5$, POSMIT's running time is superior to CB-SMoT's; however, once $h_i \geq 10$, POSMIT's running time exceeds CB-SMoT's. From Figure 5.9b, I observe that POSMIT's running time, with respect to input size, is linear (regardless of the selected h_i value); however h_i is a strong constant factor that governs the actual performance. Thus, I conclude that POSMIT's computational complexity is $O(nh_i)$; whereas CB-SMoT's is simply $O(n)$ as it does not have any parameter that impacts its running time.

5.4 Conclusion

Automatic classification of purely spatio-temporal GPS entries is certainly a challenging problem, especially when considering the varied and noisy recordings present in real-world trajectories. In this work, I have quantitatively shown, in Sections 5.3.2, 5.3.3, 5.3.4, and 5.3.5, that my solution to this problem, POSMIT, can achieve accurate stop/move classifications for real-world trajectories. Additionally, in Section 5.3.2, the results demonstrated that POSMIT is less sensitive to changes in its spatial parameter, h_d, than GB-SMoT's and CB-SMoT's are to their respective spatial parameters, R and Eps. On top of that, the experiments in Sections 5.3.2, 5.3.3, and 5.3.4 demonstrated that my parameter estimation heuristics for h_d, h_i, and ϵ performed reasonably on real-world trajectories —at times producing near optimal values. However, the experiments in Sections 5.3.2 and 5.3.3 also revealed that the h_d and h_i parameter estimation heuristics for POSMIT had their effectiveness strongly governed by how well POSMIT's ϵ was chosen. One of the design choices of the POSMIT algorithm is to allow the user to tune the minimum stop probability parameter, ϵ, to a threshold value that is suitable for their application (i.e some ambiguous true-stops may be classified as moves in exchange for a smaller result set of more likely stops). However, in cases where there is no

such application requirement, ϵ can also be estimated to try and maximise classification effectiveness. In Section 5.3.5, when all POSMIT's parameters were estimated, on two real trajectories, with varying sampling-rates, the combination of estimated parameters proved effective, as POSMIT's overall classification effectiveness was better than both CB-SMoT and GB-SMoT. This empirically demonstrated both: (1) the usability of POSMIT under fully automated numerical parameter estimation and (2) POSMIT's greater robustness to a range of trajectory sampling-rates. Though, as an aside, the experiments have also demonstrated that, if possible, trajectories should be recorded with the highest sampling-rate possible to increase stop/move classification accuracy.

In addition to parameter estimation, the experiments in Section 5.3.4 also demonstrated that the user can easily reduce the number of false-positives by simply steadily increasing the minimum stop probability, ϵ. However, I must clarify that it is not possible for the user to know whether increasing POSMIT's ϵ will decrease the number of false-positive classifications for their dataset in an unsupervised setting (i.e when there is no ground-truth). Regardless, I do argue that, at the very least, my goal from Section 5.1 was realised through POSMIT. Specifically, the results from Section 5.3.4 indicate that by using POSMIT's minimum stop probability parameter, ϵ, the user can filter out low probability stops that are unacceptable for their application. Last of all, in Section 5.3.6, I observed that POSMIT's performance remains linear to the size of the input trajectories and, at scale, can outperform CB-SMoT when $h_i \leq 5$. Overall, POSMIT's sampling-rate robustness, less sensitive spatial parameter, parameter estimation heuristics, stop filtering mechanism, and general classification effectiveness lead me to conclude that POSMIT is a useful addition to any location recommendation, trajectory data mining, or activity recognition application.

Chapter 6

Mining Semantic Patterns From Spatio-temporal Trajectories Using Complex Real-World Places

In this chapter I focused on addressing all four of the challenges I identified for trajectory data mining: spatial uncertainty, trajectory complexity, pattern complexity, and semantic meaning. I did this by bringing together several approaches and topics, introduced in the previous chapters, to create an overarching semantic trajectory data mining approach to mine semantic patterns from raw, free-space, spatio-temporal trajectories. Specifically, in this chapter, I introduced an approach called STOSEM, that mines semantic patterns from raw, free-space, spatio-temporal GPS trajectories using real-world places from OpenStreetMap. STOSEM begins by using POSMIT to annotate raw trajectory recordings as either stopping or moving. STOSEM then groups contiguously stopping entries into so-called stop episodes, each of which is then associated with a number of potential stop place candidates from the real-world place repository OpenStreetMap. Though, doing this then raised the issue of how to effectively match a single place to each stop when there are many nearby places. From my review of existing place-matching approaches, in Section 2.5, it seemed to me that none of the reviewed approaches handled all of the following in an unsupervised fashion: stop/move disambiguation; place topology; matching to point or polygon places; and considering the sequence of visited places during calculation. Thus, in this chapter I introduced a new, unsupervised place-matching approach that addresses each of the aforementioned issues.

The result of this stop episode formulation and place-matching is that the original trajectories are transformed into a discrete, greatly simplified, and more semantically meaningful sequence of place visitations. This format enables the last step of STOSEM, where frequent item-sets and sequential patterns are extracted using traditional approaches. Experiment results, with real and synthetic datasets, demonstrated STOSEM's running-time performance, robustness to GPS noise, dataset compression, and matching accuracy. Additionally, a case study, using human trajectories from the real-world Geolife dataset, revealed many interesting and seemingly real patterns. These findings suggested the general validity and applicability of STOSEM as a semantic trajectory data mining approach. The topics covered in this chapter are illustrated in bold in Figure 6.1.

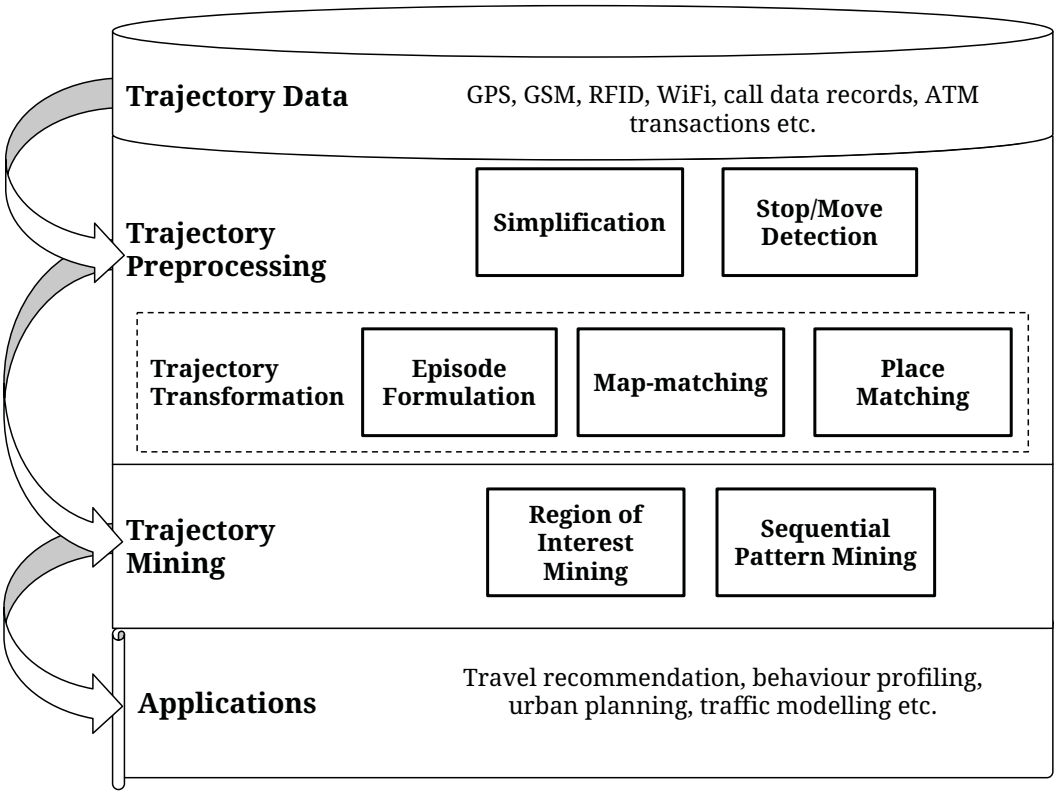


FIGURE 6.1: Overview of the trajectory data mining and knowledge discovery stages and the specific topics (the bold rectangles) covered in this chapter.

6.1 Introduction

Many trajectory patterns from first generation approaches are sequences of coordinates, spatial clusters, or spatial regions, which usually require visualisation on a map, or domain expertise by a human operator, to infer underlying semantic meaning (Cao et al., 2005; Nanni and Pedreschi, 2006; Giannotti et al., 2007; Lee et al., 2007). Put plainly, these types of trajectory patterns are not intrinsically human readable, or semantically meaningful, without using additional sources of information to add more context. Thus, as I mentioned in Section 1.2.2, the second generation of trajectory data mining approaches has shown much interest toward adding semantic information to raw spatio-temporal trajectories to semantically enrich them before data mining (Renso et al., 2013; Yan et al., 2013; Lv et al., 2016; Ghosh and Ghosh, 2017). As I discussed in the previous chapter, a common technique to semantically enrich trajectories is to detect when a tracked entity is stopping or moving (Spaccapietra et al., 2008). The assumption is that if an entity is stopping, especially for an extended period, that entity must be doing something at a particular place and, therefore, the stop itself is meaningful. That said, a series of stops is not very meaningful without additional context; it is context that provides meaning to a human operator when they interpret the results.

Fortunately, there now exists massive public repositories containing a useful type of contextual data: geographical places. Specifically, at the time of writing, the public has free access to Google Places, FourSquare, OpenStreetMap (OSM), and many government websites. These place repositories provide a wealth of context to associate with the detected stops. Thus, the task of place-matching emerges: given a series of trajectory stops, and a dataset of nearby places, each stop is matched to the most likely place that was visited. The result is a succinct and human readable set of place visitations that can easily be mined or queried to extract semantically meaningful patterns and knowledge. Readers, please note that I am not the first to investigate the problem of place-matching; thus, to serve as the motivation for this chapter I briefly recap the literature gaps I identified when reviewing existing place-matching approaches in Section 2.5.1.

I found there was no existing place-matching approach that could handle: points and polygon place geometries; stop/place disambiguation; the order of visited places; and place topology, all in an unsupervised setting without the need for expert-level parameter setting. Thus, in this chapter, I investigate such an approach. Before I proceed, I elaborate on the issue of stop/move disambiguation and place topology, from Section 2.5, using a diagram of an accentuating scenario.

Recall from the literature review that there is a number of place-matching approaches that simply match a stop with any intersecting place geometry (Liao et al., 2007; Alvares et al., 2007; Richter et al., 2012; Moreno et al., 2014; Boukhechba et al., 2015). However, assuming that a stop only intersects with a single place geometry is a highly idealised scenario which, in my experience, is rarely applicable to real-world spatio-temporal trajectories or complex real-world place repositories. Put plainly, this assumption does not hold in the scenario where multiple places intersect a given stop. However, one must select a single place to match with a stop; thus, the problem of stop/place disambiguation becomes apparent. I visually illustrate a stop/place disambiguation scenario containing topologically nested places in Figure 6.2.

Figure 6.2 presents a scenario where a stop occurs in a university that contains several topological nested places within it. In this scenario, it is ambiguous if the entity was truly visiting the coffee shop, college quad, lecture hall, toilets, or just

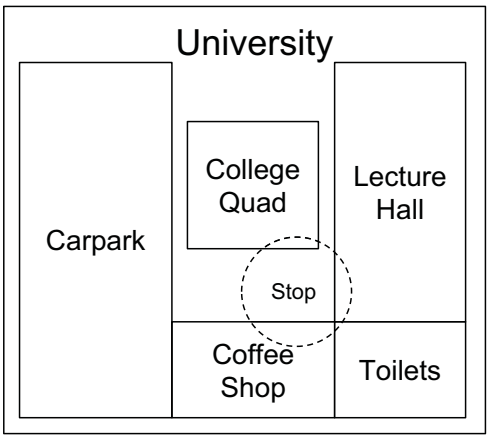


FIGURE 6.2: An ambiguous place-matching scenario where the stop is both encompassed by and overlapping with multiple places.

the university in general. I highlight, that such scenarios are not exceptional and, in fact, are common throughout real-world place repositories (especially OSM), as these repositories frequently store complex topological relationships between places. Thus, without addressing the issue of stop/place disambiguation or place topology, I assume that existing approaches must impose one of the following: (1) all places sampled for place-matching are spatially distinct (i.e non-overlapping) and distant from each other; or (2) stop/place disambiguation is handled by arbitrarily matching a given stop to one of the nearby places. Both of these options compromise the validity of any semantic patterns mined under these conditions. The first option effectively eliminates the usage of topologically complex real-world place databases such as OSM and the second option calls into question the accuracy of any places matched to stops in this manner. Therefore, it should now be clear why any such place-matching approach must address both of these issues.

The main contributions of this work are as follows:

1. I propose the combination of a stop/move classification algorithm with a stop episode clustering algorithm in order to identify stop episodes (Section 6.2.3);

2. I introduce a pre-processing step that selects and refines relevant real-world places from OSM for place-matching (Section 6.2.4);

3. I introduce an unsupervised place-matching algorithm that addresses the issue of stop/place disambiguation and place topology, while considering the order of visited places and achieving a high matching accuracy (see Section 6.2.5);

4. I introduce a semantic trajectory data mining approach that combines all of the above approaches with some traditional data mining approaches to transform raw spatio-temporal trajectories and OSM places into a succinct, human readable, and semantic set of place visitation patterns (Section 6.2);

5. I provide experiment results that demonstrate the effectiveness and efficiency of STOSEM (Section 6.3); and

6. I provide a case-study demonstrating the validity and applicability of my proposed approach (Section 6.4).

6.2 Method

Figure 6.3 introduces my approach for mining semantic patterns from raw spatio-temporal GPS trajectories. Throughout this work, I refer to my approach as STOSEM, which stands for "**S**patio-temporal Trajectories **To Sem**antic Patterns". STOSEM begins by annotating raw spatio-temporal trajectories as either stopping or moving using a stop/move classification algorithm. Next, each stop/move annotated trajectory is transformed into a sequence of stop episodes by clustering contiguous groups of stopping entries together. Each stop episode in the stop episode trajectories is then associated with nearby real-world places from Open Street Map that may have been visited. These candidate places are then modelled using a Hidden Markov Model and the most likely sequence of place visitations is matched to each stop episode trajectory. Given this sequence of place visitations, further semantic enrichment is also attempted by applying a heuristic that tries to detect likely entity homes. Lastly, these place-visitation sequences are mined to uncover semantic item-sets and semantic sequential patterns. Algorithm 4 shows STOSEM's specific steps, which I elaborate on in the following subsections.

6.2.1 Spatio-temporal Trajectories

In STOSEM spatio-temporal trajectories (recall Definition 2) are the input. Many previous trajectory data mining approaches, especially those in the first generation, attempt to perform data mining and knowledge discovery using raw spatio-temporal trajectories; in STOSEM, these raw trajectories are greatly simplified and semantically enriched before any data mining occurs.

6.2.2 Stop/Move Classification and Stop/Move Annotated Trajectories

STOSEM's first step is to annotate each raw spatio-temporal trajectory by adding an extra dimension to each recorded entry that indicates whether it is moving or stopping. The purpose of this extra stop/move dimension is to allow STOSEM to simplify raw trajectories into a sequence of extended stops; the assumption being, that extended stops occur at places that are semantically meaningful to the tracked entity and are, therefore, prime candidates for data mining. To produce these stop/move annotated trajectories, I use POSMIT from Chapter 5.

6.2.3 Stop Episode Clustering

The purpose of this stop and move information is to allow me to find extended sub-sequences of stopping entries and condense them into single discrete stop objects, which I call *stop episodes* (recall Definition 4). Converting stop/move annotated trajectories to sequences of stop episodes serves two purposes. Firstly, it greatly reduces the input size of the trajectories, and secondly, the discrete nature of the stop episodes allows them to be matched to a single real-world place.

The algorithm to form stop episodes is a one dimensional clustering algorithm that I call "SeCluster" (see Algorithm 4 line 5 for where SeCluster is called). SeCluster forms stop episodes by combining contiguous sub-sequences of stopping entries together into a single episode. I highlight that SeCluster also does this for moving entries to form move episodes. SeCluster has two parameters t_s and t_m that allow the user to filter out short stop and move episodes respectively. Note, a move episode that is filtered out by t_m becomes a part of the stop episodes that surround it.

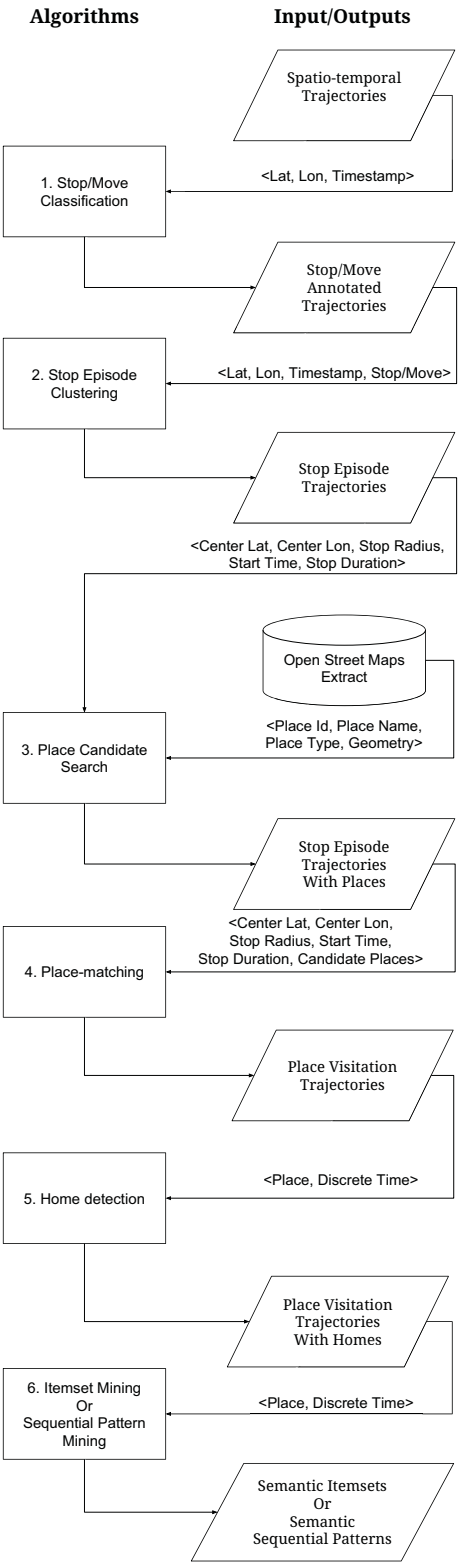


FIGURE 6.3: STOSEM: my approach for mining semantic pattern from raw spatio-temporal GPS trajectories.

Algorithm 4 STOSEM algorithm.

Input:
(1) D_T, a list of spatio-temporal trajectories $(T_1, T_2, \ldots, T_n)$.
Stop/Move classification (POSMIT) Input:
(2) h_i, index search bandwidth.
(3) h_d, stop variance.
(4) ϵ, minimum stop probability.
Stop Episode Clustering (SeCluster) Input:
(5) t_s, the minimum stop episode duration.
(6) t_m, the minimum move episode duration.
Place candidates search Input:
(7) r, the place candidate search radius.
(8) O, an OSM dataset extract.
Itemset/Sequential Pattern Mining Input:
(9) σ, the minimum support to make a pattern.
Output: A list of semantic patterns.

1: **function** STOSEM($D_T, h_i, h_d, \epsilon, t_s, t_m, r, O, \sigma$)
// **1 & 2. Classification and clustering.**
// An empty list of stop episode trajectories.
2: $D_{T_e} = ()$.
3: **for all** $(T_i \in D_T)$ **do**
// Annotate with stops/moves.
4: $T_{sm} =$ POSMIT(T_i, h_i, h_d, ϵ).
// Transform into stop episodes.
5: $T_e =$ SECLUSTER(T_{sm}, t_s, t_m).
6: Add T_e to D_{T_e}.
7: **end for**
// **3. Place candidate search.**
// Make an R-Tree, R_e from the stop episodes.
8: Make R_e from D_{T_e}.
// Read each place, P, from O.
9: **for all** $(P \in O)$ **do**
// Query for stop episodes near this place.
10: $E = R_e.query(P, r)$.
// Assign place candidate to stop episodes.
11: **for all** (Stop episode $e \in E$) **do**
12: Add P to e.
13: **end for**
14: **end for**
// **4 & 5. Place-matching and home detection.**
// An empty list of place visitation sequences.
15: $D_{T_p} = ()$.
16: $\lambda =$ LEARNPRS(D_{T_e});
// Find most likely sequence of visited places.
17: **for all** $T_e \in D_{T_e}$ **do**
18: $T_p =$ PLACEMATCHING(λ, T_e);
19: Add T_p to D_{T_p}.
20: HOMEDETECTION(T_p)
21: **end for**
// **6. Itemset and sequential pattern mining.**
22: **return** MINING(D_p, σ) .
23: **end function**

Stop episodes contain a stop centroid, a stop radius, a starting time, and a duration, which are all inferred from the stop entries that constitute the episode. Specifically, the stop centroid is calculated as the average spatial coordinate of all the stop entries in the episode. Once the stop centroid is known, the stop radius is calculated as the distance from the centroid to the most distant entry in the stop episode. Similarly, the starting time and duration are derived from the entries comprising the stop episode. The circular representation of stop is chosen because it inherently models the spatial uncertainty that is present due to GPS noise. As a small aside, I highlight that in this version of STOSEM, once stop episode clustering is complete, all move episodes are discarded as they are not used; however, in future works these move episodes could potentially be used to infer the mode of transport between stops.

6.2.4 Open Street Maps Extract and Place Candidate Search

OSM is a collaborative project where volunteers work together to create a free and editable map of the world. It is widely used in spatial analysis and spatial data mining, and I utilise OSM for STOSEM because it offers free downloads of its data in the form of binary files called extracts [1]. These extracts can be easily queried for places on disk, which is highly preferable in comparison to other similar data holders, like Google Places and FourSquare, that prefer users to query for places through Web interfaces.

In the OSM format, places have a unique id, some geometry, and a map of key-value description tags. I highlight that all the OSM extracts I used in this chapter are of whole cities. However, I assume that it is unlikely the tracked entities travelled to every place in the entire city; therefore, to reduce processing time, STOSEM only considers candidate stop places that are within a user-specified search radius, r, of the stop episode circles. In order to efficiently determine which stop episodes were within the search radius of a given place geometry STOSEM stores all the stop episodes in an R-Tree. This process is detailed in Algorithm 4 lines 8-14.

Once a place is assigned as a candidate for a stop episode, it is transformed, so that it becomes easier to work with during place-matching and mining. Specifically, recall the map of key-value tags that describes an OSM place, that map of tags is simplified to a single primary tag and a name (if present). For example, consider a tag map for a place called “Mars University": “name=Mars University", “amenity=university", “building=yes". In STOSEM, the place name is always extracted because it is the most semantically meaningful and human readable tag; however, the name does not necessarily describe the type of the place. Therefore, in addition to extracting the name in the “Mars University" example, I would generally consider “amenity=university" to be more descriptive than “building=yes"; thus, I would extract it as the primary tag. However, extraction of the primary tag is not always so clear. In fact, if there are many potentially descriptive tags, the task of selecting a single primary tag quickly becomes ambiguous.

My solution to this issue of tag ambiguity is to allow the user to provide a set of rules to score common OSM tags. Inventing some sort of descriptiveness-ranking between all OSM tags is not necessarily feasible because, in its entirety, OSM has thousands of active tags used to describe places [2]. Fortunately, for the purpose of

[1] `https://mapzen.com/data/metro-extracts/`

[2] A full repository of OSM tags can be explored at `https://taginfo.openstreetmap.org/`.

sport=*	(0.8)	military=*	(0.8)
club=*	(0.8)	aerialway=*	(0.6)
aeroway=*	(0.6)	amenity=*	(0.6)
tourism=*	(0.6)	highway=bus_stop	(0.6)
building=*	(0.6)	leisure=*	(0.6)
craft=*	(0.6)	cuisine=*	(0.6)
shop=*	(0.6)	harbour=*	(0.6)
healthcare=*	(0.6)	social_facility=*	(0.6)
mooring=*	(0.6)	office=*	(0.6)
railway=*	(0.6)	public_transport=*	(0.6)
repair=*	(0.6)	historic=*	(0.4)
industrial=*	(0.4)	man_made=*	(0.4)
place=*	(0.4)	room=*	(0.4)
natural=*	(0.2)	power=*	(0.2)
route=*	(0.2)	foot=*	(0.2)
highway=*	(0.2)	landuse=*	(0.2)
bridge=*	(0.2)	bicycle=*	(0.2)
building=yes	(0.2)	temporary_amenity=*	(0.2)
tunnel=*	(0.2)	water=*	(0.2)
waterway=*	(0.2)	*=no	(0)

TABLE 6.1: Descriptiveness scores for the OSM key-value tags I used in this study (a higher value indicates a more descriptive key-value tag). The asterisk indicates a wild-card for the key or value.

mining human trajectories, I have found there is a set of common OSM tags that effectively describes most places visited by humans in cities and urban environments. When an OSM tag is evaluated against this set of rules it is assigned the descriptiveness score associated with any rules it happens to match and, if no rules are matched, the tag is considered non-descriptive (except for the name tag which is extracted specially). In the event of a descriptiveness tie between two tags, the first tag in the list of tied tags associated with the place is chosen as the primary tag. I highlight that these descriptiveness scores worked well for discovering primary tags in my case; readers wishing to use STOSEM can easily adjust tags, accordingly, to suit their own purposes. The set of OSM tag rules are presented in Table 6.1.

6.2.5 Place-Matching and Place Visitation Trajectories

At this stage in STOSEM, I now have sequences of stop episodes that have nearby candidate places associated with each stop episode. However, the problem of stop/place disambiguation and place topology remains unaddressed; thus, I propose a HMM-based place-matching algorithm, where each sequence of stop episodes is used to probabilistically uncover the most likely sequence of visited candidate places.

As I discussed in Section 2.5.1, existing works (Yan et al., 2013; Lv et al., 2016) have already presented solutions to similar problems, where sequences of stops are matched to nearby PoIs using HMMs. However, I remind readers, that Yan et al. (2013) do not provide an automatic way to fully construct their HMM, stating that, “learning dynamic and personalized transition matrix [sic] is interesting but not the focus of this article". Additionally, Lv et al. (2016) require a sequence of visited

places as training data to construct their HMM. In contrast to these two approaches, I present a HMM for place-matching that is unsupervised and does not require any training data or expert initialisation.

Briefly, a HMM describes sequences of observable symbols that depend on hidden (unobservable) states. In this context, the observable sequence is the sequence of stop episodes, $T_e = \{e_1, e_2, \ldots, e_n\}$, and the hidden states are the candidate places associated with each stop episode that the entity may have stopped at. The candidate places associated with each stop episode are denoted as, $m(e_t) = \{p_{t_1}, p_{t_2}, \ldots p_{t_m}\}$. With the observations and states outlined, I introduce the general formulation for a HMM:

$$\lambda = (\pi, \mathcal{A}, \mathcal{B}) \tag{6.1}$$

where π is a set of initial state probabilities; $\mathcal{A}$ is a matrix containing all state to state transition probabilities; and $\mathcal{B}$ is the probability distribution for emitting a certain symbol. To define the terms in Equation 6.1, I must further define the hidden states in my model: the candidate places. Thus, let there be a total of q candidate places across all the stop episode sequences that one wishes to match: this results in a set of all place instances, $P = \{p_1, p_2, \ldots, p_q\}$. In general there are usually too many places to model the initial probabilities and transition probabilities of all of places; however, there is a comparatively small set of distinct place types - therefore, to calculate the initial (π) and transition probabilities ($\mathcal{A}$), I use the place types instead of the individual places themselves. Thus, let there be a total of g unique place types, resulting in the set of all place types, $C = \{c_1, c_2, \ldots, c_g\}$. Additionally, I denote deriving the place type from a place instance like so, $c = c(p)$. I now proceed to define each of the HMM terms: π, $\mathcal{A}$, and $\mathcal{B}$.

Initial State Probabilities

In my HMM, an initial state probability is the likelihood for a given place type to occur given no observation data. I approximate the initial state probability for each place type as the number of instances of that place type over the total number of place instances. More formally,

$$\pi = \left\{ \frac{|\{x : c(p_i) = c_1\}|}{q}, \ldots, \frac{|\{x : c(p_i) = c_n\}|}{q} \right\} \tag{6.2}$$

State Transition Probability

The transition probability is the part of my place-matching algorithm where the sequence of visited places is taken into consideration. Specifically, in my HMM, the state transition probability, $\mathcal{A}$, is modelled as a matrix of probabilities describing the likelihood of moving from one type of place to another. Specifically, $\mathcal{A} = Pr(c_a|c_b)_{g \times g}$. I assume that entities in the same trajectory dataset have somewhat homogeneous behaviours and, as a result, may generally visit the same types of places. Therefore, I construct the transition matrix of my HMM based on the potential visitations of the trajectory dataset as a group (as opposed to constructing $\mathcal{A}$ using a single sequence of stop episodes). I calculate the probability of transitioning from one place to another as follows:

$$\Pr(c_a|c_b) = \frac{|\{x : c(p_{t_i}) = c_a\} \cup \{y : c(p_{t+1_i}) = c_b\}|}{|\{z : c(p_{t_i}) = c_a\}|} \tag{6.3}$$

where $p_{t_i} \in m(e_t)$, $p_{t+1_i} \in m(e_{t+1})$, $\{e_t, e_{t+1}\} \subset T_e$, and $T_e \in D_{T_e}$.

I highlight that the initial state probabilities and transition matrix are calculated only once (see Algorithm 4 line 16) and remain the same throughout place-matching.

Emission Probability

The emission probability is the part of my place-matching algorithm where the place topology, point/polygon places, and, to some extent, the issue of stop/place disambiguation is handled. Specifically, in my HMM the probability of emitting a given symbol is modelled as the likelihood of observing a stop episode, e_t, at a given candidate place, p_{t_i}. That is, $\mathcal{B}_t = Pr(e_t|p_{t_i})$. Yan et al. (2013) calculate the emission probability between a stop and a type of PoI by modelling each PoI centre as a two-dimensional Gaussian function, where nearby PoIs, effectively, influence the stop. Ignoring the issue of how to select reasonable values for the Gaussian functions, this model does not translate well to my problem because I match stops to place geometries (as opposed to PoIs). In this problem, I assume visiting inside the boundary of a place should be no less likely than visiting inside the centre of that place. Thus, I calculate $Pr(e_t|p_{t_i})$ using the following set of cases that model the geometric relationship between a stop episode e_t, and one of its candidate places p_{t_i}:

1. **Disjoint:** When the place geometry and stop episode circle are completely disjoint, $Pr(e_t|p_{t_i}) = 0.5 \times \frac{d(e_t,p_{t_i})}{r}$ where $d(e_t, p_{t_i})$ is the smallest Euclidean distance between the stop episode centre and the place geometry; r is the user-specified search radius (recall the search radius from Section 6.2.4).

2. **Intersect:** When the place geometry intersects the stop episode circle, $Pr(e_t|p_{t_i}) = 0.5 + 0.5 \times \frac{A(e_t) \cap A(p_{t_i})}{A(p_{t_i})}$ where $A(e_t) \cap A(p_{t_i})$ is the intersection of areas between the stop episode circle and place geometry and, likewise, $A(p_{t_i})$ is the area of the candidate place.

3. **Contain:** When the stop episode circle contains the place geometry (or vice-versa), $Pr(e_t|p_{t_i}) = 1$.

Note, that the weightings applied to each of the three cases are designed to favour places that overlap or contain stop episodes, which is similar to the concept used in "area-stealing" interpolation (Gold, 1989). I highlight that calculating the emission probability directly addresses the issue of stop/place disambiguation, because each candidate is given a probability, rather than simply selecting the closest one. Additionally, calculating the emission probability, in this way (using intersection area), handles place topology, too, because the place that is most covered by the stop becomes more likely (even if two topologically nested places both intersect the stop this metric holds). Lastly, by calculating emission probability based on intersection and area, my place-matching can handle both points or polygons.

Matching places

Once the initial state probabilities and transition matrix are calculated, STOSEM begins place-matching (see Algorithm 4 line 18). Place-matching begins by constructing a HMM for the given sequence of stop episodes. The constructed HMM is a trellis that represents all possible place visitation sequences that the

sequence of stop episodes could reasonably be caused by. To extract the most likely place visitation sequence, I use the Viterbi algorithm (Viterbi, 1967). After the Viterbi algorithm is finished, a sequence of place visitations is returned, which STOSEM uses in the later mining steps.

6.2.6 Home Detection

In my preliminary tests on human trajectory datasets, I found that many patterns were produced, describing entities moving between generic, unnamed buildings. Further investigation revealed these generic buildings were recurring frequently, specific to each person, and mostly staying overnight, making them likely candidates as the person's home. Patterns describing movements between a person's home and some other place are obviously much more semantically meaningful than transitions to or from some generic building. Therefore, I developed a simple heuristic to detect the place that is most likely the person's home. The approach begins by finding all the likely home candidates that match any of the following types: "building=house", "building=residential", "building=apartments", or "landuse=residential". The next step is to count the number of times each of these likely home candidates is visited. Knowing the number of times each of these home candidates is visited, my heuristic is to label the most visited place (assuming it is visited more than once) as the person's likely home. The result of labelling a place as a home is that its place name becomes "home" and its place type becomes "building=house". Note, that I use this home detection step in all of the experiments below; however, it is optional. If home detection is not used in STOSEM it can be commented out in line 20 of Algorithm 4.

6.2.7 Data Mining Place Visitations

After the place-matching algorithm has uncovered the likely place visitation sequences, I begin mining those sequences to extract semantic patterns regarding the tracked entities. The place visitation sequences consist of a series of discrete places, which makes them suitable for both frequent itemset mining and sequential pattern mining. I can treat places visitation sequences either as a series of place names or a series of place types. Place names are more specific, and, thus reduce the pattern output: they are also more semantically meaningful to a human operator, as they identify a specific place; conversely, place types are more general because they group many real-world places under the same key-value tag, which results in more pattern output.

The difference between frequent itemset mining and sequential pattern mining is that frequent itemset mining will uncover the places in the dataset that are frequently visited in any order; sequential pattern mining will uncover the sequences of places that are frequently visited in a specific order. Additionally, each place visitation has a stop episode timestamp associated with it, which when discretised (i.e. into a time of day such as "afternoon") can be paired with the place to provide more specific and semantically meaningful frequent items and sequential patterns. In the mining step of STOSEM, Algorithm 4 line 22, I utilise the free and open source "Sequential Pattern Mining Framework" (SPMF) written in Java (Fournier-Viger et al., 2014b).

6.3 Experiments

To evaluate the efficiency and effectiveness of STOSEM, I conducted multiple quantitative experiments measuring the running time, accuracy, and compression compared to other approaches. Many components of STOSEM are found elsewhere, such as the stop/move classification and mining algorithms. Therefore, in the experiments I chose to quantitatively evaluate only the components of STOSEM that are unique to this chapter: the stop episode clustering and the place-matching. For the sake of comparison, I compare against a range of different trajectory simplification and transformation approaches. Specifically, I chose to compare SeClust against two simplification algorithms, DP and DRTA (from Chapter 3), and my place-matching algorithm against three transformation algorithms TPM (Giannotti et al., 2007), RMM Newson and Krumm, 2009, and a place-matching algorithm I implemented called Naive Place-matching. In addition to the quantitative experiments, I also empirically evaluated the applicability and overall validity of the results produced by STOSEM by performing a case study where I mined semantic patterns from a large real-world dataset of human trajectories.

In terms of the approaches I compared against, DP and DRTA are spatio-temporal trajectory simplification algorithms from Chapter 3. TPM is an algorithm that partitions the trajectory dataset into uniform grid cells and connects frequently visited cells together to form so-called regions of interest (RoIs). These RoIs are then used to transform a raw spatio-temporal trajectory into a sequence of RoI visitations (recall Section 2.1.3). RMM, which stands for road map-matching, is the name I give to the approach presented by Newson and Krumm (2009) that uses a Hidden Markov Model to transform raw spatio-temporal vehicle trajectories into a sequence roads driven (recall Section 2.1.2). Lastly, I compare my place-matching algorithm against another approach I implemented called Naive Place-matching. Naive place-matching matches each stop episode to the nearest place. The naive approach ignores the problems of stop/place disambiguation and place topology (recall Figure 6.2) and, thus, it serves as a representative for the existing approaches from the literature that do not consider these challenges (Renso et al., 2013; Yan et al., 2013; Lv et al., 2016; Ghosh and Ghosh, 2017).

6.3.1 Datasets

I conducted the experiments of this chapter using two datasets. The first was a large, real-world trajectory dataset tracking humans; the other was a synthetic dataset I generated, that roughly simulates human movement.

Geolife dataset

The real-world dataset I chose to study was the Geolife dataset (Zheng et al., 2008; Zheng et al., 2009; Zheng et al., 2010) collected by Microsoft Research Asia. The Geolife dataset contains human users moving around Beijing, and the greater area, over the course of five years (2007 to 2012). The users recorded their positions using various types of GPS loggers and phones. Additionally, the frequency of the recordings varies substantially between users —with some tracking every day and others tracking only very occasionally. This variance in tracking technology and frequency makes this dataset a good candidate to test the robustness, practicality, and, ultimately, the effectiveness, of my approach at handling real-world data.

Total participants:	179
Total entries:	20272567
Average trajectory size:	113254
Total duration (days):	33734
Average duration (days):	188.46

TABLE 6.2: Statistics for the pre-processed Geolife trajectories dataset.

00:00 to 06:00	→	“small hours"
06:00 to 09:00	→	“morning"
09:00 to 12:00	→	“before noon"
12:00 to 18:00	→	“afternoon"
18:00 to 00:00	→	“evening"

TABLE 6.3: Mappings I used for the Geolife dataset to transform time-stamps to times of the day.

Table 6.2 details the characteristics of the Geolife dataset after it was trimmed within the bounding box of Beijing.

Synthetic dataset

The synthetic dataset I used was generated by selecting 100 random places from the Beijing city OSM extract and simulating travel between them. In order to simulate a human trajectory, I simulated stopping at one of these random places for a set duration and, then, moving to another random place. This process was repeated until the required number of spatio-temporal recordings was generated. I highlight that, during each of these stop and move episodes, normally distributed spatial jitter (averaging 3 metres) was used to perturb the spatial coordinates. Furthermore, both stop and move episodes contained exactly 20 simulated spatio-temporal recordings that occurred over duration of 30 minutes. Finally, to control the total number of entries I varied both the total number of trajectories and number of stop/move episodes per trajectory.

6.3.2 Experiment Constants

Parameter tuning is a difficult, but it is essential in most data mining tasks (Han et al., 2011); it is data-dependent, and also allows users to have a control over the dataset under study. To reduce the number of experiment variables, and make the results simpler to compare, I assigned parameters in my framework to what I consider to be reasonable constants. Firstly, for POSMIT, I am, fortunately able to use to estimation schemes, introduced in Chapter 5. Conversely, the stop episode clustering, SeCluster, has two parameters, t_s and t_m, that I set to constants. Firstly, I set the minimum move duration, t_m, to 10 seconds, meaning move episodes shorter than that are merged with their neighbouring stop episodes to form a larger stop episode. Additionally, I set the minimum stop episode duration, t_s, to 10 minutes, meaning any stop episodes shorter than 10 minutes are disregarded. These parameters are motivated by the goal of finding meaningful stops: that is, I assume a 10 minute stop may indicate stopping at an important place, but a stop any shorter than 10 minutes may be caused by an uninteresting activity such as waiting in traffic.

Total participants:	101
Total visited places:	4134
Average sequences size:	41

TABLE 6.4: Statistics of Geolife place visitation sequences.

Furthermore, the place candidate searching step also has a single parameter: the search-radius r, that dictates which places are considered nearby and are eligible to become potential stop places. I set r to 200 metres, which is well outside the bounds of normal GPS noise (Defence, 2008) but still adequately large enough to encapsulate most nearby places. Additionally, as mentioned in Section 6.2.7, the temporal dimension of each stop episode can be discretised, appended as a suffix to the place names, and used during itemset and sequential pattern mining. Based on the datasets, I decided to convert all time-stamps in the stop episodes into times of the day. Table 6.3 contains the specific mappings I used to transform the time-stamps to times of the day.

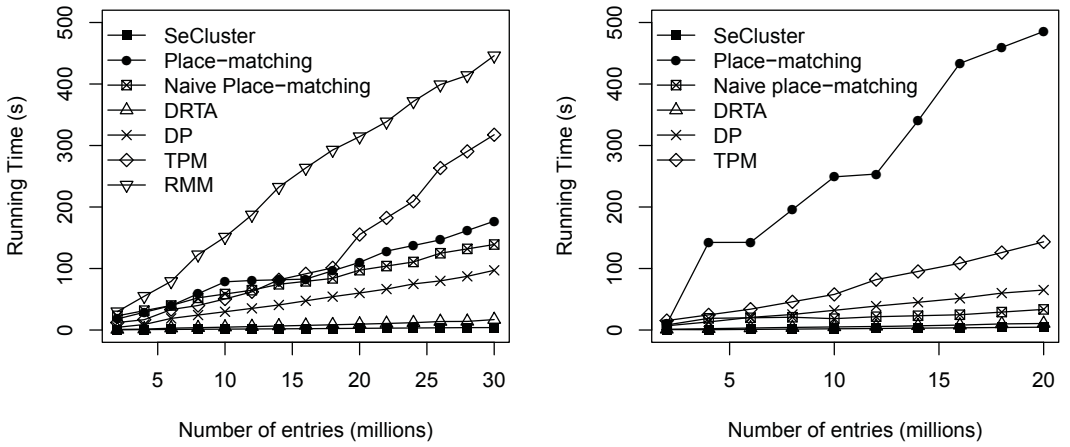
In Table 6.4, I present the statistics for the place visitation sequences that were produced using the defined constants. I highlight that, when I compare Table 6.2 and Table 6.4, the number of represented participants has decreased from 179 to 101: some participants either did not have any likely extended stop episodes or did not stop at any place known in the Beijing OSM extract. Additionally, I highlight that the number of entries has also dropped significantly. This is because stop episode clustering greatly reduces the complexity of the raw spatio-temporal trajectories.

I highlight that all experiments were conducted on a machine with an i5-520M processor and 5gb of unallocated memory. The experiments were all implemented in Java and were given adequate JVM warm-up runs. Lastly, for the experiments in Section 6.3.3,6.3.5, and 6.3.4 I varied the dataset size. This was achieved by truncating equal portions of individual trajectories until the desired total dataset size was reached.

6.3.3 Running Time

To measure the efficiency of each approach I recorded the time taken during by each of the approaches I tested (see Section 6.3 for a recap).

The results for the synthetic data are shown in Figure 6.4a. I observe, from Figure 6.4a, that my stop episode clustering algorithm, SeCluster, is quite efficient: processing up to 30 million entries under 10 seconds, which is similar to the other tested simplification algorithms DRTA and DP. Additionally, my place-matching algorithm outperforms some of the other trajectory transformation techniques, TPM and RMM, as the total dataset size increases. Additionally, I observe that my place-matching approach is only marginally slower than the naive place-matching approach, which is an acceptable trade-off for the increased accuracy shown in Section 6.3.4. In general, these results demonstrate the usability of my approach, in the context of other existing trajectory simplification and transformation approaches. Specifically, I observe that both SeCluster and my place-matching algorithm have running times that scale linearly with input size, and are able to process large datasets with millions of entries in a reasonable amount of time. In addition to the synthetic dataset, I also measured the algorithm running times on the Geolife dataset. The results are presented in Figure 6.4b.



(A) Synthetic dataset. (B) Geolife dataset.

FIGURE 6.4: Algorithm running times for the various trajectory simplification and transformation approaches I tested.

The main observation I make from Figure 6.4b is that, compared to Figure 6.4a, the place-matching algorithm takes far longer to process less entries. The difference in the place-matching running time occurs because the synthetic dataset has a fixed number of stop episodes for each of its 100 trajectories. Whereas, the Geolife dataset contains 179 trajectories and some trajectories have a large amount of stop episodes. Analysis reveals that, as the HMM trellis of potential place visitation sequences is built, the number of potential sequences grows with each potential state (stop episode candidate place) modelled. These results suggest to me, that the main factor controlling the running time of my place-matching algorithm, is the number of stop episodes per trajectory. One option to decrease this running time growth is to prune the HMM to only keep a certain selection of likely paths; though, the potential trade-off is a reduction in place-matching accuracy. I do not investigate this option in this chapter and instead leave it as a future direction.

6.3.4 Accuracy

In this section, I conducted two experiments. In the first experiment, I tested the effectiveness (f-measure) of SeCluster, DP, DRTA, and TPM at preserving stopping entries. I did this by measuring the number of true stops, false stops, and missed stops that were produced after each algorithm's simplification. In the second experiment, I measured the effectiveness (f-measure) of my place-matching algorithm at producing the true sequence of visited places taken by a given trajectory. In the place-matching experiment I compared my place-matching algorithm against a naive approach that matches each stop episode to the nearest place.

I highlight that both experiments required me to formulate a ground-truth of visited places for each trajectory. Unfortunately, there was no ground-truth of places provided in the original Geolife study. Due to presence of spatial noise and the problem of stop/place disambiguation, a manually created ground-truth for the Geolife dataset would also be of questionable validity; therefore, this test was only able to be conducted using synthetic trajectories. However, this had some advantages in terms of experiment design. Firstly, it allowed me to generate synthetic trajectories that only stopped at places that were ambiguous to correctly

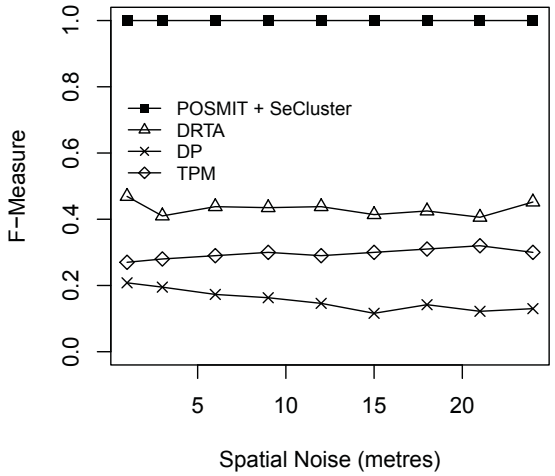


FIGURE 6.5: Accuracy of various approaches at preserving true stops (in a synthetic ground-truth) under varying spatial noise levels.

match against. That is, the places were either extremely close together or were topological nested in other places. Additionally, using synthetic trajectories enabled me to use the level of spatial noise present in the input trajectories as the independent variable. The noise was generated by a Gaussian function, with the mean set to the specified noise level and, then, each spatial coordinate of each trajectory entry was perturbed by a value generated by that function.

Additionally, I highlight that, according to the latest GPS specification (Defence, 2008), there is a 95% likelihood that spatial noise will be $\leq$ 7.8m when using modern GPS receivers under normal operating conditions (see (Defence, 2008) for a proper definition of normal operating conditions). Therefore, I conducted the place-matching classification effectiveness experiment using a range of spatial noise, from 1 to 10 metres, to encapsulate a normal operating range. The results of the stop preservation experiment are shown in Figure 6.5, whilst the results of the place-matching accuracy experiment are shown in Figure 6.6.

Figure 6.5 illustrates that DP, DRTA, and TPM poorly preserve the stops in the trajectory data: at best achieving approximately 50% accuracy. This result is expected though, as none of these approaches directly consider semantic information like moving or stopping. Conversely, SeCluster, which does consider stop and move information, perfectly preserves every stop, across all noise levels. However, I do highlight that the synthetic dataset is an ideal case for SeCluster because, in the prior stop/move classifications step, POSMIT can perfectly classify the stops and moves of the synthetic data; a real-world trajectory is far more complicated and nuanced, which means that perfect stop/move classifications are rarely achievable by POSMIT, and by extension, SeCluster.

Figure 6.6 illustrates that my place-matching algorithm achieved accuracies above 98% when the spatial noise was $\leq$ 2 metres, which is up to 28% more accurate than the naive approach, at some points within that range. Furthermore, under heavier noise, between 2 to 5 metres, my algorithm maintained accuracies above 70%, which when compared to the naive approach is approximately 13% better over that range. However, beyond 5 metres of noise, my place-matching algorithm's accuracy did drop quite quickly, approaching a similar accuracy to the naive approach at the noise levels beyond the specified normal noise range (i.e $\geq$ 7.8 metres). Investigation reveals that under heavier noise levels the stop episodes inherit much of the noise and are, often, perturbed away from their true

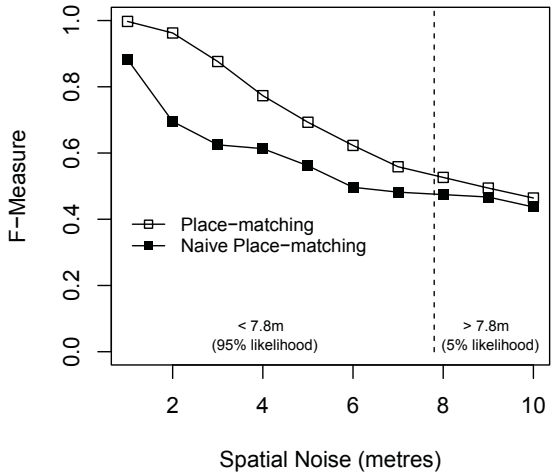


FIGURE 6.6: Accuracy of my place-matching algorithm and a naive place-matching algorithm at finding the true visited places (in a synthetic ground-truth) under varying spatial noise levels. Note, the 7.8 metres is from the latest GPS specification (Defence, 2008).

stop location, so much that the list of candidate places suggested for the stop, does not even contain the true place. Overall, I summarise from these results that my place-matching algorithm is accurate under low noise conditions. Additionally, my place-matching approach consistently outperformed the naive place-matching algorithm at all noise levels, which is a validation that my approach effectively addresses the problem of stop/move disambiguation and place topology.

6.3.5 Compression

In this experiment, the compression abilities of STOSEM and TPM were tested on both the synthetic and Geolife datasets. DRTA and DP were omitted because their output sizes are completely tunable. I formally define the compression value measured in this experiment in Equation 6.4. The results are shown in Figure 6.7.

$$Compression = 1 - \frac{|D_{PT}|}{|D_{ST}|} \tag{6.4}$$

Where,

- $|D_{PT}|$ is the total number of entries in the output after the algorithm runs (but before mining).
- $|D_{ST}|$ is the total number of spatio-temporal entries in the trajectory dataset before any processing.

Figure 6.7 demonstrates that STOSEM and TPM both achieved consistently high compressions across all datasets, at all sizes: both compressing up to 98% in some cases. I highlight, that even though the compression scores are high, the experiments in Section 6.3.4 demonstrate that STOSEM accurately preserves much of the semantic stop information —the same cannot be said for the TPM, DRTA, or DP algorithms. Therefore, Figure 6.7 is an interesting result because it suggests that my approach is an effective simplification technique for spatio-temporal trajectories that manages to retain semantic meaning in the output —something the traditional approaches I tested are ineffective at.

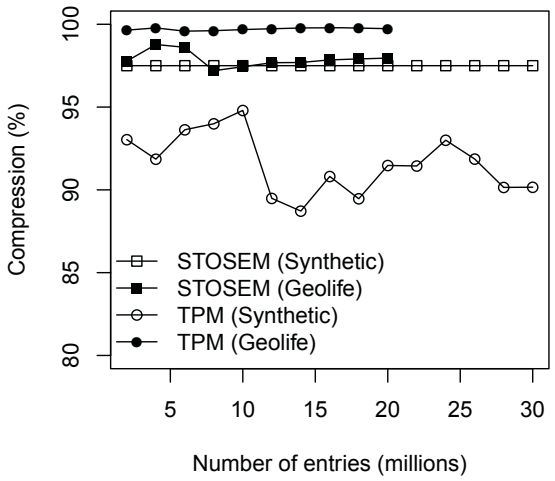


FIGURE 6.7: Compression achieved by STOSEM and TPM on the synthetic and Geolife datasets.

6.4 Case Study

To empirically evaluate the applicability and validity of STOSEM, I conducted a case study where I used STOSEM to mine semantic patterns from the real-world Geolife trajectory dataset.

6.4.1 Itemset Mining

The first mining task I performed on the Geolife dataset was to extract the top 20 frequent place and place-time itemsets. The results are presented in Table 6.5. Broadly, these results represent the popular places that were visited and, also, the combination of popular places and times of day that these visitations occurred at. I highlight that the results in Table 6.5 are frequent itemsets; therefore, any itemsets that contain more than one place or place-time are counting the visitation of those places in no particular order. That aside, there are a number of notable results to discuss.

First of all, the recurring presence of universities throughout the top 20 places and place-times is noteworthy. Specifically, 45% of the top 20 frequent place itemsets in Table 6.5 contain a university. Tsinghua University alone, constitutes 25% of the places and 35% of place-times in Table 6.5. In general, the results indicate quite a number of the participants made at least one visit to a university. In fact, Table 6.5 indicates that at least 26 of the participants from the Geolife study have some contact, specifically, with Tsinghua University. These results suggest to me that a number of the participants from the Geolife study were perhaps students or academics in Beijing.

Furthermore, analysing the place-time itemsets, I observe that some of the patterns with two items are between the same place at different times, such as, "TU (before noon), TU (small hours)". The presence of this result and other similar frequent item-sets indicate to me that these participants were likely attending classes scheduled at different times. This further reinforces my speculation that some of the participants were students or academics.

Another interesting observation I make from Table 6.5, is the presence of the Sigma Building. Investigation reveals that the Sigma Building is one of the buildings used by Microsoft Research Asia and is, in fact, the address listed by the authors of

Places	Support	Place-Times	Support
TU	26	TU (small hours)	20
Shanghai Metro Line 2	24	Beijing Metro Line 2 (small hours)	16
Home	23	Home (small hours)	14
Peking University	16	TU (before noon)	13
Beihang University	15	TU (before noon), TU (small hours)	11
Happiness Valley	11	TU (afternoon)	11
Renmin University of China	11	Peking University (small hours)	11
Summer Palace	10	TU (morning)	10
Old Summer Palace	10	Happiness Valley (small hours)	9
Qing River	10	Peking University (morning)	9
Peking University, TU	9	TU (small hours), TU (morning)	9
Sigma Building	9	TU (afternoon), TU (small hours)	8
Shanghai Metro Line 2, Home	9	Home (morning)	8
TU, Renmin University of China	8	Beijing Metro Line 2 (morning)	8
Home, TU	8	Renmin University of China (small hours)	8
Shanghai Metro Line 2, TU	8	Home (afternoon)	7
TU, Old Summer Palace	7	Home (before noon)	7
Zhichunlu Station	7	Summer Palace (morning)	7
Temple of Heaven	7	Beijing Metro Line 2 (before noon)	7
Beijing Capital International Airport	7	Happiness Valley (morning)	7

TABLE 6.5: Top 20 itemsets of places and places with times discovered for the Geolife dataset. TU stands for Tsinghua University.

the Geolife publications (Zheng et al., 2008; Zheng et al., 2009; Zheng et al., 2010). Additionally, in their works, Zheng et al. (2009) and Zheng and Xie (2011) state that 18% of the Geolife participants are Microsoft employees. The finding of the Sigma Building indicates to me that my approach has found a real pattern. Therefore, this finding is a minor validation of STOSEM, overall.

The next part of the case study was to extract the frequent itemsets of visited place types, both with, and without, the time of the day information (note: place types describe a place, whereas the top 20 places above name a place). The results from this experiment are shown in Table 6.6. Mining the place types as opposed to the actual places is more general and, as such, I did find a number of itemsets not present in Table 6.5.

My first observation is that 52 of the participants in the Geolife study visited a university at least once. Given the number of place visitation sequences in Table 6.4, this equates to a relative support of approximately 51%. This high support value further reinforces my speculation that a number of the participants in the Geolife study were students or academics. Investigation reveals that in their works, Zheng et al. (2009) and Zheng and Xie (2011) state that at least 62 of the participants in the Geolife study were college students. This parity between this finding and the actual participant demographics, again, indicates that the discovered results are real and, additionally, reinforces the validity of my approach.

Additional analysis of Table 6.6 reveals that 31 participants who went to a university also rode a bus. This is an interesting finding, as it further refines the university participants into a subgroup that catches buses. Habitual use of public transport may suggest that this subgroup of participants do not own personal vehicles. Another interesting observation is uncovered by comparing Table 6.5 against Table 6.6, and examining which places or types of places are only present in one table. Specifically, 35% of the place type itemsets contain bus stops; however, not a single bus stop is found in the top 20 place itemsets. This difference indicates to me, that whilst a number of participants caught buses, not many of them got on or off at the same bus stops. A similar explanation can explain the presence of a bank in Table 6.6 but not in Table 6.5.

Place Types	Sup	Place-Types-Times	Sup
amenity=uni	52	amenity=uni (small hours)	38
highway=bus stop	48	leisure=park (small hours)	34
leisure=park	45	amenity=uni (before noon)	30
building=yes	44	highway=bus stop (small hours)	28
amenity=uni, highway=bus stop	31	amenity=uni (before noon), amenity=uni (small hours)	27
leisure=park, highway=bus stop	29	leisure=park (morning)	27
amenity=uni, leisure=park	28	building=yes (small hours)	26
amenity=uni, building=yes	27	amenity=uni (morning)	26
amenity=restaurant	27	highway=bus stop (before noon)	24
waterway=riverbank	27	amenity=uni (afternoon)	23
building=yes, highway=bus stop	26	amenity=uni (morning), amenity=uni (small hours)	22
railway=subway	24	building=yes (before noon)	21
leisure=park, building=yes	23	building=yes (morning)	20
building=house	23	leisure=park (small hours), leisure=park (morning)	20
amenity=parking	23	amenity=uni (before noon), amenity=uni (morning)	19
amenity=uni, building=yes, highway=bus stop	21	amenity=uni (small hours), leisure=park (small hours)	19
waterway=riverbank, highway=bus stop	20	amenity=uni (evening)	19
amenity=uni, leisure=park, highway=bus stop	20	amenity=uni (small hours), amenity=uni (afternoon)	18
railway=station	20	amenity=uni (before noon), amenity=uni (morning), amenity=uni (small hours)	18
amenity=bank	19	highway=bus stop (morning)	17

TABLE 6.6: Top 20 itemsets of places types and place types with times discovered for the Geolife dataset. Uni is an abbreviation for university.

6.4.2 Sequential Pattern Mining

The second part of the case study was mining place and place-type sequential patterns from the Geolife dataset. Table 6.7 contains the top 20 most supported sequential patterns between both places and place-times. I highlight that, unlike frequent itemsets, sequential patterns capture the sequential nature of the data because they demonstrate the order in which place and place-time visitations frequently occurred. Additionally, unlike frequent itemsets, sequential patterns permit repeated visitations to the same place or place-time in a single pattern.

The major observation I draw from Table 6.7, is that some of the participants have tracked their habitual routines. These habitual routines reveal a great deal of information about the participants and allow me to begin making personalised profiles of certain participants or groups of participants. An example of this is found in Table 6.7, where a number of participants habitually and frequently went to Tsinghua University. Specifically, the results show that at least five of the participants went to Tsinghua University up to 32 times over the course of the study. This indicates to me that perhaps these participants were full-time students or employed at Tsinghua University. Additional analysis of Table 6.7 reveals that there exists subgroups of these Tsinghua University attendees who take different modes of transportation, beyond the buses observed in Table 6.6. Specifically, five participants came from a nearby car-park to Tsinghua University, whilst another five participants caught the Beijing Metro, Line 2.

These findings suggest to me that, in addition to those that catch the bus, there are at least two other types of participants from Tsinghua University: those who drive cars and those who catch the subway. I highlight that these subgroups of participants are not necessarily mutually exclusive. Another notable example that demonstrates the habitual behaviour of the participants is found in the bottom half

Place Sequence	Sup	Place-Times Sequence	Sup
Home, Beijing Metro Line 2	7	Renmin University of China (small hours) $\times 2$	6
Home $\times 8$	5	Renmin University of China (small hours), TU (small hours)	6
Home, Chinese Academy of Sciences $\times 2$	5	Summer Palace (morning) $\times 2$	5
Chinese Academy of Sciences $\times 3$	5	Beijing Metro Line 2 (morning), Beijing Metro Line 2 (before noon)	5
Peking University, TU $\times 3$	5	Beijing Metro Line 2 (small hours), Beijing Metro Line 2 (morning)	5
Peking University $\times 6$	5	Beijing Metro Line 2 (small hours) $\times 3$	5
Beihang University $\times 6$	5	Beihang University (before noon) $\times 2$	5
Old Summer Palace $\times 2$	5	Happiness Valley (small hours), Happiness Valley (morning)	5
Old Summer Palace, TU $\times 9$	5	Peking University (morning) $\times 2$	5
Summer Palace $\times 4$	5	Happiness Valley (small hours) $\times 7$	5
Beijing Metro Line 2 $\times 5$	5	Happiness Valley (morning) $\times 3$	5
Sigma Building $\times 6$	5	TU (small hours) $\times 9$	5
TU $\times 10$, Beijing Metro Line 2 $\times 2$	5	TU (before noon) $\times 6$, TU (afternoon), TU (before noon)	5
Happiness Valley $\times 8$	5	TU (small hours) $\times 7$, TU (afternoon)	5
Renmin University of China, Home	5	TU (before noon) $\times 8$, TU (afternoon)	5
Renmin University of China, TU $\times 8$	5	TU (before noon) $\times 2$, TU (small hours) $\times 7$	5
Beijing Botanical Garden $\times 2$	5	TU (before noon) $\times 6$, TU (small hours) $\times 4$	5
Carpark, TU	5	TU (before noon) $\times 6$, TU (small hours) $\times 2$, TU (before noon) $\times 2$	5
Home, TU $\times 4$	5	TU (before noon) $\times 6$, TU (small hours), TU (before noon) $\times 4$	5
TU $\times 32$	5	TU (before noon) $\times 8$, TU (small hours), TU (before noon) $\times 2$	5
TU $\times 6$, Home	5	TU (before noon) $\times 12$	5

TABLE 6.7: Top 20 length ≥ 2 place sequences (max sequential patterns) extracted from the Geolife dataset (TU $\times 3$ means TU, TU, TU). TU stands for Tsinghua University.

of the second column, in Table 6.7. Specifically, it appears that multiple Tsinghua University attendees have recorded trajectories that reveal their various attendance timetables. This finding, in particular, highlights the ability of STOSEM to automatically recover human readable, quickly interpretable, semantic patterns. Traditional spatio-temporal sequential patterns would have to visualise the patterns on a map to add semantic meaning; due to the dense overlap of patterns regarding Tsinghua University, this pattern may be obscured or at least difficult to uncover if searched for by eye.

Overall, Table 6.7 demonstrates how the presence of sequential information aids in discovering personalised and habitual patterns in the data. These kinds of patterns could not quickly be extracted by traditional approaches that require visualisation of the sequential patterns on a map. It is only due to my approach that transforms spatio-temporal trajectories into semantic trajectories that human readable results such as Table 6.7 are easily produced by applying traditional data mining techniques.

In the second part of this experiment I, additionally, conducted top-20 sequential pattern mining using place-type visitation sequences. The results are shown in Table 6.8. Table 6.8 reveals many of the same patterns as the other type of top 20 patterns I have shown so far —for example the university attendee timetables. However, I do highlight one particular type of pattern in Table 6.8, the visits to parks, such as "leisure=park $\times 6$" and "amenity=uni, leisure=park $\times 2$". Investigation reveals that many of the universities I have discovered in the results are close to parks and nature strips. Furthermore, input from domain experts has revealed that the parks near the universities are often used for social clubs and

activities associated with the university. For example, “The English Corner" is public square (classified as “leisure=park") location within the Renmin University campus, where people gather to speak English.

Place Type Sequence	Sup	Place-Type-Times Sequence	Sup
building=yes ×3	24	building=yes (small hours) ×2	19
highway=bus stop, leisure=park	23	leisure=park (small hours) ×3	17
amenity=uni ×2, building=yes	22	amenity=uni (morning), amenity=uni (small hours)	17
leisure=park, building=yes	21	amenity=uni (morning) ×2	17
amenity=uni ×2, leisure=park	21	amenity=uni (before noon), amenity=uni (morning)	16
highway=bus stop ×4	21	amenity=uni (small hours), leisure=park (small hours)	16
leisure=park ×6	21	amenity=uni (small hours) ×2, amenity=uni (before noon) ×2	16
leisure=park, highway=bus stop ×2	20	amenity=uni (small hours), amenity=uni (afternoon)	15
highway=bus stop, amenity=uni, highway=bus stop	20	amenity=uni (before noon), amenity=uni (afternoon)	15
highway=bus stop, building=yes ×2	20	amenity=uni (small hours), highway=bus stop (small hours)	15
amenity=uni, highway=bus stop ×3	20	amenity=uni (small hours), leisure=park (morning)	15
building=yes, amenity=uni ×2	20	leisure=park (small hours), amenity=uni (small hours)	15
highway=bus stop, building=yes, amenity=uni	19	leisure=park (small hours), leisure=park (morning) ×2	15
amenity=uni, highway=bus stop, building=yes	19	amenity=uni (small hours), amenity=uni (morning), amenity=uni (before noon)	15
amenity=uni, building=yes ×2	19	amenity=uni (small hours), amenity=uni (before noon) ×2, amenity=uni (small hours)	15
amenity=uni, building=yes, amenity=uni	19	amenity=uni (small hours), amenity=uni (before noon), amenity=uni (small hours), amenity=uni (before noon)	15
highway=bus stop, building=yes, highway=bus stop	19	highway=bus stop (small hours) ×2	15
amenity=uni, leisure=park ×2	19	amenity=uni (small hours), amenity=uni (before noon) ×3	15
amenity=uni, leisure=park, amenity=uni	19	amenity=uni (small hours) ×3, amenity=uni (before noon), amenity=uni (small hours)	15
leisure=park, amenity=uni ×2	19	amenity=uni (small hours) ×4, amenity=uni (before noon)	15

TABLE 6.8: Top 20 length ≥ 2 place type sequences (max sequential patterns) extracted from the Geolife dataset (amenity=uni ×3 means amenity=uni, amenity=uni, amenity=uni). Uni is an abbreviation for university.

6.5 Conclusion

Overall, I have presented my approach, STOSEM, which transforms raw spatio-temporal trajectories into place visitations that can be mined for semantic patterns. In Section 6.2.4 I presented my process for extracting places and primary tags from a repository of real-world places (OSM). Additionally, in Section 6.2.5, I introduced my unsupervised HMM-based place-matching algorithm, that considers the order of visited places, overcomes the problem of stop/place disambiguation, and handles place topology to produce sequences of likely place visitations. Quantitatively the results have shown that the combination of POSMIT and SeCluster is not only an effective simplification approach for spatio-temporal trajectories, but, unlike other existing simplification approaches, also highly effective at preserving the semantically meaningful stops. Furthermore, the results

have demonstrated my place-matching approach is less efficient than approaches that do not address stop/place disambiguation; however, in exchange for this increased running time the accuracy of my approach is shown to be superior (up to 28% more accurate in some cases). Lastly, I presented a case-study where STOSEM was applied to some real-world human trajectories from the Geolife study. The results from the case study demonstrate STOSEM was able to uncover multiple seemingly-real semantic patterns from the raw data. Furthermore, the results revealed multiple habitual behaviours from the participants: many involving universities and transport. The combination of these case study findings lead me to conclude that my approach is valid and applicable for mining semantic patterns from real-world trajectories and places.

Chapter 7

Conclusion

In this final chapter of my thesis I draw some overarching conclusions about the research and discuss its contributions to the greater body of work. Specifically, I begin this chapter with Section 7.1, which recaps the previous data chapters for the reader's convenience. Then, in Section 7.2, I evaluate each of the research hypotheses I outlined in Section 2.6. I follow this with Section 7.6, which highlights the limitations of each data chapter and some future research directions and extensions for the approaches I have introduced. Next, I discuss the theoretical and empirical contributions of the work in Section 7.3 and follow this with a discussion of the real-world contributions in Section 7.4. Finally, in Section 7.5 I conclude with some overall discussion of the research and some speculation about the future of trajectory data mining.

7.1 Chapter Summaries

Before I proceed, I present a brief recap of each chapter for the convenience of the reader.

Chapter 3. The aim of this chapter was to investigate trajectory simplification as a technique to mitigate the challenge of data complexity in raw trajectory datasets. In this chapter I presented a generalised framework for creating trajectory simplification approaches by combining entry scoring functions and processing strategies together. In all, I used five scoring functions and four processing strategies, some taken from existing poly-line and trajectory simplification approaches and others new, to create a total of twelve trajectory simplification algorithms. I quantitatively evaluated each of these algorithms in experiments, testing their respective running times on various datasets and, also, testing their preservation error in terms of synchronised Euclidean distance, synchronised area, and region-of-interest visitation. The experiment results demonstrated that I produced several $O(n)$ trajectory simplification algorithms that have efficient running times, whilst maintaining low SED and SA error levels. Additionally, the experiment results revealed that, in general, the simplification algorithms I created preserved the original visited regions (even after heavy simplification of 90%).

Chapter 4. The aim of this chapter was to investigate redundancy-controlled, contiguous sequential pattern mining as a technique to mitigate the challenge of pattern complexity in large map-matched vehicle trajectory datasets. Specifically, in this chapter, I presented DC-SPAN, my algorithm for mining a distinct (succinct and redundancy-controllable) set of contiguous sequential patterns. Using several real-world map-matched vehicle trajectory datasets, I evaluated DC-SPAN against equivalent closed- and maximal-contiguous sequential pattern mining algorithms in terms of running time, compression, lossiness, and redundancy. The experiment results demonstrated that DC-SPAN can, indeed, control the redundancy of the

pattern output with only a marginal increase in running time required. The results also showed that when DC-SPAN's maximum redundancy was set to 0%, DC-SPAN produced a pattern output approximately 30 to 60% less redundant than the set of max-contiguous sequential patterns, and 60 to 95% less redundant than the set of closed-contiguous sequential patterns. This highlights DC-SPAN's main unique contribution: its vastly lower and controllable pattern output redundancy.

Chapter 5. The aim of this chapter was to investigate a probabilistic stop/move classification approach that addresses the challenge of adding semantic meaning to raw trajectories, while also accounting for the challenge of spatial uncertainty present in real-world GPS recordings. Specifically, in this chapter, I introduced a stop/move classification algorithm called POSMIT that calculates the stop probability of each entry by considering its neighbouring entries and accounting for the presence of spatial noise. By calculating the stop probability of each entry, the user is able to set a threshold parameter to filter out stop classifications that are not likely enough for their application. To evaluate and compare POSMIT to existing approaches I collected real-world ground-truth datasets myself, using a custom Android application that allowed me to annotate stops and moves as I made them. Additionally, I found a publicly available bus dataset online, that contained stops and moves that I used as a second ground-truth dataset. Lastly, I generated some synthetic trajectories with millions of entries to test scalability. Using these as my ground-truths, I compared POSMIT to two existing approaches in terms of running time, classification effectiveness, parameter sensitivity, parameter estimation, and sampling-rate robustness. The results demonstrated POSMIT achieved higher classification effectiveness scores, less sensitive parameters, effective (sometimes near optimal) parameter estimations, and a greater robustness to sparse sampling-rates.

Chapter 6. This chapter was my final data chapter and, therefore, I used it to present an overarching approach that combined several topics and concepts from the previous chapters together. The aim of this overarching trajectory data mining approach was to account for the challenges I identified in the introduction: spatial uncertainty, trajectory complexity, pattern complexity, and adding semantic meaning. Specifically, I introduce trajectory data mining approach, called STOSEM, that mines frequent place visitation itemsets and sequences from raw spatio-temporal trajectories. STOSEM begins by using POSMIT to extract stop/move annotated trajectories. Then, these stop/move trajectories are transformed into discrete stop/move episodes using an algorithm I introduced called SeCluster. Next, each stop/move episode is associated with any nearby place geometries from OpenStreetMap. Using a novel probabilistic place-matching algorithm (that I introduced) each sequence of stop episodes is associated with the most probable sequence of visited places. This place-matching algorithm is different from existing approaches because it considers: the problem of stop/move disambiguation, place topology, the sequence of visited places, and matching to place geometries that are points or polygons —all in an unsupervised manner without the need to expert-level parameters or training data. Once place-matching is finished, the sequences of visited places are discrete and succinct; thus, they are mined by existing itemset and sequential pattern mining approaches. Experiment results with real and synthetic datasets demonstrate STOSEM's running time performance, robustness to GPS noise, dataset compression, and place-matching accuracy. Additionally, a case study using human trajectories from the real-world Geolife dataset revealed many interesting and seemingly-real patterns. These findings suggest the effectiveness of my place-matching algorithm and also the

general validity and applicability of STOSEM as a semantic trajectory data mining approach.

7.2 Evaluation of Research Hypotheses

I present an evaluation of each of my research hypotheses in the following subsections.

7.2.1 Trajectory Simplification Hypothesis

The research hypothesis for Chapter 3 was:

> *"A framework that generalises the problem of trajectory simplification into a combination of significance scoring functions and processing strategies, will be able to extend existing poly-line simplification algorithms to trajectory simplification approaches. These new simplification approaches will outperform their original counterparts in regard to spatio-temporal error metrics such as SED."*

In Chapter 3, I identified the scoring function and processing strategy of three spatial simplification algorithms (Douglas and Peucker, 1973; Visvalingam and Whyatt, 1993; Lee et al., 2007) and extended them using my framework so they could simplify spatio-temporal trajectories. I conclude that this research hypothesis is satisfied as the results in Section 3.3.2 demonstrated that SPLPD, EXTA, GRPPA outperformed their 2d counterparts (the original algorithms) in both the SED and SA experiments at all simplification levels.

7.2.2 Vehicle Trajectory Pattern Mining Hypothesis

The research hypothesis for Chapter 4 was:

> *"Mining map-matched vehicle trajectories, using a contiguous sequential pattern mining approach that prunes the pattern output, will result in a set of roads-driven that are unique (non-overlapping) and less redundant than the output from existing contiguous-closed or max-sequential pattern mining approaches."*

In this chapter I proposed DC-SPAN, which effectively post-processes the output of another contiguous sequential pattern mining algorithm to produce a redundancy controlled pattern output. I have shown quantitatively in the experiments in Section 4.4.5 that when the maximum redundancy is set to 0% the percentage of redundant pairs (i.e redundant road segments in the case of vehicle trajectories) is also 0%. In other words, by setting a maximum redundancy of 0% one can achieve unique (non-overlapping) set of travelled roads when mining vehicle trajectories; thus, satisfying my research hypothesis for this chapter.

7.2.3 Probabilistic Stop/Move Classification Hypothesis

The research hypothesis for Chapter 5 was:

> *"A probabilistic stop/move detection algorithm with a minimum stop probability threshold parameter will allow low probability (i.e. ambiguous) stop classifications to be filtered out by the user; this will ultimately result in a more controllable false-positive rate and a higher overall classification accuracy than existing approaches."*

The experiments in Section 5.3.4 quantitatively demonstrated that a user can change the stop probability parameter to filter out low probability entries from the classification result. Additionally, it was found that when the user increases the minimum stop probability parameter, the number of false-positives consistently decreased in the tested datasets: a finding I would expect to hold in other datasets. Lastly, it was also demonstrated in the experiments in Section 5.3.2 that POSMIT achieved a higher maximum classification effectiveness than the two compared approaches and, additionally, that POSMIT generally had a wider range of parameters that produced high classification effectiveness scores. Thus, I conclude that the research hypothesis for this chapter is fulfilled.

7.2.4 Probabilistic Place-matching Hypothesis

The research hypothesis for Chapter 6 was:

> *"A probabilistic place-matching algorithm that considers stop/place disambiguation, place topology, and the sequence of visited places will be more accurate than place-matching approaches that form matches by using spatial intersections."*

The experiments in Section 6.3.4 quantitatively demonstrated that my place-matching algorithm, which considers stop/place disambiguation, place topology, and the sequence of visited places, was more accurate (even under varying noise levels) than the representative algorithm I compared against that did not consider these features. Thus, I conclude that the research hypothesis for this chapter is fulfilled.

7.3 Theoretical and Empirical Contributions

I consider theoretical and empirical contributions as those that introduce ideas, concepts, methods, or results that contribute to the greater body of trajectory data mining research.

The first such contribution I highlight is the generalisation I introduced in Chapter 5 to describe and create trajectory simplification algorithms. Specifically, I introduced the notion of significance scoring functions and processing strategies, which I demonstrated can be easily combined together to create new and tailored trajectory simplification schemes. Furthermore, the generalisation seems to be a useful technique to describe and compare various trajectory simplification schemes. Thus, I conclude that this contribution may be useful to future researchers to explore and create new trajectory simplification algorithms, and also, more generally, as a tool to conceptualise and compare various approaches.

Chapter 4 also contains a number of theoretical contributions. Specifically, there is the notion of redundancy controlled output through lossiness, the supporting concepts of pairs and cover, and the notion of distinct patterns. As I highlighted in Section 2.3, many of the existing sequential pattern mining approaches reduce the

pattern output by applying some sort of constraint on the patterns. Yet, as I demonstrated in Section 4.1, such approaches can produce highly redundant pattern outputs when the input sequences are sufficiently homogeneous. Thus, I investigated the notion of a truly lossy sequential pattern mining algorithm where some subsequences are discarded entirely to produce a redundancy-controlled pattern output. Except for the similar problem of mining a compressing set of sequential patterns (Vreeken et al., 2011; Lam et al., 2014), this notion of discarding patterns is relatively unexplored —most likely because most sequential pattern mining approaches want to keep all relevant patterns. However, I argue I have shown in Chapter 4 that mining a redundancy-controlled output can be useful in specific applications, such as large map-matched vehicle trajectory datasets. Thus, by introducing my notion of distinct patterns, and providing an investigation into mining them, I argue that I have contributed to the greater body of sequential pattern mining work. I encourage future researchers to find applications where a redundancy-controlled sequential pattern mining output may be helpful and, in those cases apply distinct pattern mining.

Chapter 5 also contains two contributions: one theoretical and the other empirical. The theoretical contribution is the core concept behind POSMIT: probabilistically finding stops and, thereby, indicating the degree of certainty associated with each classification. This allows the user to filter out stops that are not certain enough for their application. This notion of providing an indication of a “goodness-of-classification" in the result is not a feature afforded by any of the existing approaches I reviewed. However, the experiment results in Section 5.3.4 demonstrated it is effective and allows the user to easily filter out less likely classifications (i.e. potential false-positives). As the results have shown POSMIT to be effective, I consider the notions of probabilistic stop/move classification and “goodness-of-classification" to be useful concepts for the field going forward; I encourage future researchers to incorporate and extend them with their own stop/move classification or general data mining schemes.

Additionally, the empirical contributions from Chapter 5 are the parameter estimation schemes I provided. None of them are novel or unique from existing parameter selection tasks. However, I argue their application to the problem is an empirical contribution to the field: from my review in Section 2.4, very few approaches provided parameter estimations (and none provided estimations for all parameters, like I did). Thus, given the effectiveness of the parameter estimation algorithms as demonstrated in the experiment results, I consider my parameter estimation approaches for POSMIT a useful tutorial for future researchers dealing with the problem of stop/move classification.

Lastly, there are three contributions from Chapter 6 that merit discussion. The first contribution is the place-matching algorithm I introduced. It is most similar to existing works by Yan et al. (2013) and Lv et al. (2016), in that is uses a probabilistic approach based on HMMs like they do; however, unlike those approaches, my place-matching is unsupervised and does not require any expert-level parameter setting or training datasets. Specifically, the concepts I use to calculate the emission probability and transition probability in the HMM —considering place geometry, place topology, and the order of visited place types —is a those are theoretical contributions future researchers can benefit from, extend, and compare against.

The second theoretical contribution from Chapter 6 is the presented trajectory data mining approach: STOSEM. I consider this a theoretical contribution for future researchers because it introduces an overarching technique for producing sequences, frequent itemsets, and sequential patterns of place visitations from raw

trajectories. It contains various concepts, of which many are not entirely new contributions; however, the combination of them all together, in this way, is unique, as far as I can tell. As STOSEM is relatively modularised I encourage future researchers to extend STOSEM by introducing further semantic enrichment steps and, ultimately, find even more knowledge rich trajectory patterns. The final contribution from Chapter 6 is the empirical contribution of the case study performed in Section 6.4. The contribution is not necessarily the specific findings but, more generally, that place visitation sequences can reveal behaviour profiles and frequent places/place sequences. I suggest that STOSEM, or at least its combination of techniques, may lead to novel trajectory data mining applications, particularly given that the only external requirement to run STOSEM is a place repository (of which there are now many excellent choices).

7.4 Real-world Contributions

I consider the real-world contributions of this thesis to be tangible research outputs that others may use in the future. Unfortunately, from the experience of conducting this thesis, it seems apparent that very few approaches are being shared publicly in the trajectory data mining community. This limits the growth of the field, as a whole, and makes objective comparison, among approaches, more difficult than it needs to be.

Thus, I consider the major real-world contribution of this thesis to be the sharing of my approaches. I have investigated the overall process of trajectory data mining and knowledge discovery, produced a number of algorithms as a result, and am sharing all of them publicly. Specifically, I produced new algorithms relating to: trajectory simplification [1], redundancy controlled contiguous sequential pattern mining [2], stop/move detection [3], and place-matching/semantic pattern mining [4] —all of which I have made publicly available for future researchers. I highlight that sharing all the algorithms is far more useful than sharing one, or some, because, as a whole, they represent an entire semantic trajectory data mining tool-kit, which as I have shown in Chapter 6, can be used to make knowledge discoveries from real-world trajectories. Additionally many of the source code repositories I share, also contain my synthetic data generators, evaluation frameworks, and existing algorithms for map-matching, trajectory RoI mining, and stop/move detection —all of which I expect may be useful for future researchers. Overall, I encourage future researchers to use, extend, and compare all the approaches provided to improve the greater body of knowledge in the field of trajectory data mining.

Other real-world contributions of my research include the ground-truth stop/move annotated trajectories I collected to evaluate POSMIT in Chapter 5. Again, I have made these publicly available[5]. From my review of existing stop/move detection approaches in Section 2.4, it seemed apparent to me that the field was sorely lacking a public ground-truth dataset they could use to evaluate and compare various approaches with. Thus, I have contributed such a dataset to the field along with the Android application used to collect it [6].

[1] https://github.com/lukehb/137-simplification
[2] https://github.com/lukehb/137-SPM
[3] https://github.com/lukehb/137-stopmove
[4] https://github.com/lukehb/STOSEM
[5] http://doi.org/10.13140/RG.2.2.29896.01281
[6] https://github.com/lukehb/137-GPS-Tracker

7.5 Concluding Remarks

As discussed in Section 1.2, this thesis is situated in the second generation of trajectory data mining, where the focus has shifted towards higher level semantic knowledge and real-world applications that use a combination of techniques. Earlier in this thesis, in Section 1.4, I identified four challenges that I consider to be prevalent now and for the future of the field: spatial uncertainty, trajectory complexity, pattern complexity, and semantic meaning. Each chapter I investigated one or multiple of these challenges in regard to various topics within the field of trajectory data mining.

Recalling the stages of the trajectory data mining and knowledge discovery process —data, pre-processing, transformation, mining, knowledge discovery, and application —this thesis has allowed me to investigate the relevant challenges at each stage. Specifically I explored: pre-processing through trajectory simplification in Chapter 3, trajectory transformation and mining through distinct pattern mining in Chapter 4, trajectory transformation and semantic meaning through stop/move classification in Chapter 5 and, lastly, all the stages in Chapter 6. Chapter 6 is particularly important to my research because that chapter is effectively a culmination of new approaches (i.e. place-matching), existing approaches, and work from previous chapters to produce STOSEM: my overarching semantic trajectory data mining approach. Additionally, in Section 6.4 of Chapter 6, I went a step further than the other chapters and performed a case-study using real-world human trajectories from the Geolife dataset. This revealed patterns that suggested several interesting behaviour profiles and groups of people who appeared to be university students or academics. These results were succinct, human readable, and highly semantically meaningful. Thus, I consider the results from this case study to demonstrate that my overall investigation has actually addressed each of the challenges of spatial uncertainty, trajectory complexity, pattern complexity, and semantic meaning. More broadly, I consider that culminating work to fulfil my initial intention of "An Investigation Into Data Mining Large And Noisy Spatio-Temporal Trajectories To Produce Succinct and Semantically Meaningful Patterns".

In terms of where this leaves the field, I have already outlined specific contributions in Section 7.3 and Section 7.4; however, more broadly, the four challenges will still persist going forward but, perhaps, they will be less of a problem as the continual research effort, which I have contributed to, produces approaches to address them. Thus, we may currently be close to a third generation of trajectory data mining where the challenges of the past are easily solved and the focus is strongly on real-world semantic applications. For example, we may see real-time decision making, smart transport, and smart urban planning applications have the technological, data availability, and public adoption prerequisites to become mainstream trajectory data mining research topics.

7.6 Limitations and Future Directions

Reflecting on this thesis, I identify a number of limitations and extension topics that I now recommend as future research directions.

7.6.1 Trajectory Simplification

The first new direction for the work in this chapter relates to the normalisation process used in my framework. It essentially normalises each spatial and temporal dimension into the unit space $[0, 1]$, using the minimum and maximum observed values for each dimension of the dataset. Normalising the data in this way imposes a mapping between spatial and temporal coordinates; however, I did not investigate the reasonableness of this mapping, nor its trade-offs compared to other schemes. Such an investigation may be an interesting research direction, but I recommend something different: investigating removing the normalisation step entirely. Specifically, the normalisation step in my framework could be removed if one the significance scoring functions are changed to do their own spatio-temporal considerations.

For example, as I discussed in Section 2.2, a number of works have already replaced the perpendicular distance functions in the Douglas-Peucker algorithm (Douglas and Peucker, 1973) with SED because it naturally incorporated both spatial and temporal dimensions. In this chapter I already introduced SA, which could be used to replace the area scoring function in the Visvalingham-Whyatt algorithm (Visvalingam and Whyatt, 1993) and, therefore, make the normalisation step unnecessary for that approach. Considering the synchronising scheme used in SED and SA, it seems feasible to invent a synchronised angle measure and a synchronised parallel distance measure —doing so would cover all the significance scoring functions I introduced in this chapter (speed is already implicitly spatio-temporal so no special consideration is required for that). Therefore, I recommend this as one future direction for the field of trajectory simplification approaches, in general, because it bypasses the need for normalising the whole dataset and handles the space-time relationship at a per entry level, which is surely more effective.

Another limitation of this chapter are the types of significance scoring functions evaluated. I only investigated approaches that calculate significance based on spatio-temporal features. However, as I discussed in Section 1.4.4, the field of trajectory data mining seems to be leaning towards uncovering semantically meaningful patterns; thus, it follows, that a trajectory simplification approach should score entries based on their semantic significance. For example, consider a significance scoring function designed to preserve the features used for stop/move detection (i.e. local sequence density and velocity). A semantic scoring function is challenging though because it raises the task of comparability. For example, how does one answer the question: is this entry more semantically meaningful than this other entry? I speculate preserving semantics will be an important research direction for trajectory simplification going forward.

The final limitation I consider for this chapter is that my framework was designed to only produce offline trajectory simplification approaches. However, as devices and infrastructure get better and cheaper there will, undoubtedly, be a greater call for real-time trajectory data collection and on-device simplification. Thus, as a specific future research direction for my framework I recommend extending it to produce online trajectory simplification approaches. The most difficult challenge in this direction appears to me to be the problem of requiring user-specified algorithm-specific error tolerance parameters. Doing so would make combining various modules together more difficult because the user would, then, be required to predict the characteristics of the incoming data, and also have some understanding of the internals of each specific scoring function they wish to use.

7.6.2 Vehicle Trajectory Pattern Mining

The first limitation I identify for this chapter is that DC-SPAN mines the pattern output of another algorithm. Therefore its performance characteristics are implicitly tied to that algorithm and more specifically it cannot outperform that algorithm. In order to break this dependency, I recommend investigating the mining of distinct-contiguous sequential patterns within a single algorithm.

The second limitation of this chapter is that I focused solely on mining vehicle trajectories; however, other fields of study, such as biology, may also find the patterns produced by DC-SPAN useful. Thus, as a future research direction, I recommend a case-study mining distinct patterns in various relevant domains and comparing the usefulness to existing approaches used in those domains. Another limitation I identify is that my implementation of DC-SPAN requires the sequence database to fit into machine memory; however, many sequence databases (especially vehicle trajectory datasets) are too large to be loaded into memory. Thus, I recommend investigating an on-disk data-structure or distributed computing modification for DC-SPAN.

The final limitation I identify is that the lossiness of DC-SPAN is unbounded, which may be unsuitable for some use-cases. Considering how the redundancy constraint is maintained in DC-SPAN, it seems plausible to me that an additional constraint of maximum lossiness could also be maintained if another user-specified parameter was passed in. Thus, my final recommendation for this chapter is to investigate extending DC-SPAN to have a maximum lossiness parameter.

7.6.3 Trajectory Stop/Move Classification

The first limitation I identify for this chapter is that stop/move data is inferred from GPS trajectories, which are increasingly being processed in real-time, as they are collected in the field. However, POSMIT is a purely offline algorithm; therefore, one future direction I recommend for POSMIT is an extension to handle online data.

Another limitation of POSMIT is that it requires two fixed parameters to account for the presence of spatial noisiness in the data. Although I have provided estimation heuristics for h_d and h_i, ultimately, they are fixed, and this makes manually tuning the algorithm beyond estimation challenging —especially for h_i, as it impacts both the running time performance and effectiveness of the algorithm. Considering that spatial uncertainty in GPS trajectories varies as the tracked entity moves between more, or less, ideal recording environments, it would be preferable if POSMIT's weighting and sampling neighbourhood, h_d and h_i, somehow varied with the relative spatial uncertainty at a given entry. Of course, this raises a new challenge of correctly identifying the level of spatial uncertainty at each trajectory entry.

My final future direction for this chapter is targeted at the field overall, rather than POSMIT directly. Specifically, I recommend investigating a generalisation of the problem similar to the one I did in Chapter 3 for trajectory simplification. My reasoning is that my literature review in Section 2.4 seems to suggest that many existing works follow a similar paradigm. Thus, a generalisation of the problem seems plausible. I speculate that such a generalisation may allow one to combine POSMIT's probabilistic classification result with other types of approaches. For example, density-based stop/move detection schemes may use the distance from the stop cluster centroid as a metric to calculate the likelihood that a given entry

belongs to that stop. Such fusions of concepts may present further and more use-case specific stop/move detection approaches.

7.6.4 Trajectory Place-matching

The first limitation I identify for this chapter is that even though both stop and move episodes are formulated, I only used the stop episodes during semantic pattern mining. The move episodes are simply discarded. Thus, as a future extension of STOSEM, I recommend using an existing inference approach, such as the work of Widhalm et al. (2012), to label the move episodes based on their likely transport mode (walk, bike, car, bus etc.). This would provide an extra dimension of semantic data that would surely be beneficial to the final semantic pattern output.

Another limitation of this chapter is that I only evaluated my place-matching using synthetic data, and my overall mining approach through observations in a case study. Ideally, in future works, I recommend evaluating these using a real-world, ground-truth of spatio-temporal trajectories, annotated with visited places. The problem is that generating such a dataset would surely require human participants to keep travel diaries, which is tedious and error prone.

The final limitation of this chapter is that I only investigated mining frequent itemset and sequential patterns from the sequences of visited places. Recall, that in the case-study section of this chapter I discussed apparent behaviour profiles being suggested by the pattern output. I argue that even richer behaviour profiles may have emerged had I mined period patterns (Han et al., 1999). Periodic pattern mining could automatically detect recurring visitations to certain sequences of places, which may indicate a certain kind of participant. Thus, as a future research extension to STOSEM I recommend mining periodic trajectory patterns from the place visitation sequences.

Bibliography

Abul, Osman, Francesco Bonchi, and Mirco Nanni (2008). "Never Walk Alone: Uncertainty for Anonymity in Moving Objects Databases". In: *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*. ICDE '08. Washington, DC, USA: IEEE Computer Society, pp. 376–385. ISBN: 978-1-4244-1836-7. DOI: 10.1109/ICDE.2008.4497446. URL: http://dx.doi.org/10.1109/ICDE.2008.4497446.

Alencar, Lucas Andre de et al. (2015). "A Rule-based Method for Discovering Trajectory Profiles". In: *The 27th International Conference on Software Engineering and Knowledge Engineering, SEKE 2015, Wyndham Pittsburgh University Center, Pittsburgh, PA, USA, July 6-8, 2015*, pp. 244–249. DOI: 10.18293/SEKE2015-143. URL: https://doi.org/10.18293/SEKE2015-143.

Alexander, Ken (2014). "US GPS program and policy update". In: *26th SBAS International Working Group*, pp. 24–29.

Alexander, Lauren et al. (2015). "Origin–destination trips by purpose and time of day inferred from mobile phone data". In: *Transportation Research Part C: Emerging Technologies* 58. Big Data in Transportation and Traffic Engineering, pp. 240 –250. ISSN: 0968-090X. DOI: http://dx.doi.org/10.1016/j.trc.2015.02.018. URL: http://www.sciencedirect.com/science/article/pii/S0968090X1500073X.

Alvares, Luis Otavio et al. (2007). "A Model for Enriching Trajectories with Semantic Geographical Information". In: *Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems*. GIS '07. Seattle, Washington: ACM, 22:1–22:8. ISBN: 978-1-59593-914-2.

Amato, G. et al. (2018). "How Data Mining and Machine Learning Evolved from Relational Data Base to Data Science". In: *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years*. Ed. by Sergio Flesca et al. Cham: Springer International Publishing, pp. 287–306. ISBN: 978-3-319-61893-7. DOI: 10.1007/978-3-319-61893-7_17. URL: https://doi.org/10.1007/978-3-319-61893-7_17.

Andrienko, Natalia and Gennady Andrienko (Jan. 2013). "Visual Analytics of Movement: An Overview of Methods, Tools and Procedures". In: *Information Visualization* 12.1, pp. 3–24. ISSN: 1473-8716. DOI: 10.1177/1473871612457601. URL: http://dx.doi.org/10.1177/1473871612457601.

Ankerst, Mihael et al. (June 1999). "OPTICS: Ordering Points to Identify the Clustering Structure". In: *SIGMOD Rec.* 28.2, pp. 49–60. ISSN: 0163-5808. DOI: 10.1145/304181.304187. URL: http://doi.acm.org/10.1145/304181.304187.

Antunes, Cláudia and Arlindo L. Oliveira (2003a). "Generalization of Pattern-Growth Methods for Sequential Pattern Mining with Gap Constraints". In: *Machine Learning and Data Mining in Pattern Recognition: Third International Conference, MLDM 2003 Leipzig, Germany, July 5–7, 2003 Proceedings*. Ed. by

Petra Perner and Azriel Rosenfeld. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 239–251. ISBN: 978-3-540-45065-8. DOI: 10.1007/3-540-45065-3_21. URL: https://doi.org/10.1007/3-540-45065-3_21.

Antunes, Cláudia and Arlindo L. Oliveira (2003b). "Generalization of Pattern-Growth Methods for Sequential Pattern Mining with Gap Constraints". In: *Machine Learning and Data Mining in Pattern Recognition: Third International Conference, MLDM 2003 Leipzig, Germany, July 5–7, 2003 Proceedings*. Ed. by Petra Perner and Azriel Rosenfeld. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 239–251. ISBN: 978-3-540-45065-8. DOI: 10.1007/3-540-45065-3_21. URL: http://dx.doi.org/10.1007/3-540-45065-3_21.

Ashbrook, Daniel and Thad Starner (Oct. 2003). "Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users". In: *Personal Ubiquitous Comput.* 7.5, pp. 275–286. ISSN: 1617-4909. DOI: 10.1007/s00779-003-0240-0. URL: http://dx.doi.org/10.1007/s00779-003-0240-0.

Ashdown, J. D. et al. (Mar. 2013). "A full-duplex ultrasonic through-wall communication and power delivery system". In: *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* 60.3, pp. 587–595. ISSN: 0885-3010. DOI: 10.1109/TUFFC.2013.2600.

Atev, S., G. Miller, and N. P. Papanikolopoulos (Sept. 2010). "Clustering of Vehicle Trajectories". In: *IEEE Transactions on Intelligent Transportation Systems* 11.3, pp. 647–657. ISSN: 1524-9050. DOI: 10.1109/TITS.2010.2048101.

Ayres, Jay et al. (2002). "Sequential PAttern Mining Using a Bitmap Representation". In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '02. New York, NY, USA: ACM, pp. 429–435. ISBN: 1-58113-567-X. DOI: 10.1145/775047.775109. URL: http://doi.acm.org/10.1145/775047.775109.

Bao, Jie et al. (July 2015). "Recommendations in Location-based Social Networks: A Survey". In: *Geoinformatica* 19.3, pp. 525–565. ISSN: 1384-6175. DOI: 10.1007/s10707-014-0220-8. URL: http://dx.doi.org/10.1007/s10707-014-0220-8.

Beber, Marco Aurelio et al. (Nov. 2016). "Towards activity recognition in moving object trajectories from Twitter data". In: *XVII Brazilian Symposium on Geoinformatics - GeoInfo 2016*. Campos do Jordão, SP, Brazil, pp. 68–79. URL: http://urlib.net/8JMKD3MGPDW34P/3NDC4BB.

Bermingham, L., K. Lee, and I. Lee (Dec. 2014). "Spatio-Temporal Trajectory Region-of-Interest Mining Using Delaunay Triangulation". In: *2014 IEEE International Conference on Data Mining Workshop*, pp. 1–8. DOI: 10.1109/ICDMW.2014.47.

Bermingham, Luke and Ickjai Lee (2014). "Spatio-temporal Sequential Pattern Mining for Tourism Sciences". In: *Procedia Computer Science* 29.0. 2014 International Conference on Computational Science, pp. 379 –389. ISSN: 1877-0509. DOI: http://dx.doi.org/10.1016/j.procs.2014.05.034. URL: http://www.sciencedirect.com/science/article/pii/S1877050914002117.

Bogorny, Vania et al. (2014). "CONSTAnT – A Conceptual Data Model for Semantic Trajectories of Moving Objects". In: *Transactions in GIS* 18.1, pp. 66–88. ISSN: 1467-9671. DOI: 10.1111/tgis.12011. URL: http://dx.doi.org/10.1111/tgis.12011.

Boukhechba, Mehdi et al. (2015). "Online Recognition of People's Activities from Raw GPS Data: Semantic Trajectory Data Analysis". In: *Proceedings of the 8th ACM International Conference on PErvasive Technologies Related to Assistive Environments*. PETRA '15. Corfu, Greece: ACM, 40:1–40:8. ISBN: 978-1-4503-3452-5. DOI: 10.1145/2769493.2769498. URL: http://doi.acm.org/10.1145/2769493.2769498.

Brakatsoulas, Sotiris et al. (2005). "On Map-matching Vehicle Tracking Data". In: *Proceedings of the 31st International Conference on Very Large Data Bases*. VLDB '05. Trondheim, Norway: VLDB Endowment, pp. 853–864. ISBN: 1-59593-154-6. URL: http://dl.acm.org/citation.cfm?id=1083592.1083691.

Buchin, Kevin et al. (2011). "Finding long and similar parts of trajectories". In: *Computational Geometry* 44.9, pp. 465 –476. ISSN: 0925-7721. DOI: http://dx.doi.org/10.1016/j.comgeo.2011.05.004. URL: http://www.sciencedirect.com/science/article/pii/S0925772111000344.

Cao, Hu, Ouri Wolfson, and Goce Trajcevski (Apr. 2006). "Spatio-temporal Data Reduction With Deterministic Error Bounds". In: *The VLDB Journal* 15.3, pp. 211–228. ISSN: 1066-8888.

Cao, Huiping, Nikos Mamoulis, and David W. Cheung (2005). "Mining Frequent Spatio-Temporal Sequential Patterns". In: *Proceedings of the Fifth IEEE International Conference on Data Mining*. ICDM '05. Washington, DC, USA: IEEE Computer Society, pp. 82–89. ISBN: 0-7695-2278-5. DOI: 10.1109/ICDM.2005.95. URL: http://dx.doi.org/10.1109/ICDM.2005.95.

Cao, Xin, Gao Cong, and Christian S. Jensen (Sept. 2010). "Mining Significant Semantic Locations from GPS Data". In: *Proc. VLDB Endow.* 3.1-2, pp. 1009–1020. ISSN: 2150-8097. DOI: 10.14778/1920841.1920968. URL: http://dx.doi.org/10.14778/1920841.1920968.

Chen, Minjie, Mantao Xu, and P. Franti (May 2012). "A Fast O(N) Multiresolution Polygonal Approximation Algorithm for GPS Trajectory Simplification". In: *Image Processing, IEEE Transactions on* 21.5, pp. 2770–2785. ISSN: 1057-7149. DOI: 10.1109/TIP.2012.2186146.

Chen, Yukun et al. (2009). "Trajectory Simplification Method for Location-based Social Networking Services". In: *Proceedings of the 2009 International Workshop on Location Based Social Networks*. LBSN '09. Seattle, Washington: ACM, pp. 33–40. ISBN: 978-1-60558-860-5. DOI: 10.1145/1629890.1629898. URL: http://doi.acm.org/10.1145/1629890.1629898.

Chen, Zaiben, Heng Tao Shen, and Xiaofang Zhou (2011). "Discovering Popular Routes from Trajectories". In: *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*. ICDE '11. Washington, DC, USA: IEEE Computer Society, pp. 900–911. ISBN: 978-1-4244-8959-6. DOI: 10.1109/ICDE.2011.5767890. URL: http://dx.doi.org/10.1109/ICDE.2011.5767890.

Cortes, Corinna and Vladimir Vapnik (Sept. 1995). "Support-Vector Networks". In: *Mach. Learn.* 20.3, pp. 273–297. ISSN: 0885-6125. DOI: 10.1023/A:1022627411411. URL: https://doi.org/10.1023/A:1022627411411.

DATA.GOV.IE (2013). *Dublin Bus GPS sample data from Dublin City Council (Insight Project)*. [Online; accessed 12-November-2017]. URL: https://data.gov.ie/dataset/dublin-bus-gps-sample-data-from-dublin-city-council-insight-project.

Defence, Department of (2008). *Global Positioning System Standard Positioning Service Performance Standard*. 4th. Department of Defence. URL: http://www.gps.gov/technical/ps/2008-SPS-performance-standard.pdf.

Diggelen, Frank van and Per Enge (2015). "Proceedings of the 28th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2015)". In: *Proceedings of the 28th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2015)*. Vol. 63. ION, 361–369. URL: https://www.ion.org/publications/abstract.cfm?articleID=13079.

Diggelen, Frank Van (2007). "Accuracy-Lies, Damn Lies, and Statistics". In: *GPS world* 18.1, pp. 26–33.

Douglas, David H and Thomas K Peucker (1973). "Algorithms For The Reduction Of The Number Of Points Required To Represent A Digitized Line Or Its Caricature". In: *Cartographica: The International Journal for Geographic Information and Geovisualization* 10.2, pp. 112–122.

Ekdemir, Sadan (2011). " Efficient Implementation of Polyline Simplification for Large Datasets and Usability Evaluation". MA thesis. Uppsala University, Department of Information Technology.

Ester, Martin et al. (1996). "A density-based algorithm for discovering clusters in large spatial databases with noise". In: *KDD '96: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*. AAAI Press, pp. 226–231.

Exarchos, Themis P. et al. (2008). "Mining sequential patterns for protein fold recognition". In: *Journal of Biomedical Informatics* 41.1, pp. 165 –179. ISSN: 1532-0464.

Feng, Z. and Y. Zhu (Apr. 2016). "A Survey on Trajectory Data Mining: Techniques and Applications". In: *IEEE Access* 4, pp. 2056–2067. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2016.2553681.

Fournier-Viger, Philippe et al. (2014a). "Fast Vertical Mining of Sequential Patterns Using Co-occurrence Information". In: *Advances in Knowledge Discovery and Data Mining: 18th Pacific-Asia Conference, PAKDD 2014, Tainan, Taiwan, May 13-16, 2014. Proceedings, Part I*. Ed. by Vincent S. Tseng et al. Cham: Springer International Publishing, pp. 40–52. ISBN: 978-3-319-06608-0. DOI: 10.1007/978-3-319-06608-0_4.

Fournier-Viger, Philippe et al. (2014b). "SPMF: A Java Open-source Pattern Mining Library". In: *J. Mach. Learn. Res.* 15.1, pp. 3389–3393. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=2627435.2750353.

Fournier-Viger, Philippe et al. (2014c). "VMSP: Efficient Vertical Mining of Maximal Sequential Patterns". In: *Advances in Artificial Intelligence: 27th Canadian Conference on Artificial Intelligence, Canadian AI 2014, Montréal, QC, Canada, May 6-9, 2014. Proceedings*. Ed. by Marina Sokolova and Peter van Beek. Cham: Springer International Publishing, pp. 83–94. ISBN: 978-3-319-06483-3. DOI: 10.1007/978-3-319-06483-3_8.

Fournier-Viger, Philippe et al. (2017). "A Survey of Sequential Pattern Mining". In: *Data Science and Pattern Recognition* 1.1, pp. 54–77. ISSN: 2520-4165. URL: http://www.ikelab.net/dspr-pdf/vol1-1/dspr-paper5.pdf.

Fu, Zhongliang et al. (2016). "A Two-Step Clustering Approach to Extract Locations from Individual GPS Trajectory Data". In: *ISPRS International Journal of Geo-Information* 5.10. ISSN: 2220-9964. DOI: 10.3390/ijgi5100166. URL: http://www.mdpi.com/2220-9964/5/10/166.

Furletti, Barbara et al. (2013). "Inferring Human Activities from GPS Tracks". In: *Proceedings of the 2Nd ACM SIGKDD International Workshop on Urban Computing*. UrbComp '13. New York, NY, USA: ACM, 5:1–5:8. ISBN: 978-1-4503-2331-4. DOI: 10.1145/2505821.2505830. URL: http://doi.acm.org/10.1145/2505821.2505830.

Ghosh, Shreya and Soumya K. Ghosh (2017). "Modeling of Human Movement Behavioral Knowledge from GPS Traces for Categorizing Mobile Users". In: *Proceedings of the 26th International Conference on World Wide Web Companion*. WWW '17 Companion. Perth, Australia: International World Wide Web Conferences Steering Committee, pp. 51–58. ISBN: 978-1-4503-4914-7. DOI: 10.1145/3041021.3054150. URL: https://doi.org/10.1145/3041021.3054150.

Giannotti, Fosca et al. (2007). "Trajectory Pattern Mining". In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. San Jose, California, USA: ACM, pp. 330–339. ISBN: 978-1-59593-609-7.

Gidófalvi, Győző and Torben Bach Pedersen (Mar. 2009). "Mining Long, Sharable Patterns in Trajectories of Moving Objects". In: *GeoInformatica* 13.1, pp. 27–55. ISSN: 1573-7624. DOI: 10.1007/s10707-007-0042-z. URL: https://doi.org/10.1007/s10707-007-0042-z.

Gold, Christopher M. (1989). "Surface Interpolation". In: *Three Dimensional Applications in Geographic Information Systems*. London, England: Taylor and Francis, pp. 21–35.

Gomariz, Antonio et al. (2013). "ClaSP: An Efficient Algorithm for Mining Frequent Closed Sequences". In: *Advances in Knowledge Discovery and Data Mining: 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part I*. Ed. by Jian Pei et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 50–61. ISBN: 978-3-642-37453-1. DOI: 10.1007/978-3-642-37453-1_5. URL: https://doi.org/10.1007/978-3-642-37453-1_5.

Gong, Lei et al. (2015). "Identification of activity stop locations in GPS trajectories by density-based clustering method combined with support vector machines". In: *Journal of Modern Transportation* 23.3, pp. 202–213. ISSN: 2196-0577. DOI: 10.1007/s40534-015-0079-x. URL: http://dx.doi.org/10.1007/s40534-015-0079-x.

GPS.gov (2000). *Data From the First Week Without Selective Availability*. URL: http://www.gps.gov/systems/gps/modernization/sa/data/ (visited on 08/05/2017).

Grünwald, Peter (2005). "A Tutorial Introduction to the Minimum Description Length Principle". In: *Advances in Minimum Description Length: Theory and Applications*. MIT Press.

Gu, Qihang et al. (2017). "Inferring Venue Visits from GPS Trajectories". In: *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS '17. New York, NY, USA: ACM. ISBN: 978-1-4503-5490-5/17/11. DOI: 10.1145/3139958.3140034.

Gudmundsson, Joachim et al. (Nov. 2009). "Compressing spatio-temporal trajectories". In: *Computational Geometry* 42.9, pp. 825–841. ISSN: 09257721.

Haining, R.P. (2003). *Spatial Data Analysis: Theory and Practice*. Cambridge University Press. ISBN: 9780521774376. URL: https://books.google.com.au/books?id=CYZSh347eiAC.

Han, Jiawei, Guozhu Dong, and Yiwen Yin (1999). "Efficient Mining of Partial Periodic Patterns in Time Series Database". In: *Proceedings of the 15th*

International Conference on Data Engineering. ICDE '99. Washington, DC, USA: IEEE Computer Society, pp. 106–. ISBN: 0-7695-0071-4. URL: http://dl.acm.org/citation.cfm?id=846218.847205.

Han, Jiawei, Micheline Kamber, and Jian Pei (2011). *Data Mining: Concepts and Techniques*. 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN: 0123814790, 9780123814791.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2001). *The Elements of Statistical Learning*. Springer.

Hernández-Muñoz, José M. et al. (2011). "Smart Cities at the Forefront of the Future Internet". In: *The Future Internet: Future Internet Assembly 2011: Achievements and Technological Promises*. Ed. by John Domingue et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 447–462. ISBN: 978-3-642-20898-0. DOI: 10.1007/978-3-642-20898-0_32. URL: https://doi.org/10.1007/978-3-642-20898-0_32.

Herrera, Juan C. et al. (2010). "Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment". In: *Transportation Research Part C: Emerging Technologies* 18.4, pp. 568 –583. ISSN: 0968-090X. DOI: http://dx.doi.org/10.1016/j.trc.2009.10.006. URL: http://www.sciencedirect.com/science/article/pii/S0968090X09001430.

Hightower, Jeffrey (Oct. 2003). "From Position to Place". In: *Proceedings of The 2003 Workshop on Location-Aware Computing*. part of the 2003 Ubiquitous Computing Conference, pp. 10–12.

Hightower, Jeffrey and Gaetano Borriello (2004). "Particle Filters for Location Estimation in Ubiquitous Computing: A Case Study". In: *UbiComp 2004: Ubiquitous Computing: 6th International Conference, Nottingham, UK, September 7-10, 2004. Proceedings*. Ed. by Nigel Davies, Elizabeth D. Mynatt, and Itiro Siio. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 88–106. ISBN: 978-3-540-30119-6. DOI: 10.1007/978-3-540-30119-6_6. URL: https://doi.org/10.1007/978-3-540-30119-6_6.

Hightower, Jeffrey et al. (2005). "Learning and Recognizing the Places We Go". In: *Proceedings of the 7th International Conference on Ubiquitous Computing*. UbiComp'05. Berlin, Heidelberg: Springer-Verlag, pp. 159–176. ISBN: 978-3-540-28760-5. DOI: 10.1007/11551201_10. URL: http://dx.doi.org/10.1007/11551201_10.

Huang, Lian, Qingquan Li, and Yang Yue (2010). "Activity Identification from GPS Trajectories Using Spatial Temporal POIs' Attractiveness". In: *Proceedings of the 2Nd ACM SIGSPATIAL International Workshop on Location Based Social Networks*. LBSN '10. San Jose, California: ACM, pp. 27–30. ISBN: 978-1-4503-0434-4. DOI: 10.1145/1867699.1867704. URL: http://doi.acm.org/10.1145/1867699.1867704.

Hwang, Sungsoon, Christian Evans, and Timothy Hanke (2017). "Detecting Stop Episodes from GPS Trajectories with Gaps". In: *Seeing Cities Through Big Data: Research, Methods and Applications in Urban Informatics*. Cham: Springer International Publishing, pp. 427–439. ISBN: 978-3-319-40902-3. DOI: 10.1007/978-3-319-40902-3_23. URL: https://doi.org/10.1007/978-3-319-40902-3_23.

Jiang, S., J. Ferreira, and M. C. Gonzalez (June 2017). "Activity-Based Human Mobility Patterns Inferred from Mobile Phone Data: A Case Study of Singapore". In: *IEEE Transactions on Big Data* 3.2, pp. 208–219. DOI: 10.1109/TBDATA.2016.2631141.

Kang, Jong Hee et al. (2004). "Extracting Places from Traces of Locations". In: *Proceedings of the 2Nd ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*. WMASH '04. New York, NY, USA: ACM, pp. 110–118. ISBN: 1-58113-877-6. DOI: 10.1145/1024733.1024748. URL: http://doi.acm.org/10.1145/1024733.1024748.

Kang, Juyoung and Hwan-Seung Yong (2010). "Mining Spatio-Temporal Patterns in Trajectory Data". In: *Journal of Information Processing Systems* 6.4, pp. 521–536.

Kellaris, Georgios, Nikos Pelekis, and Yannis Theodoridis (2009). "Trajectory Compression Under Network Constraints". In: *Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases*. SSTD '09. Aalborg, Denmark: Springer-Verlag, pp. 392–398. ISBN: 978-3-642-02981-3. DOI: 10.1007/978-3-642-02982-0_27. URL: http://dx.doi.org/10.1007/978-3-642-02982-0_27.

Khetarpaul, Sonia et al. (2011). "Mining GPS Data to Determine Interesting Locations". In: *Proceedings of the 8th International Workshop on Information Integration on the Web: In Conjunction with WWW 2011*. IIWeb '11. Hyderabad, India: ACM, 8:1–8:6. ISBN: 978-1-4503-0620-1. DOI: 10.1145/1982624.1982632. URL: http://doi.acm.org/10.1145/1982624.1982632.

Kolesnikov, Alexander (2011). "Efficient Online Algorithms for the Polygonal Approximation of Trajectory Data". In: *Proceedings of the 2011 IEEE 12th International Conference on Mobile Data Management - Volume 01*. MDM '11. Washington, DC, USA: IEEE Computer Society, pp. 49–57. ISBN: 978-0-7695-4436-6. DOI: 10.1109/MDM.2011.53. URL: http://dx.doi.org/10.1109/MDM.2011.53.

Kong, Xiangjie et al. (2016). "Urban Traffic Congestion Estimation and Prediction Based on Floating Car Trajectory Data". In: *Future Gener. Comput. Syst.* 61.C, pp. 97–107. ISSN: 0167-739X. DOI: 10.1016/j.future.2015.11.013. URL: http://dx.doi.org/10.1016/j.future.2015.11.013.

Kumar, Ravi et al. (2015). "Driven by Food: Modeling Geographic Choice". In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. WSDM '15. Shanghai, China: ACM, pp. 213–222. ISBN: 978-1-4503-3317-7. DOI: 10.1145/2684822.2685300. URL: http://doi.acm.org/10.1145/2684822.2685300.

Lam, Hoang Thanh et al. (2014). "Mining Compressing Sequential Patterns". In: *Statistical Analysis and Data Mining* 7.1, pp. 34–52. ISSN: 1932-1872. DOI: 10.1002/sam.11192. URL: http://dx.doi.org/10.1002/sam.11192.

Lange, R. et al. (2009). "Remote real-time trajectory simplification". In: *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*. IEEE Computer Society, pp. 1–10. DOI: 10.1109/PERCOM.2009.4912767.

Lange, Ralph, Frank Dürr, and Kurt Rothermel (2008). "Online Trajectory Data Reduction Using Connection-preserving Dead Reckoning". In: *Proceedings of the 5th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Mobiquitous '08. Dublin, Ireland: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 52:1–52:10. ISBN: 978-963-9799-27-1.

Laube, Patrick (May 2015). "The Low Hanging Fruit is Gone: Achievements and Challenges of Computational Movement Analysis". In: *SIGSPATIAL Special* 7.1, pp. 3–10. ISSN: 1946-7729. DOI: 10.1145/2782759.2782762. URL: http://doi.acm.org/10.1145/2782759.2782762.

Lee, Anthony J. T., Yi-An Chen, and Weng-Chong Ip (June 2009). "Mining Frequent Trajectory Patterns in Spatial-temporal Databases". In: *Inf. Sci.* 179.13, pp. 2218–2231. ISSN: 0020-0255. DOI: 10.1016/j.ins.2009.02.016. URL: http://dx.doi.org/10.1016/j.ins.2009.02.016.

Lee, Jae-Gil, Jiawei Han, and Kyu-Young Whang (2007). "Trajectory Clustering: A Partition-and-group Framework". In: *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. SIGMOD '07. New York, NY, USA: ACM, pp. 593–604. ISBN: 978-1-59593-686-8. DOI: 10.1145/1247480.1247546. URL: http://doi.acm.org/10.1145/1247480.1247546.

Lee, L. et al. (Dec. 2016). "Comparison of Accuracy and Precision of GPS-Enabled Mobile Devices". In: *2016 IEEE International Conference on Computer and Information Technology (CIT)*, pp. 73–82. DOI: 10.1109/CIT.2016.94.

Lee, Wang-Chien and John Krumm (2011). "Trajectory Preprocessing". In: *Computing with Spatial Trajectories*. Ed. by Yu Zheng and Xiaofang Zhou. New York, NY, USA: Springer New York. Chap. 1, pp. 3–33. ISBN: 978-1-4614-1629-6. DOI: 10.1007/978-1-4614-1629-6_1.

Leung, Kenneth Wai-Ting, Dik Lun Lee, and Wang-Chien Lee (2011). "CLR: A Collaborative Location Recommendation Framework Based on Co-clustering". In: *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '11. Beijing, China: ACM, pp. 305–314. ISBN: 978-1-4503-0757-4. DOI: 10.1145/2009916.2009960. URL: http://doi.acm.org/10.1145/2009916.2009960.

Li, Chun and Jianyong Wang (2008). "Efficiently Mining Closed Subsequences with Gap Constraints". In: *Proceedings of the 2008 SIAM International Conference on Data Mining*, pp. 313–322. DOI: 10.1137/1.9781611972788.28. URL: http://epubs.siam.org/doi/abs/10.1137/1.9781611972788.28.

Li, Chun et al. (2012). "Efficient Mining of Gap-Constrained Subsequences and Its Various Applications". In: *ACM Trans. Knowl. Discov. Data* 6.1, 2:1–2:39. ISSN: 1556-4681. DOI: 10.1145/2133360.2133362. URL: http://doi.acm.org/10.1145/2133360.2133362.

Li, Xingxing et al. (June 2015). "Accuracy and reliability of multi-GNSS real-time precise positioning: GPS, GLONASS, BeiDou, and Galileo". In: *Journal of Geodesy* 89.6, pp. 607–635. ISSN: 1432-1394. DOI: 10.1007/s00190-015-0802-8. URL: https://doi.org/10.1007/s00190-015-0802-8.

Liao, Lin, Dieter Fox, and Henry Kautz (Jan. 2007). "Extracting Places and Activities from GPS Traces Using Hierarchical Conditional Random Fields". In: *Int. J. Rob. Res.* 26.1, pp. 119–134. ISSN: 0278-3649. DOI: 10.1177/0278364907073775. URL: http://dx.doi.org/10.1177/0278364907073775.

Lin, C. Y., C. C. Hung, and P. R. Lei (Nov. 2016). "A velocity-preserving trajectory simplification approach". In: *2016 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pp. 58–65. DOI: 10.1109/TAAI.2016.7880172.

Lin, Meng-Chang et al. (2015). "An ultrafast rechargeable aluminium-ion battery". In: *Nature* 520.7547, p. 325.

Lin, Nancy P et al. (2007). "Fast mining maximal sequential patterns". In: *Proceedings of the 7th International Conference on Simulation, Modeling and Optimization, September*, pp. 15–17.

Liu, Guangwen, Masayuki Iwai, and Kaoru Sezaki (2013). "An Online Method for Trajectory Simplification Under Uncertainty of GPS". In: *IPSJ Online Transactions* 6.July, pp. 65–74. ISSN: 1882-6660.

Liu, J. et al. (Apr. 2015). "Bounded Quadrant System: Error-bounded trajectory compression on the go". In: *2015 IEEE 31st International Conference on Data Engineering*, pp. 987–998. DOI: 10.1109/ICDE.2015.7113350.

Liu, Kuien et al. (2014). "Compressing Large Scale Urban Trajectory Data". In: *Proceedings of the Fourth International Workshop on Cloud Data and Platforms*. CloudDP '14. Amsterdam, The Netherlands: ACM, 3:1–3:6. ISBN: 978-1-4503-2714-5. DOI: 10.1145/2592784.2592787. URL: http://doi.acm.org/10.1145/2592784.2592787.

Long, Cheng, Raymond Chi-Wing Wong, and H. V. Jagadish (Aug. 2013). "Direction-preserving Trajectory Simplification". In: *Proc. VLDB Endow.* 6.10, pp. 949–960. ISSN: 2150-8097. DOI: 10.14778/2536206.2536221. URL: http://dx.doi.org/10.14778/2536206.2536221.

— (Sept. 2014). "Trajectory Simplification: On Minimizing the Direction-based Error". In: *Proc. VLDB Endow.* 8.1, pp. 49–60. ISSN: 2150-8097. DOI: 10.14778/2735461.2735466. URL: http://dx.doi.org/10.14778/2735461.2735466.

Lou, Yin et al. (2009). "Map-matching for Low-sampling-rate GPS Trajectories". In: *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS '09. New York, NY, USA: ACM, pp. 352–361. ISBN: 978-1-60558-649-6. DOI: 10.1145/1653771.1653820. URL: http://doi.acm.org/10.1145/1653771.1653820.

Luo, Ting et al. (2017). "An Improved DBSCAN Algorithm to Detect Stops in Individual Trajectories". In: *ISPRS International Journal of Geo-Information* 6.3. ISSN: 2220-9964. DOI: 10.3390/ijgi6030063. URL: http://www.mdpi.com/2220-9964/6/3/63.

Lv, Mingqi et al. (2016). "The Discovery of Personally Semantic Places Based on Trajectory Data Mining". In: *Neurocomput.* 173.P3, pp. 1142–1153. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2015.08.071. URL: https://doi.org/10.1016/j.neucom.2015.08.071.

M. J. Smith M. F. Goodchild, P. A. Longley (2015). *Geospatial Analysis: A Comprehensive Guide to Principles, Techniques and Software Tools*. 5th ed. The Winchelsea Press.

M. M. Fischer, A. Getis (2010). *Handbook of Applied Spatial Analysis: Software Tools, Methods and Applications*. Springer.

MacQueen, J. (1967). "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, pp. 281–297. URL: http://projecteuclid.org/euclid.bsmsp/1200512992.

Marino, Daniel L. and Milos Manic (2016). "Fast Trajectory Simplification Algorithm for Natural User Interfaces in Robot Programming by Demonstration". In: *CoRR* abs/1608.07338. eprint: 1608.07338. URL: http://arxiv.org/abs/1608.07338.

Mautz, Rainer (2009). "Overview of current indoor positioning systems". In: *Geodezija ir kartografija* 35.1, pp. 18–22.

Meratnia, Nirvana and Rolf A. de By (2004). "Spatiotemporal Compression Techniques for Moving Point Objects". In: *Advances in Database Technology - EDBT 2004: 9th International Conference on Extending Database Technology, Heraklion, Crete, Greece, March 14-18, 2004*. Ed. by Elisa Bertino et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 765–782. ISBN: 978-3-540-24741-8. DOI: 10.1007/978-3-540-24741-8_44. URL: https://doi.org/10.1007/978-3-540-24741-8_44.

Misra, P. and P. Enge (2011). *Global Positioning System: Signals, Measurements, and Performance*. Ganga-Jamuna Press. ISBN: 9780970954428. URL: https://books.google.com.au/books?id=5WJOywAACAAJ.

Monreale, Anna et al. (2009). "WhereNext: A Location Predictor on Trajectory Pattern Mining". In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '09. New York, NY, USA: ACM, pp. 637–646. ISBN: 978-1-60558-495-9. DOI: 10.1145/1557019.1557091. URL: http://doi.acm.org/10.1145/1557019.1557091.

Moreno, Francisco Javier et al. (2014). "SMoT+: extending the SMoT algorithm for discovering stops in nested sites". In: *Computing and Informatics* 33.2, pp. 327–342.

Morzy, Mikolaj (2007). "Mining Frequent Trajectories of Moving Objects for Location Prediction". In: *Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition*. MLDM '07. Leipzig, Germany: Springer-Verlag, pp. 667–680. ISBN: 978-3-540-73498-7. DOI: 10.1007/978-3-540-73499-4_50. URL: http://dx.doi.org/10.1007/978-3-540-73499-4_50.

Muckell, Jonathan et al. (2011). "SQUISH: An Online Approach for GPS Trajectory Compression". In: *Proceedings of the 2Nd International Conference on Computing for Geospatial Research &Amp; Applications*. COM.Geo '11. New York, NY, USA: ACM, 13:1–13:8. ISBN: 978-1-4503-0681-2. DOI: 10.1145/1999320.1999333. URL: http://doi.acm.org/10.1145/1999320.1999333.

Muckell, Jonathan et al. (July 2014). "Compression of trajectory data: a comprehensive evaluation and new approach". In: *GeoInformatica* 18.3, pp. 435–460. ISSN: 1573-7624. DOI: 10.1007/s10707-013-0184-0. URL: https://doi.org/10.1007/s10707-013-0184-0.

Nadaraya, Elizbar A (1964). "On estimating regression". In: *Theory of Probability & Its Applications* 9.1, pp. 141–142.

Nanni, Mirco and Dino Pedreschi (2006). "Time-focused Clustering of Trajectories of Moving Objects". In: *Journal of Intelligent Information Systems* 27.3, pp. 267–289.

Newson, Paul and John Krumm (2009). "Hidden Markov Map Matching Through Noise and Sparseness". In: *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS '09. New York, NY, USA: ACM, pp. 336–343. ISBN: 978-1-60558-649-6. DOI: 10.1145/1653771.1653818. URL: http://doi.acm.org/10.1145/1653771.1653818.

Oliveira, Ricardo, Maribel Yasmina Santos, and Joao Moura Pires (Dec. 2013). "4D+SNN: A Spatio-Temporal Density-Based Clustering Approach with 4D Similarity". In: *2013 IEEE 13th International Conference on Data Mining Workshops*, pp. 1045–1052. DOI: 10.1109/ICDMW.2013.119.

Orlando, Salvatore, Raffaele Perego, and Claudio Silvestri (2004). "A New Algorithm for Gap Constrained Sequence Mining". In: *Proceedings of the 2004 ACM Symposium on Applied Computing*. SAC '04. Nicosia, Cyprus: ACM, pp. 540–547. ISBN: 1-58113-812-1. DOI: 10.1145/967900.968014. URL: http://doi.acm.org/10.1145/967900.968014.

Palma, Andrey Tietbohl et al. (2008). "A Clustering-based Approach for Discovering Interesting Places in Trajectories". In: SAC '08, pp. 863–868. DOI: 10.1145/1363686.1363886. URL: http://doi.acm.org/10.1145/1363686.1363886.

Panagiotakis, Costas et al. (2012). "Segmentation and Sampling of Moving Object Trajectories Based on Representativeness". In: *IEEE Trans. on Knowl. and Data Eng.* 24.7, pp. 1328–1343. ISSN: 1041-4347. DOI: 10.1109/TKDE.2011.39. URL: http://dx.doi.org/10.1109/TKDE.2011.39.

Parent, Christine et al. (Aug. 2013). "Semantic Trajectories Modeling and Analysis". In: *ACM Comput. Surv.* 45.4, 42:1–42:32. ISSN: 0360-0300. DOI: 10.1145/2501654.2501656. URL: http://doi.acm.org/10.1145/2501654.2501656.

Parkka, J. et al. (Jan. 2006). "Activity Classification Using Realistic Data from Wearable Sensors". In: *Trans. Info. Tech. Biomed.* 10.1, pp. 119–128. ISSN: 1089-7771. DOI: 10.1109/TITB.2005.856863. URL: http://dx.doi.org/10.1109/TITB.2005.856863.

Patroumpas, K. (2013). "Online Tracking and Summarization over Streaming Maritime Trajectories". In: *In Proceedings of MOVE Workshop on Moving Objects at Sea*. Brest, France: University of Piraeus.

Pei, Jian, Jiawei Han, and Runying Mao (2000). "CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets." In: *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*. New York, NY, USA: ACM, pp. 21–30. URL: http://dblp.org/rec/conf/dmkd/PeiHM00.

Pei, Jian et al. (2004). "Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach". In: *IEEE Trans. on Knowl. and Data Eng.* 16.11, pp. 1424–1440. ISSN: 1041-4347. DOI: 10.1109/TKDE.2004.77. URL: http://dx.doi.org/10.1109/TKDE.2004.77.

Pelekis, Nikos et al. (2009). "Clustering Trajectories of Moving Objects in an Uncertain World". In: *Proceedings of the 9th IEEE International Conference on Data Mining*. IEEE Computer Society, pp. 417–427.

Pelekis, Nikos et al. (2011). "Clustering Uncertain Trajectories". In: *Knowl. Inf. Syst.* 28.1, pp. 117–147. ISSN: 0219-1377. DOI: 10.1007/s10115-010-0316-x. URL: http://dx.doi.org/10.1007/s10115-010-0316-x.

Pesyna, Kenneth M., Robert W. Heath, and Todd E. Humphreys (2014). "Centimeter positioning with a smartphone-quality GNSS antenna". In: *27th International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS 2014*. Vol. 2. United States: Institute of Navigation, pp. 1568–1577.

Pfoser, Dieter and Christian S. Jensen (1999). "Capturing the Uncertainty of Moving-Object Representations". In: *Proceedings of the 6th International Symposium on Advances in Spatial Databases*. SSD '99. London, UK, UK: Springer-Verlag, pp. 111–132. ISBN: 3-540-66247-2. URL: http://dl.acm.org/citation.cfm?id=647226.719082.

Potamias, Michalis, Kostas Patroumpas, and Timos Sellis (2006). "Sampling Trajectory Streams with Spatiotemporal Criteria". In: *Proceedings of the 18th International Conference on Scientific and Statistical Database Management*. SSDBM '06. Washington, DC, USA: IEEE Computer Society, pp. 275–284. ISBN: 0-7695-2590-3. DOI: 10.1109/SSDBM.2006.45. URL: http://dx.doi.org/10.1109/SSDBM.2006.45.

Radziemski, Leon and Inder Raj S. Makin (2016). "In vivo demonstration of ultrasound power delivery to charge implanted medical devices via acute and survival porcine studies". In: *Ultrasonics* 64, pp. 1 –9. ISSN: 0041-624X. DOI: http://dx.doi.org/10.1016/j.ultras.2015.07.012. URL: http://www.sciencedirect.com/science/article/pii/S0041624X15001973.

Raissi, C., P. Poncelet, and M. Teisseire (Sept. 2006). "SPEED : Mining Maxirnal Sequential Patterns over Data Strearns". In: *2006 3rd International IEEE Conference Intelligent Systems*, pp. 546–552. DOI: 10.1109/IS.2006.348478.

Renso, Chiara et al. (2013). "How you move reveals who you are: understanding human behavior by analyzing trajectory data". In: *Knowledge and Information*

Systems 37.2, pp. 331–362. ISSN: 0219-3116. DOI: 10.1007/s10115-012-0511-z. URL: https://doi.org/10.1007/s10115-012-0511-z.

Richter, Kai-Florian, Falko Schmid, and Patrick Laube (2012). "Semantic trajectory compression: Representing urban movement in a nutshell". In: *Journal of Spatial Information Science* 2012.4, pp. 3–30.

Rissanen, Jorma (1978). "Modeling by shortest data description". In: *Automatica* 14.5, pp. 465 –471. ISSN: 0005-1098. DOI: https://doi.org/10.1016/0005-1098(78)90005-5. URL: http://www.sciencedirect.com/science/article/pii/0005109878900055.

Rocha, J. A. M. R. et al. (July 2010). "DB-SMoT: A direction-based spatio-temporal clustering method". In: *2010 5th IEEE International Conference Intelligent Systems*, pp. 114–119. DOI: 10.1109/IS.2010.5548396.

Rodriguez, Alex and Alessandro Laio (2014). "Clustering by fast search and find of density peaks". In: *Science* 344.6191, pp. 1492–1496. ISSN: 0036-8075. DOI: 10.1126/science.1242072. eprint: http://science.sciencemag.org/content/344/6191/1492.full.pdf. URL: http://science.sciencemag.org/content/344/6191/1492.

Satopaa, Ville et al. (2011). "Finding a "Kneedle" in a Haystack: Detecting Knee Points in System Behavior". In: *Proceedings of the 2011 31st International Conference on Distributed Computing Systems Workshops*. ICDCSW '11. Washington, DC, USA: IEEE Computer Society, pp. 166–171. ISBN: 978-0-7695-4386-4. DOI: 10.1109/ICDCSW.2011.20. URL: http://dx.doi.org/10.1109/ICDCSW.2011.20.

Savage, Norma Saiph et al. (2010). "Frequent Trajectory Mining on GPS Data". In: *Proceedings of the 3rd International Workshop on Location and the Web*. LocWeb '10. Tokyo, Japan: ACM, 3:1–3:4. ISBN: 978-1-4503-0412-2. DOI: 10.1145/1899662.1899665. URL: http://doi.acm.org/10.1145/1899662.1899665.

Shaw, Arthur A. and N.P. Gopalan (2014). "Finding frequent trajectories by clustering and sequential pattern mining". In: *Journal of Traffic and Transportation Engineering (English Edition)* 1.6, pp. 393 –403. ISSN: 2095-7564. DOI: https://doi.org/10.1016/S2095-7564(15)30289-0. URL: http://www.sciencedirect.com/science/article/pii/S2095756415302890.

Shi, Wenzhong and ChuiKwan Cheung (2006). "Performance Evaluation of Line Simplification Algorithms for Vector Generalization". In: *The British Cartographic Society* 43.1, pp. 27–44.

Silva, T. L. C. d., K. Zeitouni, and J. A. F. d. Macêdo (June 2016). "Online Clustering of Trajectory Data Stream". In: *2016 17th IEEE International Conference on Mobile Data Management (MDM)*. Vol. 1, pp. 112–121. DOI: 10.1109/MDM.2016.28.

Song, Renchu et al. (May 2014). "PRESS: A Novel Framework of Trajectory Compression in Road Networks". In: *Proc. VLDB Endow.* 7.9, pp. 661–672. ISSN: 2150-8097. DOI: 10.14778/2732939.2732940. URL: http://dx.doi.org/10.14778/2732939.2732940.

Spaccapietra, Stefano et al. (2008). "A Conceptual View on Trajectories". In: *Data Knowl. Eng.* 65.1, pp. 126–146. ISSN: 0169-023X. DOI: 10.1016/j.datak.2007.10.008. URL: http://dx.doi.org/10.1016/j.datak.2007.10.008.

Spinsanti, Laura, Fabrizio Celli, and Chiara Renso (2010). "Where you stop is who you are: understanding people's activities by places visited". In: *BMI '10: Proceedings of the 5th BMI, Workshop on Behaviour Monitoring and Interpretation 2010*. Karlsruhe, Germany: CEUR-WS, pp. 38–52.

Srikant, Ramakrishnan and Rakesh Agrawal (1996). "Mining Sequential Patterns: Generalizations and Performance Improvements". In: *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology*. EDBT '96. London, UK, UK: Springer-Verlag, pp. 3–17. ISBN: 3-540-61057-X. URL: http://dl.acm.org/citation.cfm?id=645337.650382.

Takeuchi, Yuichiro and Masanori Sugimoto (2006). "CityVoyager: An Outdoor Recommendation System Based on User Location History". In: *Proceedings of the Third International Conference on Ubiquitous Intelligence and Computing*. UIC'06. Wuhan, China: Springer-Verlag, pp. 625–636. ISBN: 3-540-38091-4, 978-3-540-38091-7. DOI: 10.1007/11833529_64. URL: http://dx.doi.org/10.1007/11833529_64.

Tang, Yuxin et al. (2017). "Water-Soluble Sericin Protein Enabling Stable Solid–Electrolyte Interphase for Fast Charging High Voltage Battery Electrode". In: *Advanced Materials* 29.33, 1701828–n/a. ISSN: 1521-4095. DOI: 10.1002/adma.201701828. URL: http://dx.doi.org/10.1002/adma.201701828.

Tauberer, Joshua (2014). "Open Government Data". In: 2nd ed., pp. 1–20. URL: {https://opengovdata.io/}.

Teunissen, P. J. G. and A. Khodabandeh (Mar. 2015). "Review and principles of PPP-RTK methods". In: *Journal of Geodesy* 89.3, pp. 217–240. ISSN: 1432-1394. DOI: 10.1007/s00190-014-0771-3. URL: https://doi.org/10.1007/s00190-014-0771-3.

The White House (May 2000). *Statement By The President Regarding The United States' Decision To Stop Degrading Global Positioning System Accuracy*. U.S. Office of the Press Secretary (Producer). URL: https://www.navcen.uscg.gov/?pageName=gpsSelectiveAvailability.

Thierry, Benoit, Basile Chaix, and Yan Kestens (2013). "Detecting activity locations from raw GPS data: a novel kernel-based algorithm". In: *International journal of health geographics* 12.1, p. 14.

Tobler, W. R. (1970). "A Computer Movie Simulating Urban Growth in the Detroit Region". In: *Economic Geography* 46, pp. 234–240. ISSN: 00130095, 19448287.

Toole, Jameson L. et al. (2015). "The path most traveled: Travel demand estimation using big data resources". In: *Transportation Research Part C: Emerging Technologies* 58. Big Data in Transportation and Traffic Engineering, pp. 162 –177. ISSN: 0968-090X. DOI: http://dx.doi.org/10.1016/j.trc.2015.04.022. URL: http://www.sciencedirect.com/science/article/pii/S0968090X15001631.

Trajcevski, Goce (2011). "Uncertainty in Spatial Trajectories". In: *Computing with Spatial Trajectories*. New York, NY: Springer New York, pp. 63–107. ISBN: 978-1-4614-1629-6. DOI: 10.1007/978-1-4614-1629-6_3. URL: http://dx.doi.org/10.1007/978-1-4614-1629-6_3.

Trajcevski, Goce et al. (2004). "Managing uncertainty in moving objects databases". In: *ACM Transactions on Database Systems (TODS)* 29.3, pp. 463–507.

Tran, Le Hung et al. (2011). *Robust and Hierarchical Stop Discovery in Sparse and Diverse Trajectories*. Tech. rep. EPFL: EPFL.

Visvalingam, M. and J. D. Whyatt (1993). "Line Generalisation by Repeated Elimination of Points". In: *The Cartographic Journal* 30.1, pp. 46–51.

Viterbi, A. (1967). "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In: *IEEE Transactions on Information Theory* 13.2, pp. 260–269. ISSN: 0018-9448. DOI: 10.1109/TIT.1967.1054010.

Vreeken, Jilles, Matthijs Leeuwen, and Arno Siebes (July 2011). "Krimp: Mining Itemsets That Compress". In: *Data Min. Knowl. Discov.* 23.1, pp. 169–214. ISSN: 1384-5810. DOI: 10.1007/s10618-010-0202-x. URL: http://dx.doi.org/10.1007/s10618-010-0202-x.

Vrotsou, K. et al. (Jan. 2015). "SimpliFly: A Methodology for Simplification and Thematic Enhancement of Trajectories". In: *IEEE Transactions on Visualization and Computer Graphics* 21.1, pp. 107–121. ISSN: 1077-2626. DOI: 10.1109/TVCG.2014.2337333.

Wang, Jianyong and Jiawei Han (2004). "BIDE: Efficient Mining of Frequent Closed Sequences". In: *Proceedings of the 20th International Conference on Data Engineering*. ICDE '04. Washington, DC, USA: IEEE Computer Society, pp. 79–. ISBN: 0-7695-2065-0. URL: http://dl.acm.org/citation.cfm?id=977401.978142.

Wang, Yilun, Yu Zheng, and Yexiang Xue (2014). "Travel Time Estimation of a Path Using Sparse Trajectories". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '14. New York, NY, USA: ACM, pp. 25–34. ISBN: 978-1-4503-2956-9. DOI: 10.1145/2623330.2623656. URL: http://doi.acm.org/10.1145/2623330.2623656.

Wei, Hong et al. (2012). "Fast Viterbi Map Matching with Tunable Weight Functions". In: *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. SIGSPATIAL '12. New York, NY, USA: ACM, pp. 613–616. ISBN: 978-1-4503-1691-0. DOI: 10.1145/2424321.2424430. URL: http://doi.acm.org/10.1145/2424321.2424430.

Widhalm, P., P. Nitsche, and N. Brändie (Nov. 2012). "Transport mode detection with realistic Smartphone sensor data". In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pp. 573–576.

Xia, Hao et al. (2014). "Using smart phone sensors to detect transportation modes". In: *Sensors* 14.11, pp. 20843–20865.

Xiang, Longgang, Meng Gao, and Tao Wu (2016). "Extracting Stops from Noisy Trajectories: A Sequence Oriented Clustering Approach". In: *ISPRS International Journal of Geo-Information* 5.3. ISSN: 2220-9964. DOI: 10.3390/ijgi5030029. URL: http://www.mdpi.com/2220-9964/5/3/29.

Xie, Kexin, Ke Deng, and Xiaofang Zhou (2009). "From Trajectories to Activities: A Spatio-temporal Join Approach". In: *Proceedings of the 2009 International Workshop on Location Based Social Networks*. LBSN '09. Seattle, Washington: ACM, pp. 25–32. ISBN: 978-1-60558-860-5. DOI: 10.1145/1629890.1629897. URL: http://doi.acm.org/10.1145/1629890.1629897.

Yan, Xifeng, Jiawei Han, and Ramin Afshar (2003). "CloSpan: Mining: Closed Sequential Patterns in Large Datasets". In: *Proceedings of the 2003 SIAM International Conference on Data Mining*, pp. 166–177. DOI: 10.1137/1.9781611972733.15. URL: http://epubs.siam.org/doi/abs/10.1137/1.9781611972733.15.

Yan, Zhixian et al. (2013). "Semantic Trajectories: Mobility Data Computation and Annotation". In: *ACM Trans. Intell. Syst. Technol.* 4.3, 49:1–49:38. ISSN: 2157-6904. DOI: 10.1145/2483669.2483682. URL: http://doi.acm.org/10.1145/2483669.2483682.

Yang, Hanqing, Le Gruenwald, and Mathilda Boulanger (2013). "A Novel Real-time Framework for Extracting Patterns from Trajectory Data Streams". In: *Proceedings of the 4th ACM SIGSPATIAL International Workshop on GeoStreaming*. IWGS '13. New York, NY, USA: ACM, pp. 26–32. ISBN: 978-1-4503-2532-5. DOI: 10.1145/2534303.2534313. URL: http://doi.acm.org/10.1145/2534303.2534313.

Yang, Jiong and Meng Hu (2006). "TrajPattern: Mining Sequential Patterns from Imprecise Trajectories of Mobile Objects". In: *Advances in Database Technology - EDBT 2006: 10th International Conference on Extending Database Technology, Munich, Germany, March 26-31, 2006*. Ed. by Yannis Ioannidis et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 664–681. ISBN: 978-3-540-32961-9. DOI: 10.1007/11687238_40. URL: https://doi.org/10.1007/11687238_40.

Yang, Zhenglu, Yitong Wang, and Masaru Kitsuregawa (2007). "LAPIN: Effective Sequential Pattern Mining Algorithms by Last Position Induction for Dense Databases". In: *Proceedings of the 12th International Conference on Database Systems for Advanced Applications*. DASFAA'07. Berlin, Heidelberg: Springer-Verlag, pp. 1020–1023. ISBN: 978-3-540-71702-7. URL: http://dl.acm.org/citation.cfm?id=1783823.1783946.

Yin, Zhijun et al. (2011). "Diversified Trajectory Pattern Ranking in Geo-Tagged Social Media". In: *Proceedings of the 2011 SIAM International Conference on Data Mining*, pp. 980–991. DOI: 10.1137/1.9781611972818.84. eprint: http://epubs.siam.org/doi/pdf/10.1137/1.9781611972818.84. URL: http://epubs.siam.org/doi/abs/10.1137/1.9781611972818.84.

Ying, Josh Jia-Ching, Wang-Chien Lee, and Vincent S. Tseng (2014). "Mining Geographic-temporal-semantic Patterns in Trajectories for Location Prediction". In: *ACM Trans. Intell. Syst. Technol.* 5.1, 2:1–2:33. ISSN: 2157-6904. DOI: 10.1145/2542182.2542184. URL: http://doi.acm.org/10.1145/2542182.2542184.

Yuan, J. et al. (Jan. 2013). "T-Drive: Enhancing Driving Directions with Taxi Drivers' Intelligence". In: *IEEE Transactions on Knowledge and Data Engineering* 25.1, pp. 220–232. ISSN: 1041-4347. DOI: 10.1109/TKDE.2011.200.

Yuan, Jing et al. (2010). "T-drive: Driving Directions Based on Taxi Trajectories". In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS '10. New York, NY, USA: ACM, pp. 99–108. ISBN: 978-1-4503-0428-3. DOI: 10.1145/1869790.1869807. URL: http://doi.acm.org/10.1145/1869790.1869807.

Yuan, Jing et al. (2011). "Driving with Knowledge from the Physical World". In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '11. New York, NY, USA: ACM, pp. 316–324. ISBN: 978-1-4503-0813-7. DOI: 10.1145/2020408.2020462. URL: http://doi.acm.org/10.1145/2020408.2020462.

Yuan, Jing, Yu Zheng, and Xing Xie (2012). "Discovering Regions of Different Functions in a City Using Human Mobility and POIs". In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '12. Beijing, China: ACM, pp. 186–194. ISBN: 978-1-4503-1462-6. DOI: 10.1145/2339530.2339561. URL: http://doi.acm.org/10.1145/2339530.2339561.

Yuan, N. J. et al. (Mar. 2015). "Discovering Urban Functional Zones Using Latent Activity Trajectories". In: *IEEE Transactions on Knowledge and Data Engineering* 27.3, pp. 712–725. ISSN: 1041-4347. DOI: 10.1109/TKDE.2014.2345405.

Zaki, Mohammed J. (2001). "SPADE: An Efficient Algorithm for Mining Frequent Sequences". In: *Machine Learning* 42.1, pp. 31–60. ISSN: 1573-0565. DOI: 10.1023/A:1007652502315. URL: http://dx.doi.org/10.1023/A:1007652502315.

Zhang, Jingsong, Yinglin Wang, and Dingyu Yang (2015). "CCSpan: Mining closed contiguous sequential patterns". In: *Knowledge-Based Systems* 89, pp. 1 –13. ISSN: 0950-7051. DOI: http://dx.doi.org/10.1016/j.knosys.2015.06.014. URL: http://www.sciencedirect.com/science/article/pii/S0950705115002324.

Zheng, K. et al. (Aug. 2014a). "Online Discovery of Gathering Patterns over Trajectories". In: *IEEE Transactions on Knowledge and Data Engineering* 26.8, pp. 1974–1988. ISSN: 1041-4347. DOI: 10.1109/TKDE.2013.160.

Zheng, Kai et al. (2012). "Reducing Uncertainty of Low-Sampling-Rate Trajectories". In: *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*. ICDE '12. Washington, DC, USA: IEEE Computer Society, pp. 1144–1155. ISBN: 978-0-7695-4747-3. DOI: 10.1109/ICDE.2012.42. URL: http://dx.doi.org/10.1109/ICDE.2012.42.

Zheng, Yu (May 2015). "Trajectory Data Mining: An Overview". In: *ACM Trans. Intell. Syst. Technol.* 6.3, 29:1–29:41. ISSN: 2157-6904. DOI: 10.1145/2743025. URL: http://doi.acm.org/10.1145/2743025.

Zheng, Yu and Xing Xie (2011). "Learning Travel Recommendations from User-generated GPS Traces". In: *ACM Trans. Intell. Syst. Technol.* 2.1, 2:1–2:29. ISSN: 2157-6904. DOI: 10.1145/1889681.1889683. URL: http://doi.acm.org/10.1145/1889681.1889683.

Zheng, Yu and Xiaofang Zhou (2011). *Computing with Spatial Trajectories*. 1st. New York, NY, USA: Springer New York. ISBN: 978-1-4614-1628-9. DOI: 10.1007/978-1-4614-1629-6.

Zheng, Yu et al. (2008). "Understanding Mobility Based on GPS Data". In: *Proceedings of the 10th International Conference on Ubiquitous Computing*. UbiComp '08. Seoul, Korea: ACM, pp. 312–321. ISBN: 978-1-60558-136-1. DOI: 10.1145/1409635.1409677. URL: http://doi.acm.org/10.1145/1409635.1409677.

Zheng, Yu et al. (2009). "Mining Interesting Locations and Travel Sequences from GPS Trajectories". In: *Proceedings of the 18th International Conference on World Wide Web*. WWW '09. Madrid, Spain: ACM, pp. 791–800. ISBN: 978-1-60558-487-4. DOI: 10.1145/1526709.1526816. URL: http://doi.acm.org/10.1145/1526709.1526816.

Zheng, Yu, Xing Xie, and Wei-Ying Ma (2010). "GeoLife: A Collaborative Social Networking Service among User, location and trajectory". In: *IEEE Data(base) Engineering Bulletin*. URL: https://www.microsoft.com/en-us/research/publication/geolife-a-collaborative-social-networking-service-among-user-location-and-trajectory/.

Zheng, Yu et al. (2014b). "Urban Computing: Concepts, Methodologies, and Applications". In: *ACM Trans. Intell. Syst. Technol.* 5.3, 38:1–38:55. ISSN: 2157-6904. DOI: 10.1145/2629592. URL: http://doi.acm.org/10.1145/2629592.

Zhu, Xingquan and Xindong Wu (2007). "Mining Complex Patterns Across Sequences with Gap Requirements". In: *Proceedings of the 20th International Joint Conference on Artifical Intelligence*. IJCAI'07. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 2934–2940. URL: http://dl.acm.org/citation.cfm?id=1625275.1625748.

Zhu, Zack, Ulf Blanke, and Gerhard Tröster (Feb. 2016). "Recognizing Composite Daily Activities from Crowd-labelled Social Media Data". In: *Pervasive Mob. Comput.* 26.C, pp. 103–120. ISSN: 1574-1192. DOI: 10.1016/j.pmcj.2015.10.007. URL: http://dx.doi.org/10.1016/j.pmcj.2015.10.007.

Zimmermann, Max, Thomas Kirste, and Myra Spiliopoulou (2009). "Finding Stops in Error-Prone Trajectories of Moving Objects with Time-Based Clustering". In: *Intelligent Interactive Assistance and Mobile Multimedia Computing: International Conference, IMC 2009, Rostock-Warnemünde, Germany, November 9-11, 2009. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 275–286. ISBN: 978-3-642-10263-9. DOI: 10.1007/978-3-642-10263-9_24. URL: https://doi.org/10.1007/978-3-642-10263-9_24.