Date of acceptance Grade

Instructor

Mobility Modelling through Trajectory Decomposition and Prediction

Farbod Faghihi

Helsinki June 3, 2017

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Farbod Faghihi

Työn nimi — Arbetets titel — Title

Mobility Modelling throughTrajectory Decomposition and Prediction

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
	June 3, 2017	71 pages + 0 appendices

Tiivistelmä — Referat — Abstract

The ubiquity of mobile devices with positioning sensors make it possible to derive user's location at any time. However, constantly sensing the position in order to track the user's movement is not feasible, either due to the unavailability of sensors, or computational and storage burdens. In this thesis, we present and evaluate a novel approach for efficiently tracking user's movement trajectories using decomposition and prediction of trajectories. We facilitate tracking by taking advantage of regularity within the movement trajectories.

The evaluation of our approach is done using three large-scale spatio-temporal datasets, from three different cities: San Francisco, Porto, and Beijing. Two of these datasets contain only cab traces and one contains all modes of transportation. Therefore, our approach is solely dependent on the inherent regularity within the trajectories regardless of the city or transportation mode.

ACM Computing Classification System (CCS):
A.1 [Introductory and Survey],
I.7.m [Document and text processing]

Avainsanat — Nyckelord — Keywords
Trajectory Prediction, Trajectory Analysis, Human Mobility, Mobility Modelling

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

Human mobility modelling is the study of people's movement habits. It is used for the characterisation of movement patterns, for example, in transitions between different places or how the public transportation infrastructure is used. Understanding and characterising human mobility is a major scientific problem that has relevance to many scientific domains such as controlling epidemics [NW09, BGB11], analysis and forecasting of traffic [GLPC12, PZWS13, JWS08], urban planning [YZX12, ZLYX11, QLL+11], and numerous innovative mobile and network applications [SBBK+11, ZLH13, HNT13, RZL+12, SKJH06]

An important usage of mobility modelling is in context aware computing. In order to provide a richer human-computer interaction, context can be utilised by the computers. Context aware computing was first introduced by [ST94] as a computer program that changes its behaviour adaptively according to its usage location and people and objects around it. From then, context aware mobile applications have grown at a great rate. In the beginning, it was a fundamental part of ubiquitous and pervasive computing applications, but over time it became an important part of desktop applications, web applications, mobile computing, and nowadays, a part of the Internet of Things (IoT) [PZCG14]. One of the key characteristics to derive context is location-aware computing by modelling user's mobility [Gus02]. Mobility models have been the focus of different studies, and they cover different key areas. Deriving important places in each user's life by processing their movement traces [NB08], extracting the statistical characteristics of movement in terms of movement habit [SQBB10], and designing predictive models capable of predicting future movement behaviour of users [SMM+11] are a few of the examples in this area. These studies have predominantly focused on the movement between places; however, what is not known is how regular the movement between these locations are, i.e., how regular human movement trajectories are. In this thesis, our focus is on exploiting the existing regularity within movement trajectories which is mostly overlooked in the previous studies.

Existing research papers explore and analyse people's movement data to detect and extract their patterns and characteristics [SQBB10, GHB08]. However, the results reported in these studies are based on very high levels of abstraction and low resolution data; one of the main reasons for this matter is the type of datasets used in the studies. The most common data source for these studies is call data records (CDR). CDR are the data that mobile operators receive and log based on

their users' usage. For example, whenever users make phone calls, send messages, or change the cell-tower that they are connected to while moving the mobile operator will receive and log some data based on these actions. This data can be used to derive the user's location by multi-lateration of radio signals between cell towers. The accuracy of the obtained location is heavily affected by the number of available cell towers and their positions. In best case scenarios, the approach can only derive the location in block levels; hence, CDR data constrains the accuracy at which we can analyse mobility, and it is not possible to study the movement behaviours in higher level of detail. Therefore, these studies mostly analyse the movement in terms of the transitions between different areas. Figure 1 displays an example of this kind of mobility analysis.

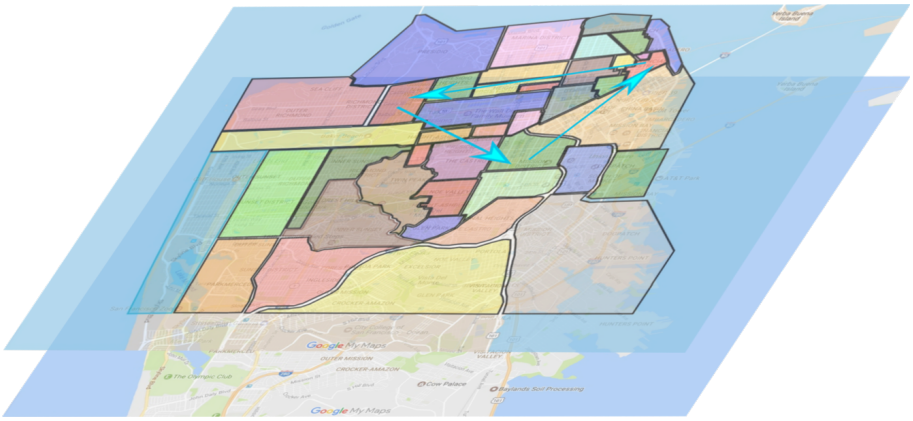

Figure 1: Precise analysis of movement trajectories is mostly overlooked by mobility modelling studies where their analysis is done in high levels of abstraction like the transition of the user in different areas of the city.

In addition, there are studies that take the movement data either with high accuracy like GPS traces or low accuracy like CDR location data as input and process them to extract the meaningful or distinct places for each user [LFK07, NB08, IBC+12]. These places are those that have some meaning in user's life like home, work place, gym, and favourite restaurant. After extracting these places, the movement between them is studied to create mobility models or analyse the predictability of transitions as illustrated in Figure 2.

In this work, we present a study on the regularity of route mobility. As mentioned, existing research has mostly overlooked the route mobility concept and does not

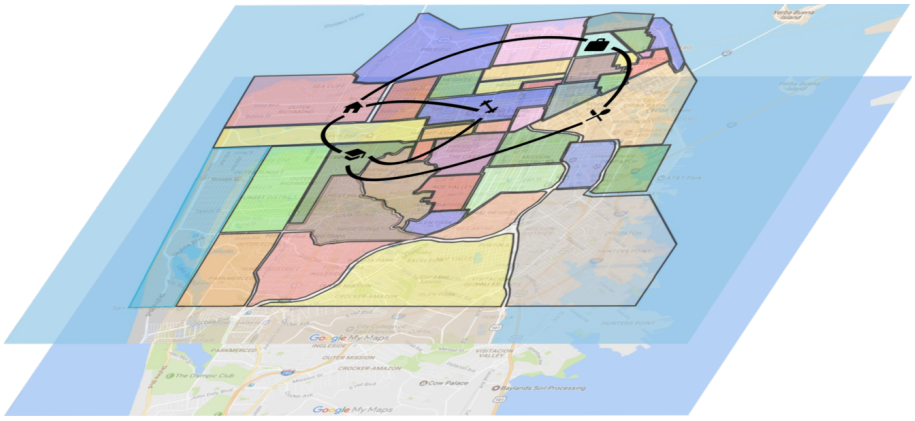

Figure 2: The movement data is processed first to extract the meaningful places for each person. The movement patterns between these places is studied afterwards.

give any information regarding the actual path taken by the user for transitioning from one location to another, therefore the actual movement trajectory between locations is missing as illustrated in Figure 3 (a). By studying the route mobility, we provide insights regarding the actual paths taken between any two points as shown in Figure 3 (b) and present how regular the movement between these points are. For this aim, we carry out our analysis using three different mobility datasets in three different cities that contain different modes of transportation. Cabspotting [PSDG09a, PSDG09b] is the first dataset, which contains the GPS traces of more than 500 cabs from a one-month period in San Francisco. GeoLife [ZZXM09, ZLC+08, ZXM10] is the second dataset that contains the GPS traces of 182 users that has been collected over a period of five years, and most of the traces are in the city of Beijing. Finally, the last dataset is the Taxi Service Trajectories [MMGF+13, MMGF+16], and it contains the GPS traces of more than 400 cabs from a 24 day interval.

Everyday human mobility in urban areas is constrained by the urban infrastructure and road network; therefore, making the cabs of the two datasets a feasible proxy for human mobility. It's worth mentioning that the datasets used in this work are freely available which assists in reproducibility of results. In contrast to the other studies that focus on the movement of each user or moving object, we analyse the movement trajectories in general which can be used for any user in the area of observations.

As our first technical contribution, we develop a principled methodology for mea-

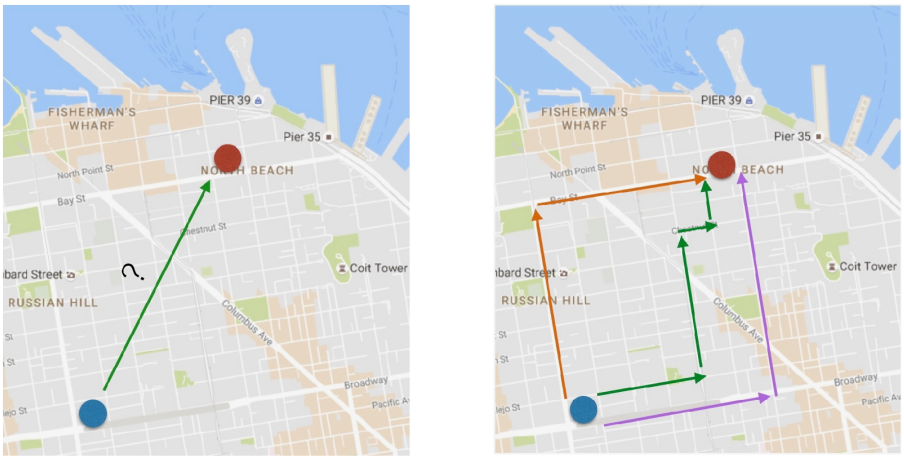

Figure 3: (a) The source and destination of a trip is known and can be used for studying the transition between places, but the information regarding the actual path taken between these two locations is missing. (b) Shows three different possible paths that a moving object can use to reach its destination.

suring the regularity of movement trajectories by computing the entropy of all trajectories for any given spatial and temporal resolution. For this aim, we convert the trajectories into Markov chains and quantify their regularity using Markov trajectory entropy [EC93]. Using this method, we analyse and quantify the overall regularity of trajectories and demonstrate that, as a whole, trajectories appear irregular and hard to predict. Consequently, we reveal that only a subset of points along the trajectories are responsible for irregularity within the trajectory. We extract and call these points fork-points. By representing the trajectories with their fork-points, we can show a higher predictability for them. Motivated by this result, as the second contribution of this thesis, we introduce a segmentation algorithm that segments the trajectories into more regular sub-segments. The segmentation can be used as an alternative compressed representation of the trajectories. Finally, we introduce a predictive framework capable of predicting moving objects trajectories.

2 Related Work

An extensive part of our work explores the human movement traces and analyses the inherent degree of regularity present within them. We categorise previous studies into three groups: human mobility, trajectory analysis, and energy efficiency. We first introduce the related work done in the area of human mobility which covers general characteristics of human movement. Next, we give an overview of trajectory analysis studies as our system relies on such techniques to provide predictions regarding the destination and rest of the trajectory. Finally, due to the applicability of trajectory processing approaches in systems that provide energy efficient tracking, we mention the important works done in this area. In this section we provide a brief summary of the work done in the mentioned areas and discuss how our work differentiates from them.

2.1 Human Mobility

Human mobility patterns and the amount of regularity within them are studied in numerous scientific studies. These studies can be categorised either as mobility displacement modelling or characterisation of regularity within the movement patterns. Here we introduce the works done in each category.

In terms of the movement displacement, several key attributes have been discovered in different studies. It is shown that even though human motion is not a random walk, it can be modelled with Lévy flight [BHG06, GHB08, RSH+11]. According to the Lévy flight model, human movement consists of straight line trips with no pause or directional change which are known as flights. The length or distance of these flights follow a truncated power-law distribution – meaning that these travelling distances decay as a power-law. In a similar way, it also means that these travels are mainly consisted of many small flights interleaved with longer flights. In another attempt [ZMH+15] Zhao et al. explains the shared features of human walks using Lévy walks by decomposing the movement patterns based on their transportation mode. Therefore, by having information regarding transportation mode, the Lévy flights can be broken into a mixture of log-normal distributions, while the flights in each mode contain power-law distributed jump sizes. Furthermore, other studies [RSH+11, KKK06] have shown that in addition to the flights, the pause times of human walks demonstrate a truncated power-law distribution as well. Another important characteristic of human motion as demonstrated in [GHB08] shows that

people will only move within a confined area; however, this confined area differs among people.

There are works that simulate individuals' movement through human mobility models [LHK+09, LHK+12, MCN11, SKWB10]. These studies exploit human walking traces and significant patterns in human mobility to create systems that are able to simulate human movement as realistically as possible. These systems exploit the identified features to simulate the human movement inside communities and areas such as university campus or companies. For pedestrian specific simulations, Blue et al. [BA01] uses Cellular automata microsimulation for simulating the pedestrians movement in discrete cells of 0.21 m^2. Yoon et al. [YNLK06] introduces a framework for statistical mobility model generation. To do this, they make use of coarse-grained wi-fi traces in a campus area.

Besides mobility displacement modelling, there have been studies investigating the regularity within human movement patterns, mainly the transitions between frequently visited locations. Song et al. [SQBB10] uses a dataset of call data records (CDR) to generate movement trajectories and investigate their predictability. The study reports an upper limit of 93% for predictability of human movement. However, the user's location in CDR data is a low resolution approximation; and hence, it contains a large amount of uncertainty. In a similar way, Lin et al. [LHL12] uses GPS data in order to have location data of higher resolution compared to call data records. The paper, used a grid model to provide a building level granularity and reported a 90% predictability for mobility. There are two main issues regarding these two papers. First, the mobility patterns are constructed with a fixed sampling rate. As a result, if a user visits multiple locations during this sampling interval, only one place will be represented for that time, which will lead in an incomplete reconstruction of user's visitations. The other issue arises from the fact that the length of the location visitation sequence is determined by the sampling rate which affects entropy. Lu et al. [LWB+13] analyses movement patterns of individuals in Cote d'Ivoire to measure regularity. Similarly, the authors reported an upper bound of 88% for predictability. In addition, Lu et al. shows that the mentioned limit can be reached by implementing Markov Chain based predictors.

While the previous studies analysed the data in high levels of abstraction, Smith et al. [SWGB14] explores the predictability of human movement by taking real-world topological constraints into account; therefore, reporting a more realistic and tighter upper bound. The study analyses the effect of sampling rate and spatial

resolution on the predictability and reports that the predictability upper bound is 11% to 24% lower than what previous studies have claimed. As a solution, Smith et al. suggests the integration of external data sources to improve the predictions based on the application requirements. De Domenico et al. [DDLM13] reveals a relationship between social interactions and mobility. The study combines the data regarding user's social interactions with movement data and improves the accuracy of user's position predictions.

Li et al. [LJH+14] analyse movement predictability in terms of vehicular mobility. For this aim, they focus on predicting the next most likely major intersection along a route. For deriving a predictability limit, entropy theory is used. The research reports the predictability of vehicular mobility to be larger than 78%; therefore, showing presence of a strong regularity in the movement patterns.

This thesis provides the first ever study of regularity of route mobility. We demonstrate that movement trajectories can be decomposed into sub-segments with high predictability, despite using a very small set of points along the route for predicting the route accurately. Compared to the previous studies, we use trajectories in high level of details.

2.2 Trajectory Analysis

A trajectory captures a moving object's path through space as a function of time. Hence, a trajectory represents the path taken by a moving object and is an important piece for modelling and analysing its movement. Even when trajectory information is missing, studies make assumptions regarding trajectories or try to approximate them. For example, Zheng et al. [ZHL06] studies the effect of social activities on users' whereabouts. They combine social attributes with geographical data to derive their location based on a survey data. Since the survey data contains no trajectory information, Dijkstra's shortest path algorithm was used for recreating the paths. But not all studies use the information regarding the actual trajectories. These studies mostly use the trajectory data only as an input data to derive other information such as places that each user visits. In [LZX+08], Li et al. uses the GPS traces of 65 users over 6 months in order to provide a similarity measure to match more alike users with each other based on their movement behaviour. They introduce a framework named hierarchical-graph-based similarity measurement (HGSM) that is based on user's location visitation history. Therefore, the GPS trajectories are only used to detect user's distinct location visitations.

One of the first works to focus on trajectory tracking was proposed by Lange et al. [LFDR09]. The study tracks and processes the actual trajectory of a moving object and simplifies it according to a certain accuracy bound to provide an approximation. Liao et al. [LPFK07] uses GPS traces to capture regularity within user's movement behaviour. Accordingly, their model is able to detect undefined or irregular behaviour within mobility patterns of a user. Their system is built upon a hierarchical Markov model that uses Rao-Blackwellized particle filters at different hierarchical levels to provide a more efficient inference for user's destination and mode of transportation. Our model can capture the regularity of all user's in an geographical area and we are able to show why some movement might be irregular.

In a series of studies, Krumm et al. analyse GPS traces to create predictive mobility models. In [Kru06, KH06] destination prediction based on a probabilistic approach is presented with the intuition that drivers choose efficient routes. In a similar approach they use a grid representation of the world for discretisation of trajectories. What differentiates our work from these studies is that we analyse the human movement trajectories in a more general level. We make no claim regarding the users' mode of transportation as we aggregate all movement trajectories of all users together. In addition, our model is able to predict the path in addition to destinations. In [Kru10] the author uses a dataset of driver's turning decisions at intersections to predict the driver's route. This approach can be used to predict the trajectory, but it can only provide prediction at the intersection for the next segment of the trip and not the whole trajectory.

2.3 Energy Efficiency

Tracking user's movement and extracting his position in a continuous matter and with high accuracy using mobile sensors such as GPS sensor can be costly action in terms of energy. Therefore, providing energy-efficient tracking for mobile devices requires exploring and analysing of movement of users and their corresponding trajectories. For this reason, we have decided to introduce a selection of these studies that are related to mobility in this section.

Most of the first works on energy-efficient mobility monitoring focused on optimising energy consumption by reducing the sampling rate of GPS receivers. We can categorise these into two groups. The first group depends on the fact that the accuracy requirement of localisation varies. Moreover the reduction of sampling is done by either using less expensive alternative localisation methods such

as Wi-Fi and cell-tower triangulation – whenever it is allowed by the accuracy requirements [LKLZ10, PKG10] – or by exploiting other external information that can provide information regarding the movement. SensTrack [ZLJG13] system uses the information regarding the acceleration and direction derived from the less expensive sensors to trigger GPS sampling. Whenever a user moves indoors, he uses WiFi localisation. The study is related to our work where it uses a Gaussian process regression method to reconstruct the trajectories from partial samples. The stored samples are similar to the fork-points that we use to decompose the trajectory, but unlike our work they are not related to the regularity of movement. Fang et al. [FZ11] exploits the information regarding constraints and limitations on daily movement. The authors present EnAcq, a trajectory data acquisition scheme that utilises the information regarding the road networks with map matching techniques to improve energy consumption. In order to reduce energy usage, Nodari et al. [NNS15] model movement trajectories. The approach samples and communicates the location only when the moving object deviates from predicted trajectory.

The other group of studies that are relevant to our work, utilise line simplification methods, such as Douglas Peucker [DP73], to simplify the actual trajectory with an approximation. The points in the compressed trajectory do have similarities to our detected fork-points that can be used to compress the trajectories as well. Lange et al. [LDR11] use line simplification techniques to reduce the sampling rate and at the same time use dead reckoning to select communication intervals. Constandache et al. [CGS+09] introduce EnLoc, an energy efficient localisation framework. EnLoc exploits the regularity of users' movement. By using the habitual characteristic of mobility, it predicts the population's location to provide an offline optimal solution for energy efficient localisation. The EnLoc system schedules location updates at so-called uncertainty points, which roughly correspond to fork-points in the context of our work. Entracked [KBBN11, BBKN15] is another system for trajectory tracking. In EnTracked, application designers provide a desired degree of accuracy for position and/or trajectory tracking accuracy and the system determines the optimal strategy to achieve the desired accuracy with the least amount of energy. The system design of EnTracked combines intelligent sensor management strategies with trajectory simplification techniques and intelligent uploading policies to preserve energy.

It is worth mentioning that there are also other approaches that provide optimal approaches without the reduction of sampling. For example, instead of adapting GPS duty cycle, Ramos et al. [RZL+11] propose the LEAP system which saves energy by offloading computationally intensive parts of the GPS processing pipeline

to a server or cloud platform. We exclude detailed discussion about these works as they do not take advantage of human mobility analysis for energy reduction.

3 Data Collection

We analyse regularity in route mobility using three different datasets that contain the movement trajectories of moving objects. Spatial trajectories contain rich information that can be utilised in different applicable areas, but because of the way they are captured and stored they contain may contain noise and inaccuracies. Therefore, they should be first processed according to our needs.

We have performed our analysis using three different datasets. Cabspotting [PSDG09a, PSDG09b], which contains GPS traces of more than 500 cabs from a one-month period in San Francisco; GeoLife [ZZXM09, ZLC+08, ZXM10] contains the GPS traces of 182 users that has been collected over a period of five years, and most of the traces are in the city of Beijing; and finally, Taxi Service Trajectories [MMGF+13, MMGF+16], which contains the GPS traces of more than 400 cabs from a 24 day interval.

In this chapter we cover the concepts that enable us to process our spatial data and making it ready for our computations. We first summarise the factors that are intertwined with collecting spatial data. Next, as in our case the data is already collected, the majority of this section describes the techniques required for processing the spatial trajectories to be transformed to a clean and ready-shape version for our analysis. Finally, we close this chapter by demonstrating the datasets after the post-processing steps.

3.1 Trajectory Preprocessing

To record the continuous movement of an object we need to sample it in the form of discrete points. This will lead into storing an approximation of the continuous movement. The higher the sampling rate, The higher the accuracy of the original movement in the samples.

Even though more accurate representation of the movement is desired while working with trajectories, there are certain factors that prevent us from collecting the most accurate trajectories. For example, the higher the sampling rate the larger the number of measurements will be, which can lead into massive datasets. In addition, higher sampling rates also require more network communication to transmit the samples. Therefore, storage space and network communication capacity are two of these constraints. To overcome this issue, simplification methods are proposed to

save storage and communication costs and remove redundant data.

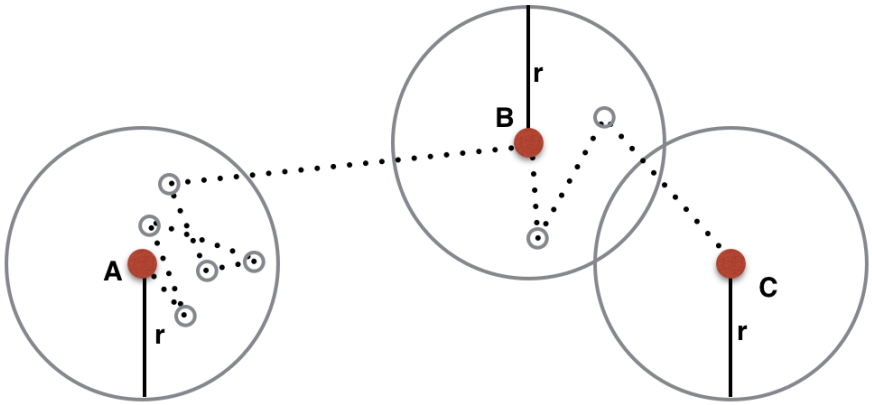

Figure 4: Point policy is an update policy where an application specific error distance (r) is selected. From each sampled location, a new sample is only recorded when the object moves further than the error threshold from its previous sampled location.

Gathering more samples puts a computational burden on the devices leading into higher consumption of energy. Reading continuously from energy expensive tracking sensors like GPS can deplete energy much faster. To address this issue, the research community has proposed several solutions. These studies integrate methods such as dead reckoning to estimate the future position of the object based on its previously known location, direction, speed, and time. This information can be extracted from available tracking sensors such as accelerometers and gyroscopes. By using this approach it is possible to reduce the usage of energy hungry sensors like GPS. In addition, update schemes are used to decide when to store or communicate the location of a device. Point policy [CJNP04] is an example of a simple update policy. As illustrated in Figure 4, point policy defines an error threshold and only stores or reports the position when the user moves beyond the defined error threshold. Therefore, an application specific error threshold can be selected for the point policy to store a sample of the movement. There are advanced methods such as the Douglas-Peucker algorithm [DP73] that can be used for approximating the original trajectory. More advanced works in the area of energy efficient tracking have already been discussed in Section 2.

An important factor is the inherent error present in the tracking devices. For example, when using call data records (CDR) to track the user, the derived locations contain a high amount of error. Thus, it is not meaningful to capture the location for every few seconds. GPS sensors, on the other hand, have an average error of 10 meters. At times during the sampling of movement, the observed position may have much higher error which is not in line with other samples leading to the creation of outliers in the captured data. Due to the captured noise, the trajectories are never perfectly accurate. To solve this issue, it is possible to use a variety of filtering techniques. In order to familiarise the reader with this issue we provide an example trajectory and describe two simple filtering methods to resolve the noise. The first simple method to remove the noise is to use a mean filter. Mean filtering works as sliding window that covers the last n temporally adjacent values of the last observation z. Therefore, calculating the mean of the last n observations. Equation 1 describes the mean filtering.

$$\hat{x}_i = \frac{1}{n}\Sigma_{j=i-n+1}^{i} z_j \tag{1}$$

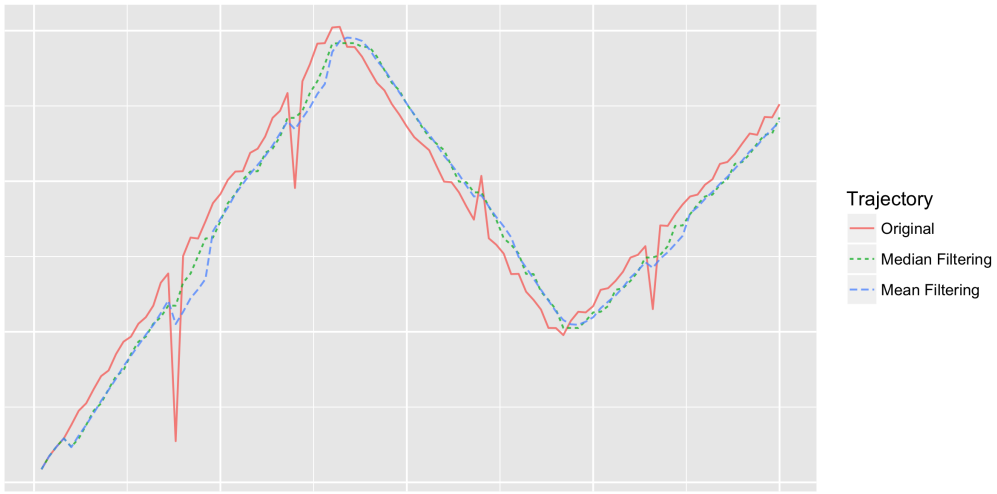

Figure 5: The captured noise in trajectories can be removed by using the filtering methods. Both median and mean filtering introduce lag to the original trajectory; however, unlike mean filtering, median filtering is not sensitive to outliers.

Two downsides of the mean filtering is that it introduces some lag to the original trajectory, and its sensitivity to outliers. Figure 5 displays a trajectory before and after mean filtering with a window of size five is applied to it.

The second method that we describe here is called median filtering and it can be used to mitigate the mean filters sensitivity to the outliers. Median filtering, as described in Equation 2, is similar to the mean filtering. The only difference is that we take the median of the last n observation instead of the mean:

$$\hat{x}_i = median\left\{z_{i-n+1}, z_{i-n+2}, ..., z_{i-1}, z_i\right\} \tag{2}$$

Both techniques have the disadvantage of adding lag to the original trajectory. The aim of introducing filtering in this section is to only help the reader to understand the characteristics of real world data better. For more advanced filtering techniques, we invite the readers to look into Kalman filtering [Gel74] and Particle filtering [DDFG01].

3.2 Cleaning Datasets: Postprocessing

In every data collection there are different factors that can affect the quality of the captured data. These factors can be related to the accuracy of the sensors or the approaches used for the study. Based on available resources and tracking devices, their energy capacities, and their average error level, a meaningful sampling rate is chosen. Thus, there is a trade-off between the accuracy of the captured trajectory and the available resources. In addition, different applications may need data with certain standards which may be different than the original application that led into the data collection study.

In our work we use three different datasets for our experiments, and we have to process and clean these datasets to meet our desirable level of quality. In this section, we discuss the methods needed to process and format the datasets according to our needs. We first discuss the general principles for cleaning the datasets and provide specific thresholds values used by cleaning methods for each dataset in Section 3.2.4. This will enable our readers to follow the steps we took to produce the results.

3.2.1 Velocity pruning

In analysing the regularity of route mobility, it is important for the data to have sufficient degree of accuracy so that it would provide a good approximation of how people move. One attribute that we can use to see if the trajectory is faulty is to check the velocity between any two consecutive observations. After computing the

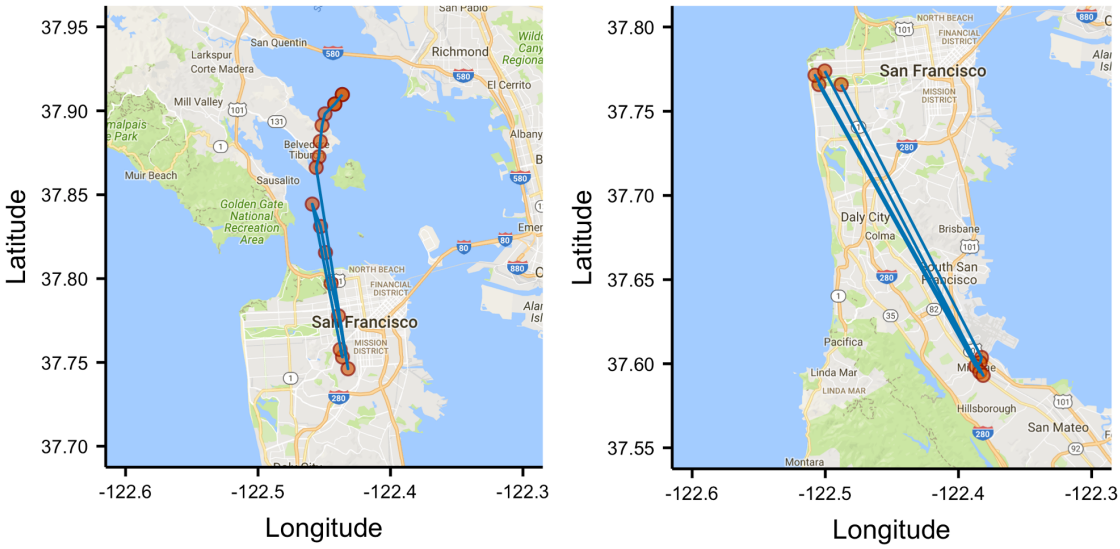

Figure 6: Two sampled trajectories from the Cabspotting dataset is shown. It is evident that both of the trajectories have unrealistic segments that look like very long jumps which are faulty. The trajectories that contain segments with unrealistic speed values are removed.

velocity, we remove any trajectory that contains a pair of observations with unrealistic speed. The removal of segments where speed is not within the possible velocity range of that metropolitan area is called velocity pruning. Therefore, all trajectories that contain a segment with an unacceptable speed value are removed from the dataset. Figure 6 illustrates two examples to provide a better understanding regarding how the velocities above the maximum range may affect the trajectories.

3.2.2 Trip Merging and Pruning

Another characteristic that can be used to remove faulty trajectories is the distance between the source and destination of a trip. There are trips where the distance is really small for the trip to be meaningful, which can correspond to either erroneous observations or cases where the user did not start any trips but was recording some observations. Figure 7 provides two examples of such trips. We remove any trip that has an overall distance smaller than a threshold.

Similar to the overall trip distance, trip duration is another factor that can give us

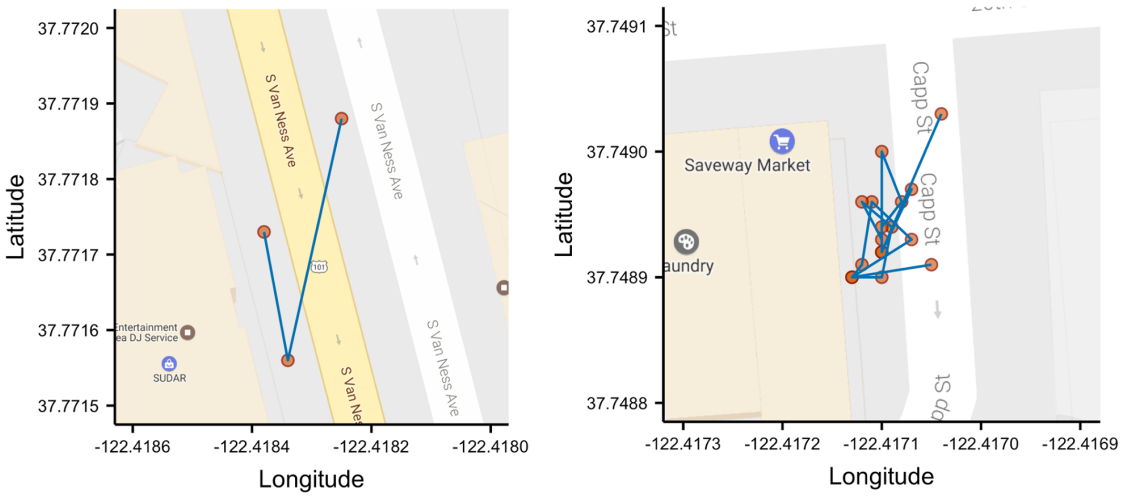

Figure 7: Two sampled trajectories from the Cabspotting dataset is shown. The distance between the source and destination of some trips is too small to be a representation of an actual trip. These cases are removed from the dataset.

information regarding the trip. For example, a trip that lasted for 10 seconds is not meaningful to be counted as a real trip. Any duration for trips that are not within a selected range is removed.

3.2.3 Distance and Update Rate Between Consecutive Samples

As already mentioned, each spatial trajectory in the dataset is made of discrete data-points. Therefore, there is a gap between any two consecutive data-points. Based on the size of these gaps there are three different cases. First, if the size of the gaps is small the quality of the trajectory is at a desirable level; thus, nothing needs to be done. Second, the length of the gaps in some cases are bigger than what is required for our application but still reasonable. In such cases we can reduce the length of these gaps, and reconstruct the original path using linear interpolation, which is discussed later. Finally, in some cases the gap between two consecutive observations is too large to be meaningfully filled by linear interpolation. Figure 8 illustrates two instances of such cases. Trips that contain these non-useful segments are removed as part of the cleaning pipeline.

Similarly to the distance between consecutive observations, the update interval be-

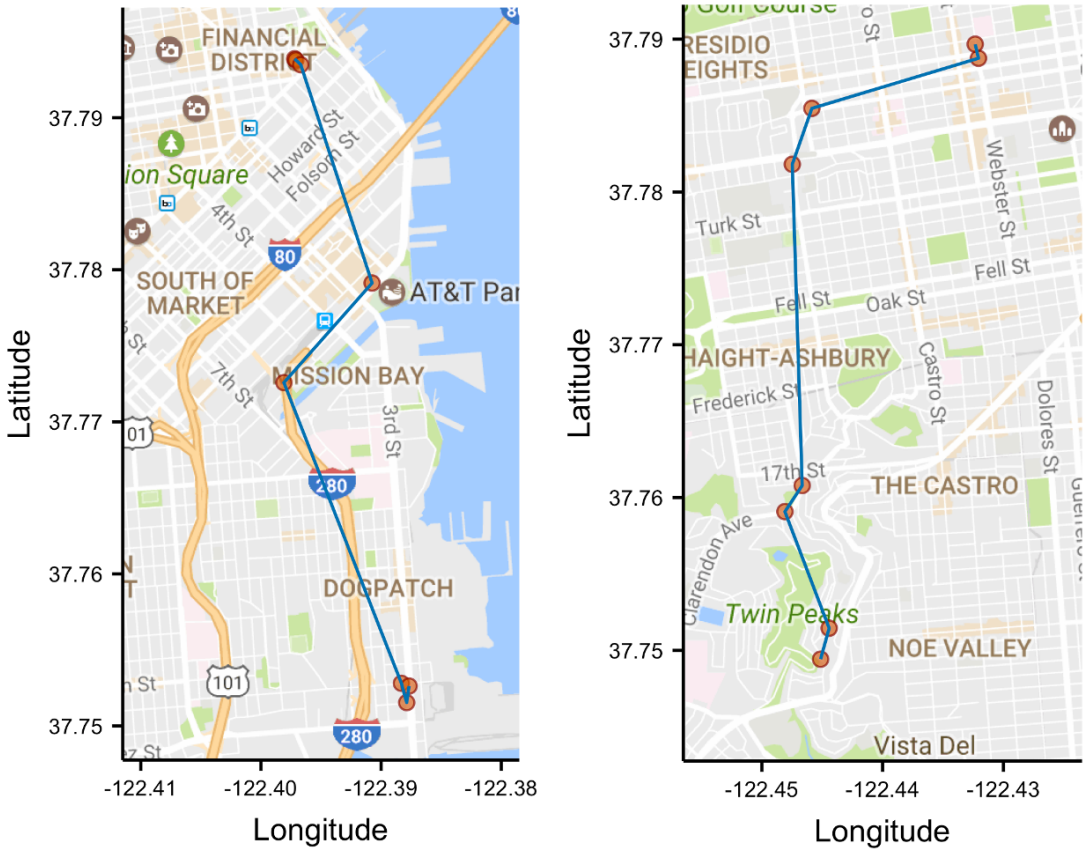

Figure 8: Two sampled trajectories from the Cabspotting dataset. In some cases, the gaps between any two consecutive observations of a trajectory is too big to be meaningfully filled with linear interpolation.

tween every consecutive observation can affect the linear interpolation. Thus, all trajectories that contain a segment of consecutive samples where the update interval is bigger than a threshold are removed.

3.2.4 Dataset Specific Thresholds

Each of the three selected datasets for our study has their specific level of quality. Accordingly, we have selected a set of thresholds to have the best possible level of accuracy in each dataset. All threshold values are stated table 1.

Since Cabspotting dataset contains traces both when cabs had a passenger and when they were empty, we remove all the empty cab traces in addition to the described

Table 1: Threshold values used for cleaning for each dataset

	Cabspotting	Geolife	Taxi Service Trajectories
Velocity Threshold	$34_{m/s}$	$34_{m/s}$	$34_{m/s}$
Minimum Trip Distance	$250m$	$200m$	$250m$
Minimum Trip Duration	$60s$	$30s$	$60s$
Maximum Trip Duration	$14400s$	$14400s$	$14400s$
Maximum Sampling Interval	$180s$	$180s$	$180s$
Maximum Consecutive Samples Distance	$1000m$	$400m$	300

cleaning steps for all the datasets.

3.2.5 Interpolation

Because of the update interval between two position readings, we might still have gaps in the extracted trajectories. To get rid of the gaps, we use linear interpolation on the sequence of cells as an approximation for missing cells in the trajectory. To clarify, assume we have two data points P_1 and P_2 with the following coordinates, (x_1, y_1) and (x_2, y_2). If we want to fill in the gap between these two points, we consider each axis separately. Here, we explain how the linear interpolation works on one axis, and the value on the y-axis can be calculated in a similar way. Let P_1 and P_2 be sensed at times t_1 and t_2 respectively. Therefore, in order to approximate the value x_t on x-axis at some time t between t_1 and t_2, we calculate:

$$x_t = x_2 - (x_2 - x_1)\frac{(t - t_1)}{(t_2 - t_1)} \tag{3}$$

We use interpolation to reduce the time interval between the data points to ten seconds. We did not choose a smaller value for the interval between the updates since we might introduce harsh oscillations to our trajectories. After this step, we remove those traces that still contain gaps. We have categorised trip trajectories into different groups based on their source. Each of these trajectories $T_{s,d}$ consist of a sequence of n cells $c_1, ..., c_n$, where $c_1 = s$ and $c_n = d$. Whenever the moving object of our interest starts to move from a location, we will select and extract all

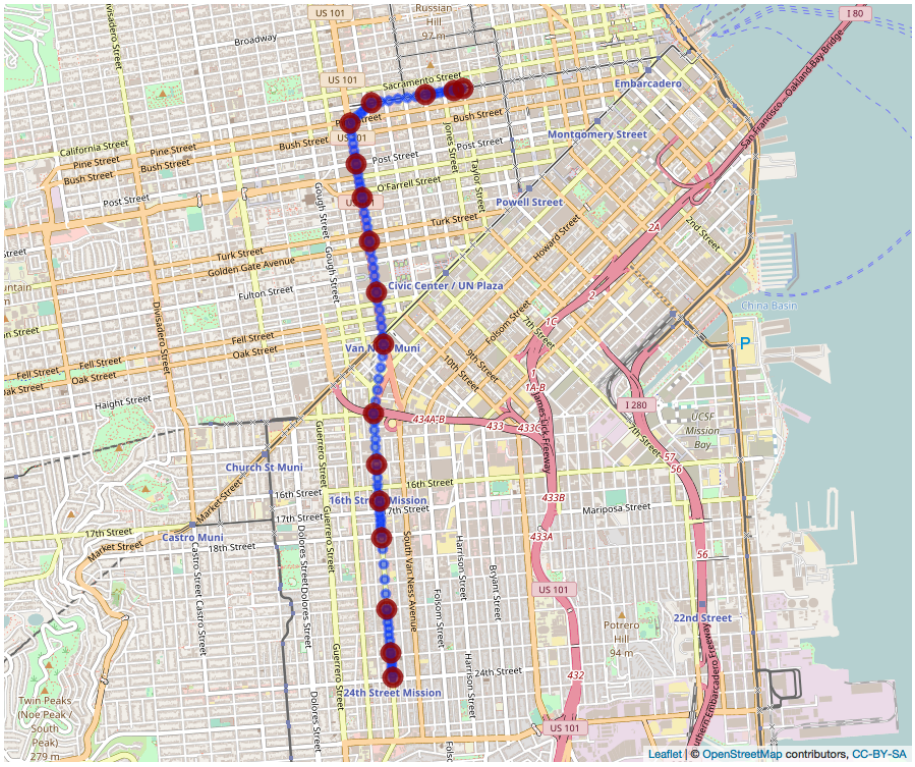

Figure 9: Interpolation example

of the trajectories in our system that have the same cell as their source for the next step of processing. Before going further, we have to demonstrate the necessary tools for measuring regularity.

3.3 Grid Transformation

To analyse and quantify the regularity of trajectories, we use a grid network model over the world. We then discretise the trajectory data by mapping each observation into a discrete grid cell. The mapping is done by converting each longitude and latitude pair into a cell index on a $d \times d$ sized grid according the algorithm described by Nurmi et al. [NBK10]. Figure 10 illustrates an example of using a grid network to discretise a trajectory into multiple cells. This enables us to be able to index the data points to grid cells and aggregate all the corresponding data points in one cell together. Accordingly, each mobility trace will become a sequence of grid cells. Using a grid model to discretise the trajectories has some advantages. Discretising

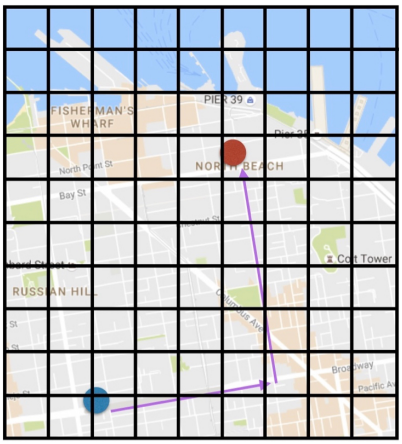

Figure 10: Before being able to use our data for regularity analysis, we discretise the trajectory using a grid model into discretised cells.

the measurements is essential for computational tractability, and helps to overcome inaccuracies in location measurements, e.g., due to driving on a different lane or due to inaccurate GPS fixes.

In our analysis, we use different cell sizes for each of the three datasets based on their average sampling rate. Cell sizes: $400m \times 400m$, $200m \times 200m$, and $250m \times 250m$ are selected for the Cabspotting, Geolife, and Taxi Service Trajectories datasets, respectively. The cell sizes are selected as a trade-off between location accuracy and computational requirements. Choosing lower values for d results in trajectories with higher resolution, but such values will lead to inaccurate trajectories since the original dataset's observations did not have the required accuracy.

As a final step, we have to make sure that the source is always connected to the destination. It means that from any cell, the next cell along the path should be one of the eight neighbours of the current cell. Therefore, after the discretisation of the trajectories, we again remove any trajectory that may end up with a gap.

3.4 Data Description

Before moving into the regularity analysis chapter, it is worth looking into some statistics describing the final cleaned datasets to get more familiar with them.

After the cleaning pipeline, Cabspotting dataset has 535 unique users and $318,011$

distinct trajectories, Geolife dataset has 173 unique users and 41,756 distinct trajectories, and Taxi Service Trajectories dataset has 439 unique users and 945,881 distinct trajectories. Table 2 provides more detailed statistics for each dataset after the cleaning process.

After discretising the data using a grid network, each trajectory is then defined by a sequence of cells. Figure 11 shows the distribution of lengths of the trajectories. We can observe that the trajectories in the Cabspotting dataset have less variety compared to the other two datasets, and the average length of its trajectories is just a bit over five cells. Thus, this dataset has mostly short trajectories. This is mainly because of the larger cell sizes used for discretisation in the Cabspotting dataset because of its relatively poor quality compared to the other datasets. In addition, compared to the other datasets, Cabspotting contained more errors; therefore, more trajectories got removed during the cleaning process.

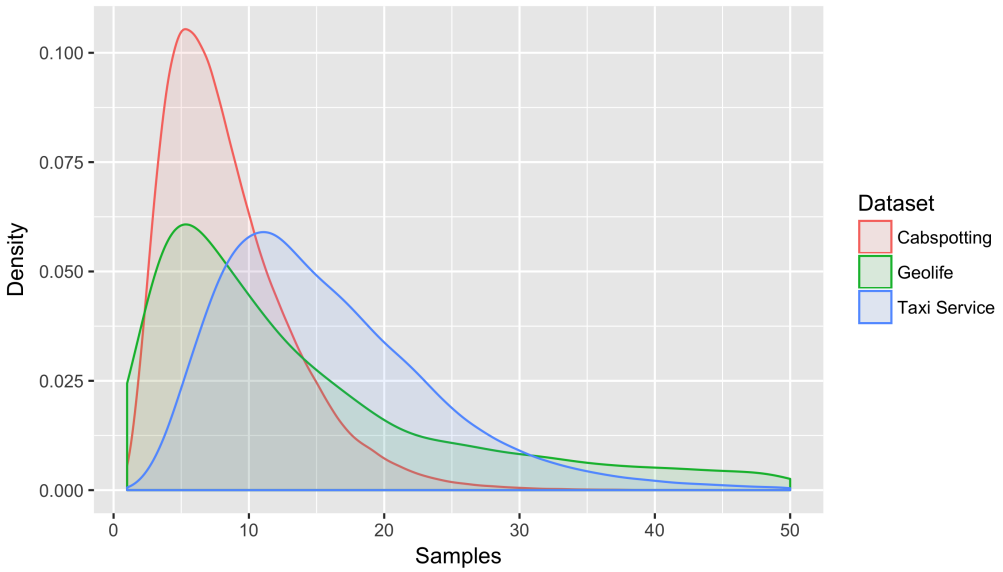

Figure 11: The distribution of discretised trajectory length for each dataset.

On the other hand, Geolife trajectories contain more variety but still the mean of the trajectory lengths is around five samples, just like the Cabspotting dataset. Considering the fact that we used the smallest cell size for the Geolife dataset, it might be expected that the dataset should have longer trajectories, but it should be taken into account that in addition to the driving trajectories, Geolife dataset contains other modes of transportation like walking. This is one of the reasons that

Table 2: Cabspotting dataset after cleaning

Cabspotting					
	Maximum consecutive points distance	Maximum sampling rate	Maximum trip duration	Maximum trip distance	Maximum segment velocity
Minimum	140.306	21.000	60.000	250.074	0.145
1st Quartile	473.201	109.000	301.000	1207.944	7.047
Median	545.905	129.000	450.000	1894.011	9.312
Mean	553.619	147.667	501.730	2189.747	9.533
3rd Quartile	630.594	175.000	640.000	2853.580	11.833
Maximum	1101.062	6713.000	9880.000	11456.437	33.988
Geolife					
	Maximum consecutive points distance	Maximum sampling rate	Maximum trip duration	Maximum trip distance	Maximum segment velocity
Minimum	55.567	5.000	30.000	200.071	0.163
1st Quartile	214.729	98.000	325.000	503.792	4.064
Median	247.327	172.000	697.000	1204.249	11.288
Mean	254.859	228.467	1003.558	3278.627	12.031
3rd Quartile	287.981	270.000	1293.000	3584.881	18.815
Maximum	520.612	12009.000	14021.000	101859.025	33.998
Taxi Service Trajectories					
	Maximum consecutive points distance	Maximum sampling rate	Maximum trip duration	Maximum trip distance	Maximum segment velocity
Minimum	99.951	15.000	60.000	250.049	0.050
1st Quartile	327.045	90.000	375.000	1306.066	11.395
Median	363.153	105.000	540.000	2015.175	13.985
Mean	363.443	143.198	629.459	2299.213	13.678
3rd Quartile	399.784	150.000	765.000	3028.302	16.499
Maximum	679.239	13455.000	14295.000	47779.620	20.000

originally Geolife trajectories are shorter compared to the other two datasets. For the Taxi Service Trajectories we have more variance compared to the other datasets and also longer trajectories in general. In the Taxi Service Trajectories dataset the mean of the trajectory lengths is around 11 cells. While the dataset is similar to the Cabspotting dataset in terms of transportation mode, relatively it has higher quality trajectories. We also selected a cell size smaller than the one used for discretising the Cabspotting. Therefore, generally longer trajectories were expected for the Taxi Service Trajectories dataset.

4 Trajectory Regularity

Previous studies on human mobility have demonstrated that individual mobility patterns, characterised by transitions between important locations, are predictive to a high degree [LWB+13]. While the regularity of location transitions has been established, less is known about the routes that people take while transitioning between locations. That is to say, the regularity of human movement trajectories has mostly been overlooked and no conclusive information is available.

Individuals' everyday movements are affected by different factors, which make them more regular. As an example, consider the movement trajectories of a person that has several frequently visited locations, such as home and work. His movement trajectories between these locations tend to overlap to a high degree since the person usually takes the same familiar path. In addition to the personal attributes of a person, the movement trajectories are also constrained by the overall structure of road networks and public transportation routes.

In this section we describe methods for quantifying the amount of randomness present in movement trajectories. We first demonstrate how to use the discretised trajectory data, so that we can use the Markov chains. Next, entropy is described and used to quantify the irregularity of each discrete cell. Then the entropy of the Markov trajectories is presented as a way to quantify the randomness between any source and destination. Finally, the entropy of conditional Markov trajectories is discussed as means to measure the irregularity of trajectories by conditioning on an intermediate cell.

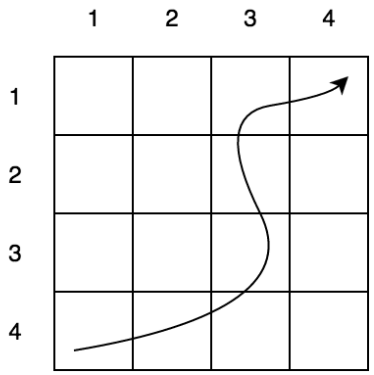

Figure 12: A movement trajectory discretised over a 2-D coordinate system.

4.1 Representing the Trajectories as Markov Chains

In the previous section, we have shown how to discretise GPS traces into a 2-D coordinate systems with square-shaped cells. As an example consider Figure 12, which shows a movement trajectory discretised over square cells. Here, the trajectory takes on a path passing through the cells $C_{4,1}, C_{4,2}, C_{4,3}, C_{3,3}, C_{2,3}, C_{1,3}, C_{1,4}$. Therefore, each trajectory is a sequence of transitions between subsequent cells starting from the source cell and ending at the destination cell. We quantify regularity of trajectories by modelling them as (first order) Markov chains. Each cell along a trajectory corresponds to a state in a Markov chain and movements from one cell to another correspond to transitions between states. We construct a transition matrix based on the trajectories and compute the corresponding probability transition matrix. Given the collection of all trajectories $\mathcal{T}$, we create a Markov chain $\mathcal{M}(\mathcal{T})$ by constructing a transition probability matrix $P = p_{i,j}$ where $p_{i,j}$ denotes the probability of observing a transition between grid cells i and j. Next, as an initial attempt to analyse the regularity, we calculate the entropy of each state to learn about their irregularity. Entropy described by Equation 4 is a measure from Information Theory, and it is used to quantify the unpredictability of a state.

$$H(X) = \Sigma_{i=1}^{n} P(x_i) I(x_i) = -\Sigma_{i=1}^{n} P(x_i) log_b P(x_i) \tag{4}$$

In general, it is possible to have a transition between any two states of the Markov chain. Since we are using a grid model and the generated Markov chain is a representation of our trajectories, the only transition is possible from any state is only to its eight neighbours as illustrated in Figure 13. Therefore, $P(x_i)$ from Equation 4 corresponds to the probability of the transition between the current cell and its i'th neighbour. Based on this constraint we can re-write Equation 4 as follows:

$$H(X) = -\Sigma_{i=1}^{8} P(x_i) \log_2 P(x_i) \tag{5}$$

In the case where the transition probability is evenly distributed between all the neighbours – meaning that the current state contains the highest amount of irregularity – Equation 5 is equal to the value 3. On the other hand, if all the transitions from the current cell are to one of the eight neighbours – meaning that the outcome of the transition from this cell is completely predictable – the Equation 5 will be equal to 0. As a result we can see that the cell entropy value is bounded as follows:

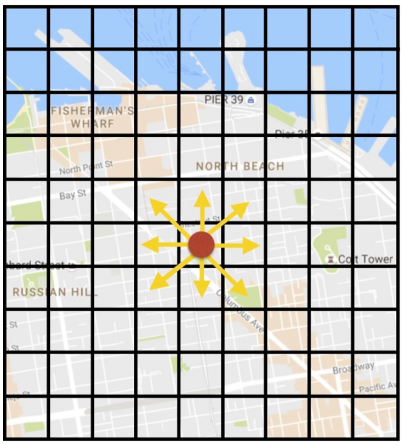

Figure 13: In our application environment, the only possible transitions from any states is to its eight neighbours.

$$0 \leq H(X) \leq 3 \tag{6}$$

We modelled the trajectories of the three datasets as (first order) Markov chains. Next, we calculated the popularity of each cell by counting the number of times the cell was visited by different trajectories. Furthermore, we computed the entropy for all states. Finally, we have plotted the results using heat-maps as illustrated in Figure 14.

Consider Figure 14, in cell visitation figures (a, c, e) we can locate the areas where most of the movement trajectories are concentrated. Since we have the smallest cell size and highest sampling resolution for the Geolife dataset, we can observe that the dense areas are mostly around the main roads. Because of the higher cell sizes for the other two datasets, there is no specific paths where the trajectories would be concentrated around. Nevertheless, based on the cell irregularity plots (b, d, f) of all three datasets, it is obvious that there are only a subset of cells in the dense areas where they demonstrate the highest amounts of irregularity. This is in line with our assumption that trajectories consist of more regular segments connected by highly irregular points that are responsible for most of the irregularity exhibited by the trajectory. In the next section, we look into the cell entropies along the path in more detail.

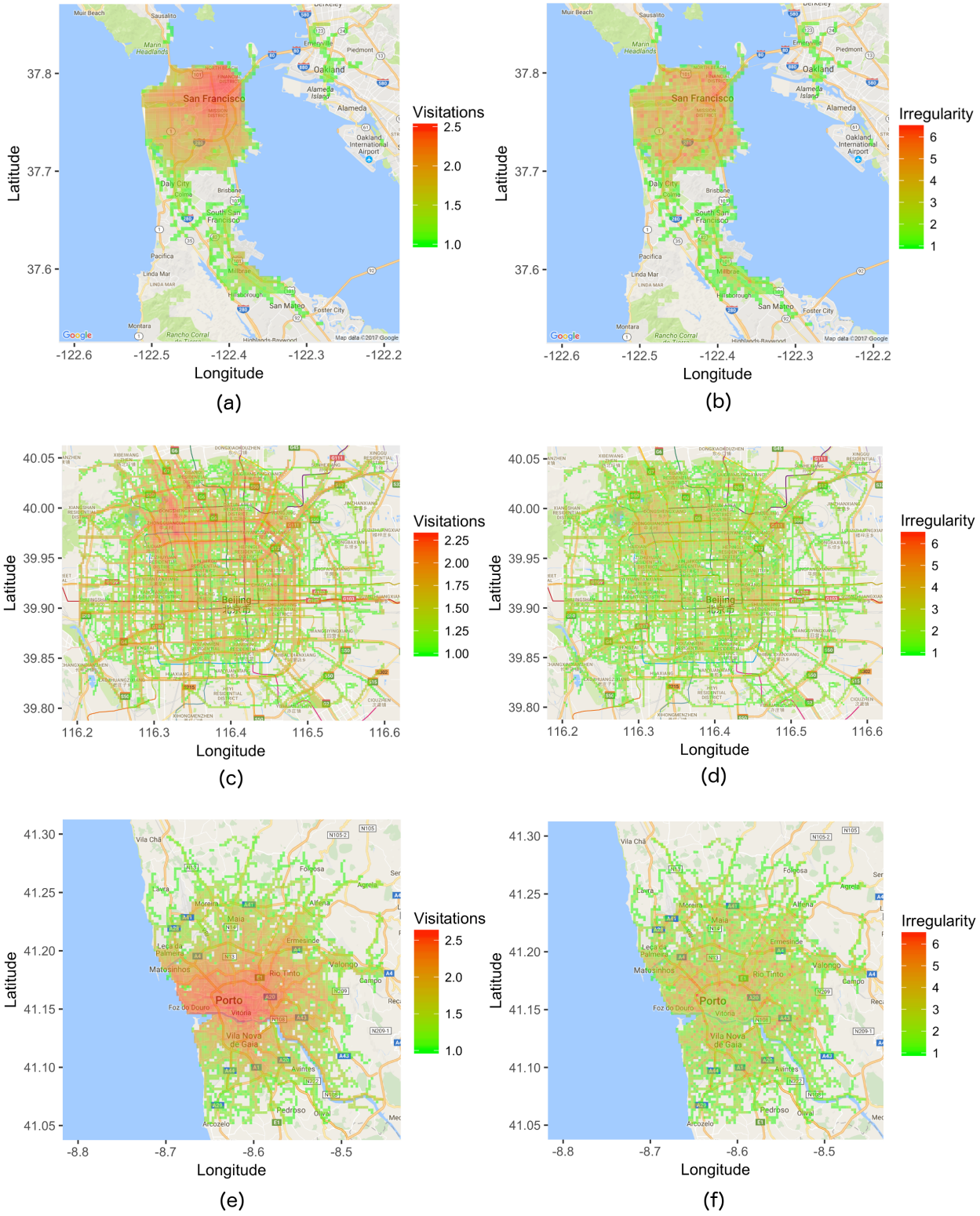

Figure 14: a, c, and e display the popularity of each cell by computing the $loglog(x)$ where x is the number of visitations to that cell. b, d, and f shows the amount of irregularity within each cell by computing 2^x where x is the cell entropy.

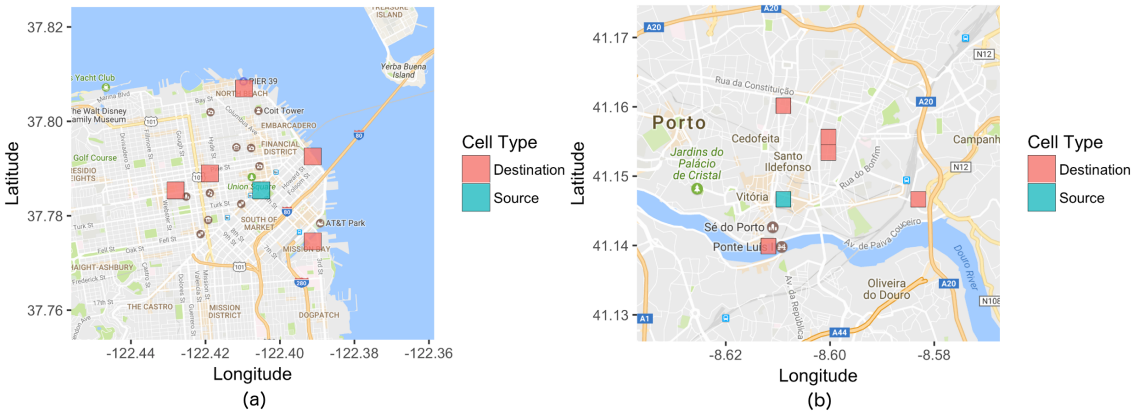

Figure 15: For Cabspotting and Taxi Service Trajectory trajectories, the most popular source is selected. Accordingly, based on the trajectories that start from the selected cell, five popular destinations are selected.

4.2 Distribution of Entropy within a Trajectory

To gain more familiarity with cell entropies within a trajectory, we select the most popular source cell (cell where highest number of trajectories started from) of the Cabspotting and Taxi Service Trajectories datasets. For Cabspotting dataset, $11,888$ trajectories start from the most popular source cell. For Taxi Service Trajectories, this number is $38,024$. Next, we select five of the most popular destinations based on the trajectories starting from the previously chosen source cell. Figure 15 shows these cells for the two mentioned datasets.

As the next step, we randomly sample one trajectory from each of the selected source destination pair which gives us five trajectories for each dataset. We match all the cell entropies along the trajectories and interpolate them along 20 points, so that they are of the same length. Figure 16 shows the interpolated cell entropies for the five selected trajectories of each dataset. Since a large number of trajectories start from the selected sources, we can observe that the cell entropy is relatively high at the beginning of the trip. As we progress along the paths, there are points where the entropy becomes relatively low that results in higher predictability of the next transition.

Cells with high local entropy value appear as peaks in our plot. These correspond to decision points where trajectories start to deviate from each other, these are namely

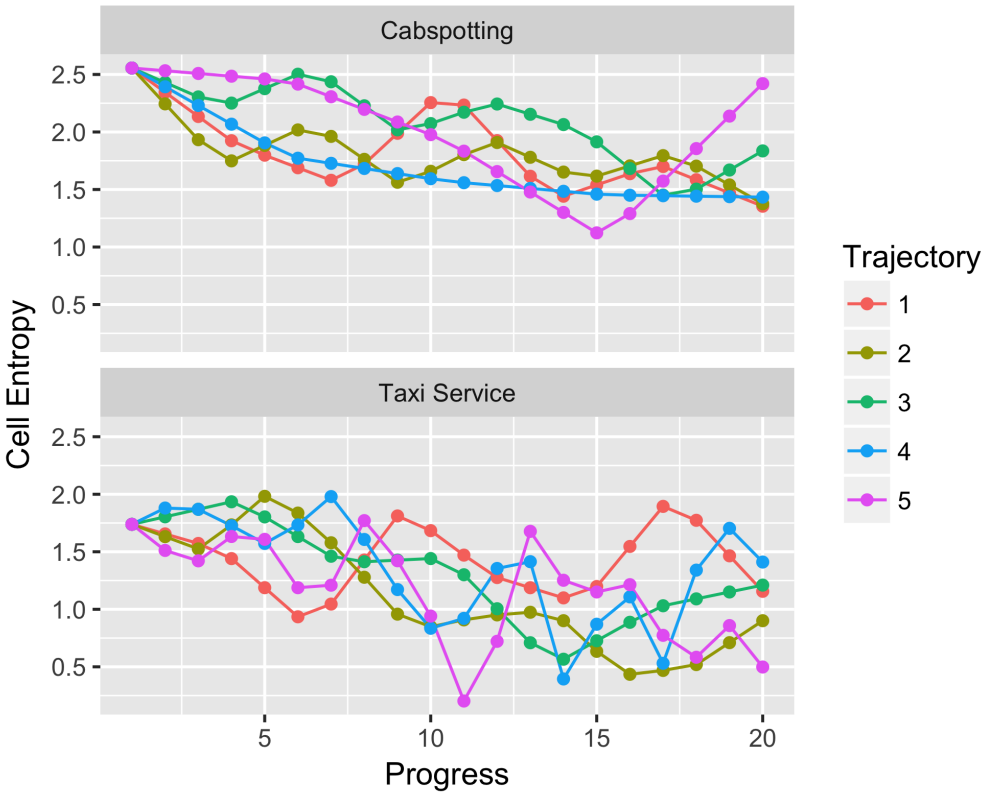

Figure 16: Cell local entropies along five different trajectories chosen from the two datasets between the most popular source and five popular destinations.

our fork-points. Having knowledge about the transition from the fork-point for a moving object gives higher information for predicting the future of the trajectory. The segments created by these fork-points are highly regular; therefore, it is our desire to extract the points that are mainly responsible for the amount of randomness in a trajectory to be able to encode the whole trajectory with them. The fork-points in each trajectory can help us to split the trajectories into segments that have higher regularity. In Section five we introduce an algorithm for segmenting the trajectories into these more regular segments by using fork-points. In the next section, we look into the quantification of randomness within trajectories as a whole.

4.3 The Entropy of Markov Trajectories

We are interested in a method that enables us to quantify the overall amount of randomness within a trajectory. We use the entropy of Markov trajectories as described by Ekroot et al. [EC93] to quantify the randomness of trajectories in a irreducible finite state Markov chain. To model the discretised trajectories of all datasets, we created a transition matrix. The size of this square matrix depends on the total number of cells after discretisation. Therefore, if we have n unique cells in our grid network, we create a transition matrix $T_{n,n}$. We compute the frequency of each possible transition in the dataset and update the transition matrix accordingly. Next, we extract the probability transition matrix $P_{n,n}$ as an intermediate step in order to be able to compute the Markov trajectory entropies. $P_{n,n}$ can simply be created from $T_{n,n}$ by summing the values in each row, and dividing the values of each row by the calculated sum. The value of $P_{i,j}$ corresponds to the transition probability from state i to state j. Using the constructed Markov chain, it is possible to represent the trajectories of a moving object as a weighted random walk on a graph. In such a graph, each trajectory cell is represented by vertices and possible transitions by edges. This graph is known as the state transition diagram.

In order to compute the entropy of Markov trajectories, we create a trajectory entropy matrix H, and we use an example along the steps for better understanding. Figure 17 displays the state transition diagram of a 5-state Markov Chain with the probability transition matrix of:

$$P_{5,5} = \begin{pmatrix} 0 & 0.7 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0.2 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0.6 & 0 & 0 & 0.4 & 0 \end{pmatrix}$$

The associated state entropy rate of the given Markov chain example is given by:

$$H(\chi) = -\Sigma_{i,j}\mu_i P_{i,j} log P_{i,j} \tag{7}$$

In Equation 7, μ is the stationary distribution and can be calculated by solving $\mu = \mu P$. A trajectory T from the source i to the destination j is presented as $T_{i,j}$, in which no intervening state is equal to j, and we want to compute the trajectory entropies $H(T_{i,j})$. A general closed form solution used for computing the $H(T_{i,j})$ is

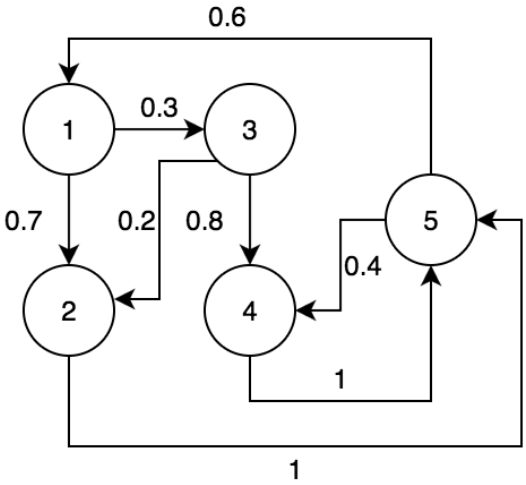

Figure 17: State transition diagram for a 5-state Markov Chain.

as follows:

$$H = K - \tilde{K} + H_{\Delta} \tag{8}$$

The matrix H_{Δ} is the entropy rate of a state. It is a diagonal matrix with entries $(H_{\Delta})_{i,i} = \frac{H(\chi)}{\mu_{i\cdot}}$. Here $H(\chi)$ is the entropy rate of a state. Equation 9 describes how to calculate H_{Δ}.

$$H(\chi) = -\Sigma_{i,j}\mu_i P_{i,j} log P_{i,j} \tag{9}$$

And matrix K in Equation 8 can be calculated using:

$$(I - P + A)^{-1}(H^* - H_{\Delta}) \tag{10}$$

$\tilde{K}$ is a matrix in which the element on ith row and in jth column equals the diagonal element $K_{j,j}$ of K; A is the matrix of stationary probabilities with entries $A_{i,j} = \mu_j$; H_* is the matrix of single-step entropies with entries $H^*_{i,j} = H(P_{i\cdot}) = -\Sigma_k P_{i,k} log P_{i,k}$; and H_{Δ} is a diagonal matrix with entries $(H_{\Delta})_{i,i} = \frac{H(\chi)}{\mu_{i\cdot}}$

Now, we are able to compute the matrix of trajectory entropies Hn, n as described by Equation 8. Hn, n contains the entropies of the trajectories between any two

cells in the extracted data points. For our example Markov Chain we would get the following Markov Trajectory entropies:

$$H_{5,5} = \begin{pmatrix} 2.7161 & 1.9555 & 6.7135 & 3.3746 & 1.0978 \\ 1.6182 & 3.5738 & 8.3318 & 2.9957 & 0 \\ 2.3401 & 3.5810 & 9.0537 & 1.3210 & 0.72192 \\ 1.6182 & 3.5738 & 8.3318 & 2.9957 & 0 \\ 1.6182 & 3.5738 & 8.3318 & 2.9957 & 1.6296 \end{pmatrix}$$

For example, the entropy of the $T_{1,5}$ trajectory is 1.6296 bits, while the entropy of deterministic trajectories – meaning the transitions are completely predictable – $T_{4,5}$ and $T_{2,5}$ is equal to zero which means that these trajectories do not contain any randomness.

Previously, we gave an example of how people move between important places. For example, people tend to commute on the same path between their home and work. We made an assumption that for these more frequent routes, there will be more regularity present. To investigate this matter, we have extracted all the trajectories starting from the most popular source in each dataset and computed the Markov trajectory entropy matrix. We can observe from the result, as shown in Figure 18 that our assumption was in fact true. The more trips we have for a given source and destination, the more regular the paths are. In the next section, we look into how knowing which cells are visited between a source and destination changes the entropy of Markov trajectories. Up until now, we analysed the trajectories by discretising them into states and showing that some of these states have much lower uncertainty than others. Then, we described the entropy of Markov trajectory which enables us to compute the randomness within any complete trajectory between any two locations. Next, we introduce entropy of conditional Markov trajectories which helps us to compute the trajectory entropy after conditioning on an intermediate cell.

4.4 The Entropy of Conditional Markov Trajectories

Based on the technique introduced by Ekroot and Cover [EC93], we were able to calculate the amount of randomness in an irreducible finite state Markov chain. Therefore, we were able to compute the entropy of the trajectory between any two states in the graph that models the user's mobility. However, if we gain new infor-

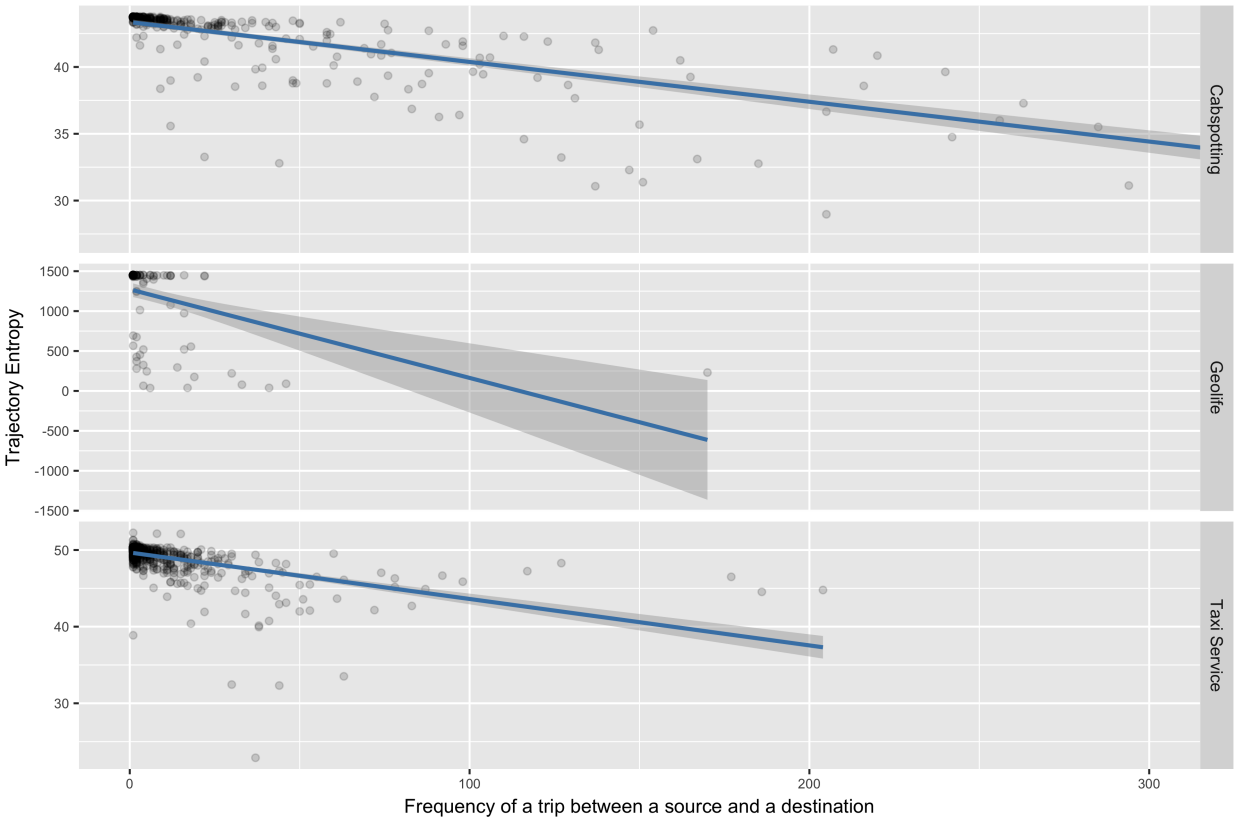

Figure 18: The Markov trajectory entropy is computed for all the trajectories started from the most popular source for all three datasets. There is a decreasing trend in the trajectory entropy values as we observe more trajectories between a source and destination.

mation that the user passed through states on its path to reach the destination, we cannot utilise the new information with the previous approach. Here, we describe a method for computing the entropy of conditional Markov trajectories by introducing a way to transform the original Markov chain so that it expresses the conditional distribution of trajectories based on the work done by Kafsi et al. [KGT13].

To understand this matter better, consider the following example. Based on our example Markov Chain shown in Figure 17, we computed the entropy of the trajectory between the two states 1 and 5 to be 1.6296 bits. If we learn that the user has gone through state number 4 before reaching the destination, we would like to compute the trajectory entropy $T_{1,5}$ conditioned on visiting the fourth state along the way.

Before going further and describing the entropy of conditional Markov trajectories, it should be noted that the additive property does not hold for the entropy of Markov trajectories. Based on our example we would have the $H_{1,4} + H_{4,5} = 3.3746$ but after conditioning on visiting the fourth state and computing the conditional entropy we get $H_{1,5|4} = 0$. Therefore, it is evident that the additional information about the user passing state number 4 makes the trajectory fully predictable. By extracting such intermediate states, it is possible to fully encode the trajectory. Therefore, if there exists a set of intermediate cells such that by conditioning on them the conditional entropy of Markov trajectories becomes zero – meaning the path changing into being fully predictable – we can break the whole trajectory into segments connected by the cells of this set. As a result we would have fully regular segments, connected by the decision points.

Before going through the algorithm for computing the entropy of conditional Markov trajectories, we first demonstrate a few concepts that are needed for the algorithm. We can define the conditional entropy of a trajectory between the source s and destination d, $T_{s,d}$, that passes an intermediate state u by:

$$H_{sd|u} = H(T_{sd}|T_{sd} \in \mathcal{T}_{sd}^{u})$$
$$= -\Sigma_{t_{sd} \in \mathcal{T}_{sd}^{u}} p(t_{sd}|T_{sd} \in \mathcal{T}_{sd}^{u}) log p(t_{sd}|T_{sd} \in \mathcal{T}_{sd}^{u})$$

where $\mathcal{T}_{s,d}^{u}$ is a set that contains all the trajectories between the source s and the destination d that go through the intermediate state u. Now consider a sequence of intermediate cells $\mathbf{u} = u_1 u_2 ... u_i$. Then $\mathcal{T}_{sd}^{\mathbf{u}}$ is the set of all trajectories in the set $\mathcal{T}_{sd}$ that visits every intermediate cell in $\mathbf{u}$ in order, before reaching the destination. The entropy of the trajectory between source s and destination d where it visits the intermediate cells $\mathbf{u}$ is:

$$H(T_{sd}|T_{sd} \in \mathcal{T}_{sd}^{u}) = \sum_{k=0}^{l-1} H_{u_k u_{k+1}|\bar{d}} + H_{u_l d} \tag{11}$$

where u_0 is the source s.

As part of the algorithm we have to compute the entropy of the Markov trajectories which was described in the last section. During the steps of this algorithm, our Markov chain is not necessarily irreducible. As a result, we cannot use the algorithm by Ekroot and Cover as it is valid only for the irreducible Markov chains. Because of this problem, we demonstrate an alternative method in to compute the entropy

Algorithm 1 Conditional Markov Trajectory Entropy

1: $u_0 \leftarrow s$

2: for k = 0 to l-1 do

3: Compute P'_k from P using:

$$(P'_k)_{ij} = \begin{cases} 0 & \text{if } i = u_{k+1}, d \text{ and } i \neq j, \\ 1 & \text{if } i = u_{k+1}, d \text{ and } i = j, \\ P_{ij} & \text{if } i \neq u_{k+1}, d \text{ and } \alpha_{idu_{k+1}} = 1, \\ \frac{1-\alpha_{jdu_{k+1}}}{1-\alpha_{idu_{k+1}}} P_{ij} & \text{if } i \neq u_{k+1}, d \text{ and } \alpha_{idu_{k+1}} < 1. \end{cases}$$

4: Compute $H(T'_{u_k u_{k+1}})$ from P'_k using Equation 13

5: Assign the computed value $H(T'_{u_k u_{k+1}})$ in the last step to $H_{u_k u_{k+1}|\bar{d}}$

6: end for

7: Computed $H_{u_l d}$ from P using Equation 13

8: $H_{sd|u_1 \ldots u_l} = \sum_{k=0}^{l-1} H_{u_k u_{k+1}|\bar{d}} + H_{u_l d}$

9: Return $H_{sd|u_1 \ldots u_l}$

of a Markov trajectory that can be used for non-irreducible chains.

Consider a Markov chain such that there exists a path with positive probability from any state to a give state d with the transition probability matrix P. Then, by removing the d^{th} row and column of P we can get the sub-matrix Q_d as follows:

$$\left[\begin{array}{c|c} Q_d & \begin{matrix} P_{1d} \\ \vdots \end{matrix} \\ \hline \begin{matrix} P_{d1} \ldots \end{matrix} & P_{dd} \end{array}\right] \tag{12}$$

Then the entropy of the trajectory between source s and destination d such that $s \neq d$ is given by the following equation:

$$H_{sd} = \sum_{k \neq d} ((I - Q_d)^{-1})_{sk} H(P_{k.}) \tag{13}$$

Now we can compute the conditional Markov trajectory entropies using algorithm 1 as described by [KGT13]. The algorithm takes as an input, source and destination states s, d, sequence of visited middle states $u = u_1 ... u_l$, and probability transition matrix P of the corresponding Markov chain. As the output, the algorithm returns the entropy of Markov trajectory conditioned on the intermediate cells $H_{sd|u_1 \ldots u_l}$.

With a probability transition matrix with N states, and l intermediate states, the algorithm has running time-complexity of $O(lN^3)$. It should be mentioned that the value of $H_{sd|u}$ would not necessary be smaller than H_{sd} as some may expect. This is due to the fact that $H_{sd|u}$ is conditioned on learning a dependent random variable and not the entropy of T_{sd} given another random variable. Therefore, $H_{sd|u}$ can be bigger than H_{sd}.

In this chapter, we covered the methods that can assist the quantification of randomness in trajectories. In the first two parts, we demonstrated how to transform our trajectory dataset into first-order Markov chains and looked into the distribution of cell entropies in trajectories. We also provided methods to both measure the entropy of Markov trajectories and the entropy of conditional Markov trajectories. By using these methods, we will provide two segmentation methods in the next chapter that can be used to break the trajectories into more regular parts.

5 Trajectory Segmentation and Compression

In reality, movement paths of people do have common parts that overlap with each other. These shared segments can be for example along main roads until they reach a crowded intersection that acts a terminal point. This is the point that the people's paths start to deviate from each other in order to reach their very own destinations, which makes the rest of the path more specific to each individual. In the Trajectory Regularity chapter we showed that most of the uncertainty associated with route choices is concentrated along a small set of fork-points. Each fork-point can be effectively understood as a point that segments the overall trajectory into sub-segments that are relatively more regular compared to the whole trajectory.

We are interested in extracting these fork-points. For example, cells with transitions evenly distributed among their neighbours. These states demonstrate higher entropy values as shown previously by discretising the trajectories and computing their corresponding local entropy. In this chapter, we introduce two different algorithms that can be used to segment the trajectories by identifying the fork-points.

5.1 Conditional Markov Entropy Based Segmentation

In Section 3.3 we described how a grid model can be used to discretise trajectories. After the discretisation we represented our trajectories as Markov chains. This will result in a general model of how people move between different states that requires no timing information. In this section we describe a segmentation algorithm proposed by Kafsi et al. [KGT15]. The algorithm uses the conditional Markov entropy measure introduced in Section 4.4 to segment trajectories when no timing information is available.

Let s and d be the source and destination of a given trajectory, and let u be an intermediate point that the user visits along his path from s to d. The algorithm segments the trajectories by finding the intermediate states where the ratio between conditional and unconditional entropy, i.e. $H_{sd|u}/H_{sd}$ is over a specified threshold.

In the previous chapter, we discussed how conditioning on an intermediate cell may increase the Markov trajectory entropy. To understand this matter better, consider the following example. We select a source cell s and a destination d, and we want to analyse the entropy distribution of trajectories between the pair. We measure the entropy of the whole trajectory and move on to measure the conditional

entropies. We do this by conditioning on the intermediate cells between s and d as observed one by one. If observing an intermediate cell u results in a conditional Markov trajectory entropy $H_{sd|u}$ bigger than the Markov trajectory entropy H_{sd}, the intermediate cell u is a decision point that acts as the intermediate cell of many other trajectories in addition to the trajectories between s and d. Therefore, it results in lower predictability and higher uncertainty for the trajectories between s and d. As a result, the posterior distribution can be very different from the prior distribution. On the other hand, consider the intermediate cells that lead into $H_{sd|u}/H_{sd}$ ratios smaller than one. These are the points that act as intermediate points for the source s and destination d. As a result, they increase the amount of predictability of trajectories. Therefore, we can extract the segmentation points by finding the set of intermediate cells $U_\alpha(t_{sd})$ such that:

$$U_\alpha(t_{sd}) = \{u \in t_{sd} | H_{sd|u} > \alpha H_{sd}\} \tag{14}$$

where α is a predefined threshold. For $\alpha = 1$, the set $U_\alpha(t_{sd})$ contains all the intermediate cells that will increase the trajectory entropy t_{sd}. The set $U_\alpha(t_{sd})$ can be derived using the recursive segmentation algorithm of Kafsi et al. [KGT15]; see Algorithm 2. The algorithm extracts the segmentation points by finding the intermediate cells that have a $H_{sd|u}/H_{sd}$ ratio bigger than threshold α.

The algorithm, takes as an input the trajectory t_{sd}, transition probability matrix P, and threshold α. As the output, it produces the indices of segmentation points. With threshold $\alpha = 0$ the segmentation points set U will include all the intermediate cells of all trajectories between s and d. The algorithm, will recursively segment the trajectory by finding the intermediate cells that maximise the conditional entropy. It is possible to calculate all the conditional Markov entropies $H_{sd|u}$ beforehand to make the algorithm more efficient. But initially, due to the inversion of a matrix of size $O(N)$, the algorithm has $O(N^3)$ complexity.

5.2 Extracting the Segmentation Points

In addition to the segmentation using the conditional Markov trajectory entropy, we introduce our own alternative. This segmentation method is based on the algorithm described in our work [FN16]. Instead of conditional Markov trajectory entropy, our algorithm relies on local cell entropy values to detect segmentation points.

Algorithm 3 describes our fork-point detection approach. The core idea is to contin-

Algorithm 2 Trajectory segmentation

```
1: begin
2: U←0
3: if len(traj) > 2 then
4:     segment(traj, 0, len(traj)-1)
5:     end
6: return U
7: end
8: function SEGMENT(traj, i, j)
9:     k←partition(traj, i, j)
10:    if k≥0 then
11:        U←U∪k
12:        if i + 1 < k then
13:            segment (traj,i,k)
14:        if k + 1 < j then
15:            segment(traj,k,j)
16: function PARTITION(traj, i, j)
17:    s←traj, d←traj[j]
18:    k←argmax_{i<k<j} H_{sd|traj[k]}
19:    u←traj[k]
20:    if H_{sd|u} > αH_{sd} then
21:        return k
22:    else
23:        return -1
```

uously monitor the entropy rate (see Equation 4) of grid cells that are encountered, and to return cells that have a significantly higher entropy rate compared to the distribution of cell entropies of the corresponding Markov chain. These will make our set of fork-points. To accomplish this, we first have to compute the entropy rate of each grid cell of the corresponding probability transition matrix. Next, using Algorithm 3, we can determine significant deviations in entropy rates in a robust fashion and return those as the fork-points.

To compute the entropy rate of every grid cell, we should compute the entropy rate of each state of the probability transition matrix P of the underlying Markov chain. Each state corresponds to a cell along a trajectory and at any state the moving object has the possibility of moving to any of it's eight neighbouring states. Therefore, this

Algorithm 3 ForkPointDetection

```
1: allStateEntropies ← set of cell entropies in the probability transition matrix
2: μ ← mean(allStateEntropies)
3: σ ← sd(allStateEntropies)
4: z ← thresholdValue
5: for each trajectory in the dataset do
6:     stateEntropyArray ← state entropy of the trajectory cells in order
7:     window ← emptylist
8:     forkPoints ← emptylist
9:     for each cell of the selected trajectory do:
10:        stateEntropyArray ← stateEntropyArray.append(entropy(cell))
11:        if (entropy(cell) − μ)/σ ≥ z then
12:            window ← window.append(entropy(cell)).
13:        else
14:            if window is not empty then
15:                forkPoints ← forkPoints.append(max(window))
16:                window ← emptylist
17: return forkPoints
```

can be simply accomplished by keeping a running count of the number of transitions between grid cells. Note that the interpolation of the measurements ensures that subsequent measurements are in neighbouring cells; consequently, we only need to extract eight values per cell.

To find the set of fork-points, we detect significant deviations in the entropy rates using a statistical significance test. Specifically, we calculate estimates of the mean and standard deviation of the overall entropy rate of states of the corresponding probability transition matrix. Next, we go through each trajectory of the dataset and derive a z-score using Equation 15 for each cell that is encountered in that trajectory.

$$z = \frac{x - \mu}{\sigma} \tag{15}$$

The z value is like a threshold that provides us with a measurement of how off-target the current state entropy is from its distribution. Whenever the z-score of a cell exceeds a threshold of statistical significance, we initiate peak detection and buffer measurements until the score of the cell falls below the initial threshold. The cell

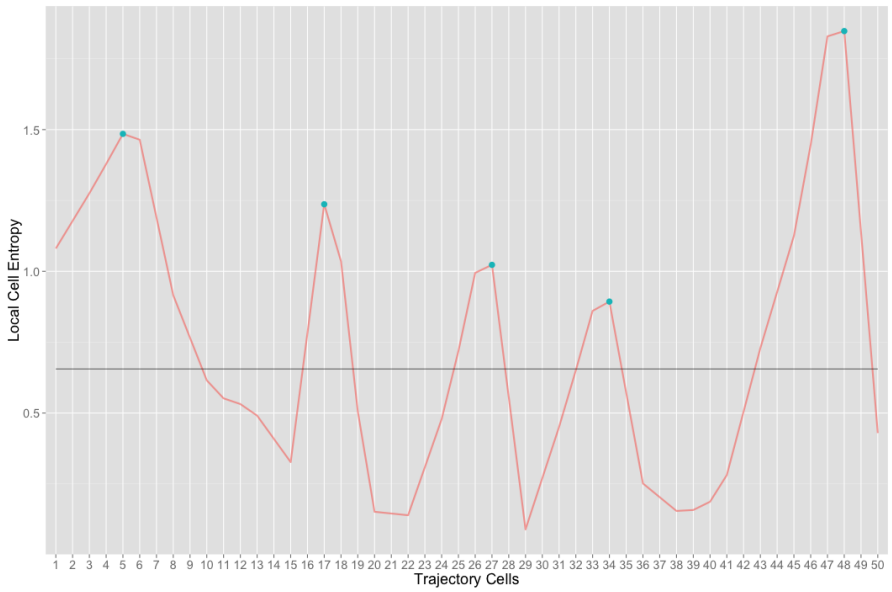

Figure 19: Whenever the entropy passes the threshold value, the cell entropies are added into a window and the maximum value from the window is selected as a segmentation point when the cell entropy value goes below the threshold.

with the maximal z-score is then selected as the fork point. Figure 19 demonstrates our fork-point detection algorithm over an example trajectory. The fork-points are shown as blue dots and the black line is is our threshold. We use a threshold of 0.5 for initiating the peak detection algorithm.

6 Prediction

In the previous chapters, we have covered methods that can be used to analyse and measure the amount of randomness in trajectories or how predictable they are. In this chapter, we move towards predicting the paths. For this aim, we introduce five popular trajectory similarity measures and evaluate their performance in our problem setup. Using the trajectory similarity measures, we first look into the problem of predicting the destination of trajectories. Furthermore, we look into predicting the segments of the trajectories by predicting the fork-points along a trajectory.

6.1 Destination Prediction

Whenever a user starts a new trip, we would like to learn more about his trip. For example, we want to predict the destination that the user is headed to. Without knowing anything regarding the trajectory of the user, this task becomes very difficult. When the user starts a new trip, the only available spatial information to us is the source of the trip. Therefore, to have a baseline, we first try to predict the destination by extracting all the destinations reached from that source and their corresponding frequencies. We select the most frequently reached destination of that set as the predicted destination.

Figure 20 shows the result of predicting the destination based on the most frequent destination for all the trajectories grouped by each source of the three datasets. From the plot we can observe that as the source becomes the starting point of more trajectories, the accuracy decreases significantly. When there are more trajectories starting from the same source the space of possible destinations grows; therefore, the result is expected. The accuracy decreases significantly and stabilises at 20 trajectories per source. For sources that are the origin of 20 or more trajectories, the accuracy is bounded between 0.125 and 0.25.

In the baseline, we looked into predicting the exact destination by selecting the most frequently reached destination for each source. But this approach does not provide any information about the quality of the predicted destination or how close it is to the actual destination. Therefore, we also extract how close is the predicted destination to the actual one. Figure 21 displays the results on how far is the predicted destination to the actual one. In this case, the distance increases immediately and stabilises at 5 trajectories per source.

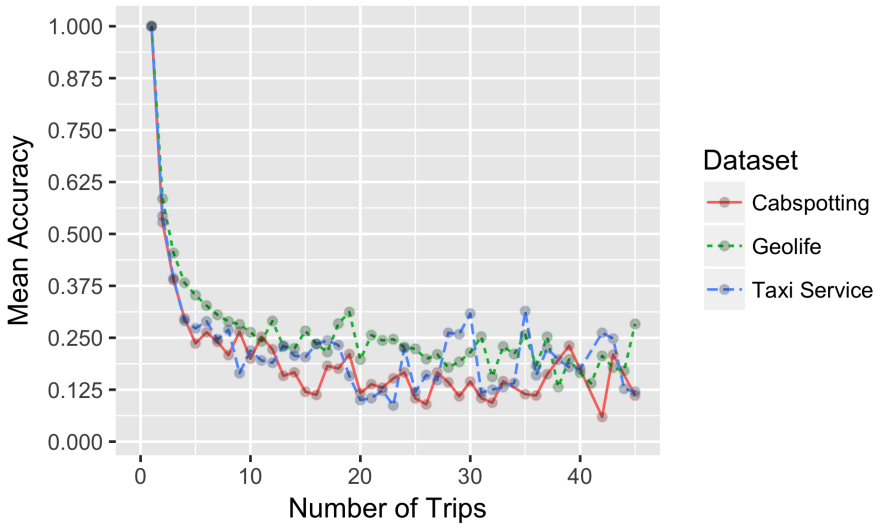

Figure 20: The average accuracy of the destination prediction based on the most frequent destination.

6.2 Trajectory Similarity Measures

As previously shown, when we have more sample trajectories between a source and destination pair, the entropy of the trajectories between those pairs tend to go lower. Based on this finding, we would like to identify which trajectories are similar to each other. Motivated by the wide variety of applications for trajectories, many similarity measures have been developed by the research community. Here we describe five popular trajectory similarity methods.

6.2.1 Euclidean Distance

One of the simplest distance measures used to quantify the similarity of trajectories is Euclidean distance [JDLVDVV80]. Euclidean distance or L_2-norm is the distance of the straight-line connecting two points. Euclidean distance is simple to implement and has a linear complexity. Therefore, it is capable of handling large sized datasets. It was first introduced as a distance measure for time series but later was used for trajectories as well. Given two trajectories T_1 and T_2 we can compute the euclidean distance using 16 where $d(p_{1,i}, p_{2,i})$ is the distance of the straight-line connecting the two points in the cartesian space. The two trajectories should have the same length so that we can use the Euclidean distance.

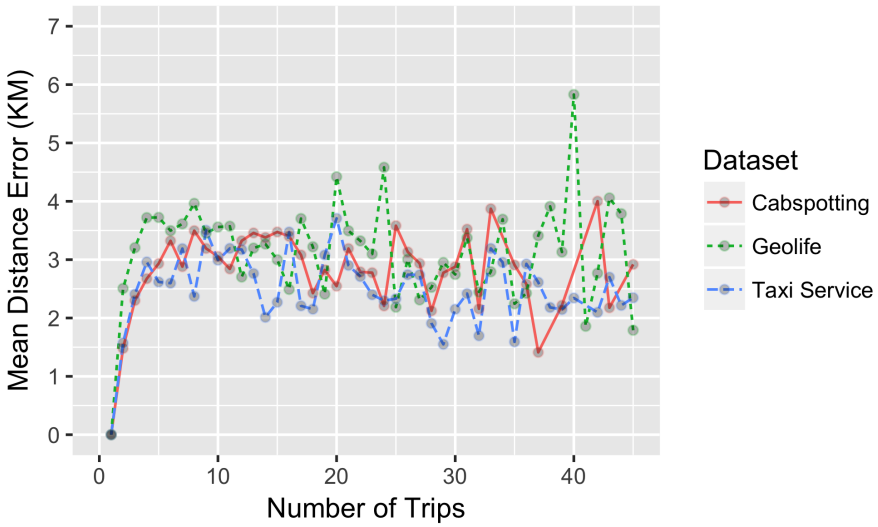

Figure 21: The average distance error of the destination prediction based on the most frequent destination to the actual destination.

$$\frac{\Sigma_{i=1}^{n} d(p_{1,i}, p_{2,i})}{n} \tag{16}$$

6.2.2 Dynamic Time Warping

Dynamic Time Warping (DTW) [SR88] is another time series similarity measure that is used in a variety of applications from speech recognition to signature recognition. It is also used for finding the similar trajectories. DTW has an advantage when it comes to temporal aspects of two trajectories as the number of observations in each trajectory and the speed of the moving object do not effect this measure. The distance is computed using a recursive approach and can be implemented using dynamic programming. Algorithm 4 describes Dynamic Time Warping. It takes two trajectories a and b with lengths n and m respectively. The algorithm checks all the possible point combinations between the two trajectories to find the lowest distance.

6.2.3 Longest Common Subsequence

Longest Common Subsequence (LCSS) [KH90] is string similarity measure that can be used with trajectories as well. One of the benefits of using LCSS for trajectories is its capability to handle noisy trajectories, and sequences of different lengths.

Algorithm 4 Dynamic Time Warping Distance

```
1: dtw ← array[n, m]
2: for i ← 1 to n do
3:     dtw[i, 0] ← infinity
4: for i ← 1 to m do
5:     dtw[0, i] ← infinity
6: dtw[0, 0] ← 0
7: for i ← 1 to n do
8:     for j ← 1 to m do
9:         cost ← d(a[i], b[j])
10:        dtw[i, j] ← cost + minimum(dtw[i − 1, j], dtw[i, j − 1], dtw[i − 1, j − 1])
11: returndtw[n, m]
```

Initially, the algorithm tries to find the longest common pattern between the two strings, or in our case discretised trajectories. We can use a threshold so that whenever the distance of two observations is below the defined threshold we count them as similar. This enables the measure to handle noisy observations. Algorithm 5 describes how to implement the measure. It takes two trajectories a and b with lengths m and n respectively.

Algorithm 5 Longest Common Subsequence

```
1: lcss ← array[m, n]
2: for i ← 0 to m do
3:     lcss[i, 0] ← 0
4: for i ← 0 to n do
5:     lcss[0, i] ← 0
6: for i ← 1 to m do
7:     for j ← 1 to n do
8:         if |a[i] − b[j]| < threshold then
9:             lcss[i, j] ← lcss[i − 1, j − 1] + 1
10:        else
11:            lcss[i, j] ← max(lcss[i, j − 1], lcss[i − 1, j])
12: returnlcss[m, n]
```

6.2.4 Edit Distance with Real Penalty

Edit Distance with Real Penalty (ERP) [CN04] is another measure for evaluating the similarity of trajectories. ERP is based on edit distance and uses the L_1-norm to quantify distances. By using L_1-norm, ERP is proven to be a metric measure. This is an advantage compared to DTW and LCSS. ERP behaves identically to DTW in the transformation case, but compares to a constant, g, whenever a deletion or insertion is used. The authors suggest a constant of 0. As with edit distance, ERP is calculated through dynamic programming. The recursion equations for ERP are given as follows:

$$\begin{aligned} erp(i,j) = \min\{&d(A_i, B_j) + erp(i-1, j-1), \\ &d(A_i, g) + erp(i-1, j), \\ &d(B_i, g) + erp(i, j-1)\}. \end{aligned} \tag{17}$$

Here $d(A_i, B_j)$ is a distance measure between the elements in the sequence.

6.2.5 Edit Distance for Real Sequences

Like ERP, Edit Distance for Real Sequences (EDR) [COO05] is another edit distance based similarity measure. It incorporates a threshold similar to LCSS to match similar points. For two trajectories a and b, EDR defines a match function for a pair of points (p_i, p_j) where $p_i \in T_a$ and $p_j \in T_b$. If $|p_{i,x} - p_{j,x}| \leq threshold$ and $|p_{i,y} - p_{j,y}| \leq threshold$ then the match method will return true; therefore, the sub-cost between the points p_i and p_j is zero. Otherwise, the sub-cost will be one.

6.2.6 Applying the Similarity Measures

We have covered five different popular distance measures used for trajectories. We use these measures to implement an algorithm to predict the destination of each trajectory. Since the trajectories are discretised into discrete cells, for each trajectory we extract all those that start from the same source. Next, as we move and visit the next cells, we compute the similarity between the selected trajectory and each of the extracted trajectories. We sort the extracted trajectories based on their similarity. Finally, using a majority voting between the destinations of the first five trajectories

we predict the destination. The approach is described by Algorithm 6 which takes a trajectory T with length n as input.

Algorithm 6 Similarity-based Destination Prediction

1: $T_{set} \leftarrow$ `trajectories that start from the same source as T`
2: $n \leftarrow len(T_{set})$
3: *predictions* $\leftarrow$ `empty list`
4: **for** $i \leftarrow 2$ to n **do**
5: `similarities` $\leftarrow$ `empty list`
6: **for** $j \leftarrow 1$ to n **do**
7: **if** $len(T_{set,j}) \geq i$ **then**
8: $append(sim(T[1..i], T_{set,j}[1..i])$, destination of $T_{set,j})$ to similarities
9: `sort the similarities based on the similarity value`
10: `use majority voting between top five similarities to select the prediction at step i and append it to predictions`
11: return predictions

Here $sim(T_i[1..i], T_j[1..i])$ uses one of the five introduced similarity measures to compute distance. The algorithm takes trajectory T and starts to traverse each of its cells one by one. At each step, given the length of the traversed cells on the original trajectory, we extract the beginning of other trajectories. Therefore, at each step the length of all trajectories fed to the similarity function is equal. Then, one of the similarity measures is used to compute the similarity between the pairs. Finally, the top five trajectories with lowest distance are selected and using a majority voting between those a destination is predicted.

This algorithm can be used also to predict the next cell along a trajectory instead of the destinations. The only difference is at step 10 of Algorithm 6 where instead of doing a majority voting on destinations, we do it for the next visited cell in each trajectory.

6.3 Trajectory Prediction

By now, we have looked into how predictable paths are by creating a baseline where we tried to predict the destination of trajectories by only using frequency. Afterwards, we covered five popular distance measures for trajectories to provide a better destination prediction framework. Although, we did not take into account the fork-

points derived by trajectory segmentation as covered in the last chapter. In this section, we introduce an algorithm that utilise the fork-points for the prediction.

As discussed, our framework is capable of aggregating movement objects' trajectories and analysing their regularity. We extract the points where paths starts to diverge from each others as fork-points. As a result, a trajectory contains regular segments that are connected with each other at segmentation points. These points are mostly responsible for the amount of irregularity in the trajectories. Knowledge regarding these points results in the highest reduction in trajectory entropy. In other words, knowing that a moving object has passed these points gives the highest amount of information possible about the path that the object is taking. Therefore, it is possible to encode the entire trajectory by the segmentation points along the way.

Accordingly, we introduce Algorithm 7 that uses an encoded version of trajectories to predict the paths, and finally the destination. The algorithm is very similar to Algorithm 6, but instead of extracting the destination of the trajectories through majority voting, we extract the next fork-point along the path for each of the trajectories using majority voting. The algorithm takes the trajectory T_c and its fork-points TE_c as input. Then, it extracts all the trajectories starting from the same source T_{set} and their corresponding fork-points TE_i.

In this chapter we explored the predictability of trajectories and covered trajectory similarity measures. In addition, we introduced two algorithms that use the similarity measures to predict the destination of trajectories and also the fork-points along them. In the next chapter we will analyse and evaluate the performance of our algorithms.

Algorithm 7 Fork-point Prediction

$T_c \leftarrow$ `current trajectory`
$TE_c \leftarrow T_c$ `encoded by its fork-points`
$T_{set} \leftarrow$ `trajectories that start from the same source as` T_c
$n \leftarrow len(T_{set})$
for $i \leftarrow 1$ to n **do**
    $TE_i \leftarrow T_i$ `encoded by its fork-points`
$predictions \leftarrow$ `empty list`
for $i \leftarrow 2$ to n **do**
    $similarities \leftarrow$ `empty list`
    for $j \leftarrow 1$ to n **do**
        if $len(T_{set,j}) \geq i$ **then**
            $forkpoint_c \leftarrow$ `the first fork-point of` TE_c `that appears after i'th cell of trajectory` T_c
            $append(sim(T[1..i], T_{set,j}[1..i])$ `, first fork-point of` TE_j`) that appears after i'th cell of trajectory` $T_{set,j}$`) to similarities`
    `sort the similarities based on the similarity value`
    `use majority voting between top five similarities to select the next fork-point at step i and append it to predictions`
return predictions

7 Evaluation

This chapter describes our experimental setup used for evaluation. We explore the performance of the segmentation, compression, and prediction of the approaches introduced in this thesis. First, we evaluate the segmentation algorithms' compression power by looking into their compression rates. Then, we explore the effectiveness of our segmentation algorithm by comparing it to the conditional Markov entropy based segmentation algorithm. Afterwards, we evaluate the predictability of destinations and segments.

In order to perform the evaluations we selected a subset of trajectories from each dataset due to computational limitations. From each dataset, we select the most popular source cell where the highest amount of the trajectories have started from. After extracting the trajectories that start from the most popular source we have $11,888$ trajectories for the Cabspotting dataset, 731 trajectories for the Geolife dataset, and $6,208$ trajectories for the Taxi Service Trajectories.

Figure 22 shows a heatmap for the trajectories of each dataset that start from the most popular source. The figure demonstrates the number of visitations to each cell and the corresponding entropy value of each cell. It can be easily observed that close to the popular source cell the visitation frequency is high for the cells. As we get further away from the source of trajectories there are fewer visitations for each cell until we reach destinations that have been visited only few times. Regarding the cell entropies, it can be observed that there is no relationship between the cell visitations and entropy value. It again asserts our theory that there is an inherent regularity within how people move. The cells with high cell entropy values are the fork-points that segment the trajectories. Now that we covered the properties of the data used for evaluations we move onto the next sections.

7.1 Compression

We have covered two different algorithms that utilises the entropy information to do the trajectory segmentation. The first algorithm by [KGT15] uses conditional entropy of Markov trajectories. We also introduced our own segmentation algorithm that uses the cell entropies [FN16]. We use these algorithms to find the set of fork-points within each trajectory. By using the segmentation methods and detecting the fork-points, it is possible to compress and encode the trajectory with its fork-points. Figure 23 illustrates the compression power of our segmentation approach using cell

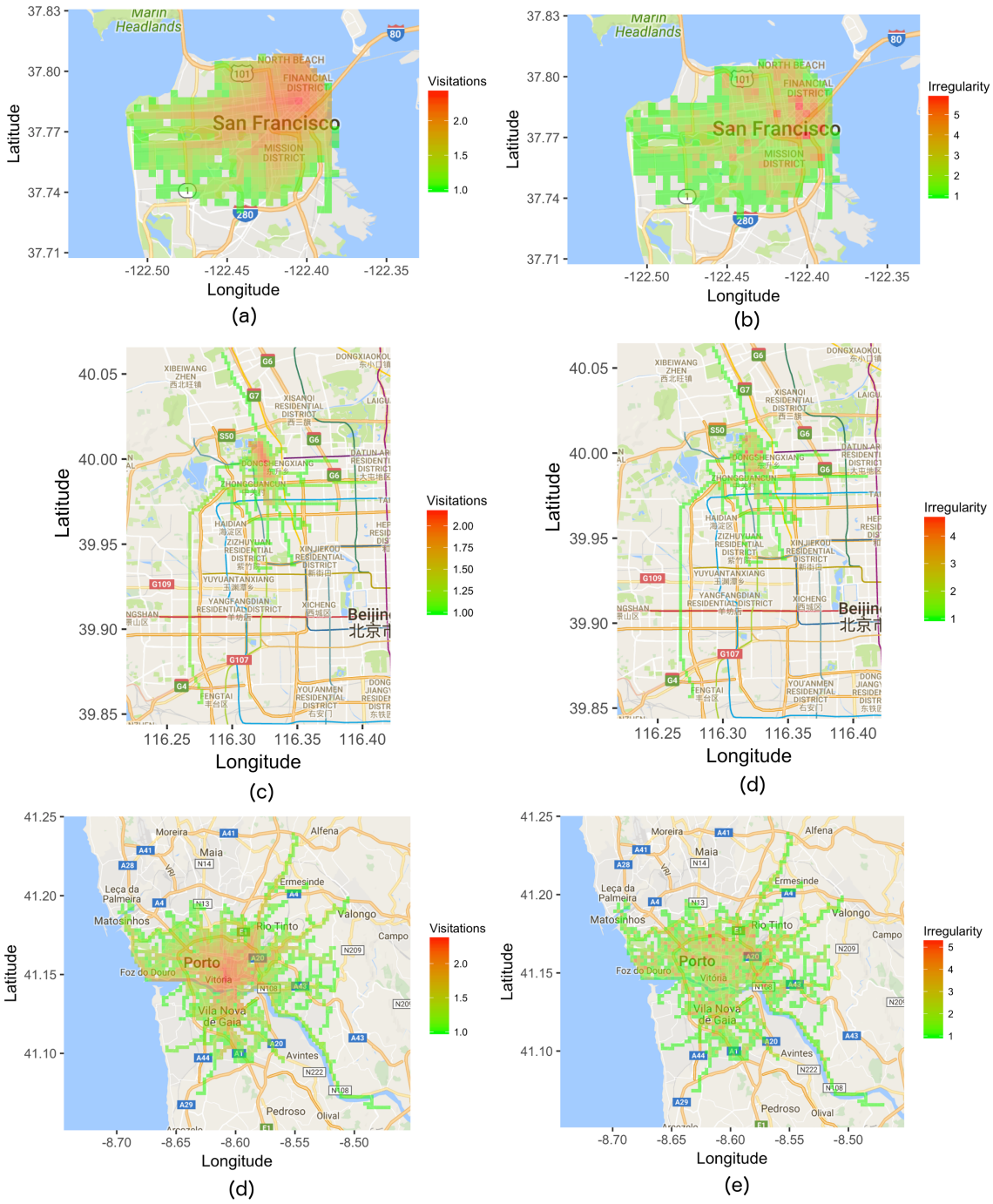

Figure 22: Heatmap of the cell visitation and entropy of the trajectories starting from the most popular source for each dataset.

entropies. It can be observed from the density plot that for most of the trajectories we can encode the trajectories with less than thirty percent of its points.

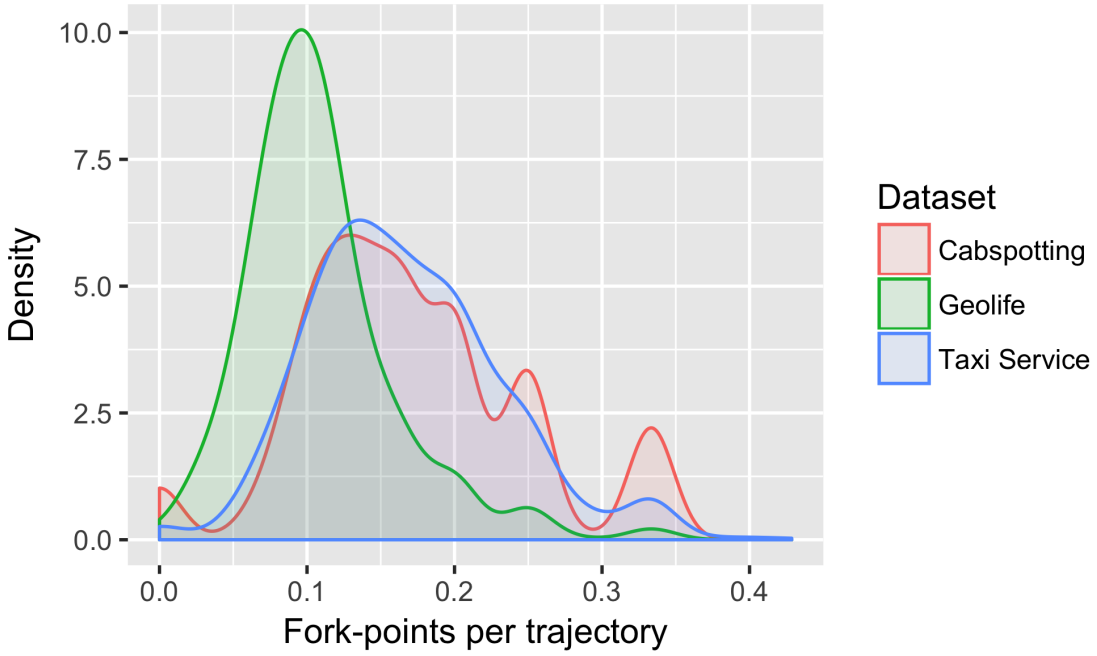

Figure 23: The density of fork-points per trajectory based on our segmentation approach using cell entropy.

On the other hand, by using the conditional entropy of Markov trajectories on the Cabspotting dataset we get the compression ratios as shown by Figure 24. As evident in the Figure, by using this segmentation approach we cannot achieve a good compression of the trajectories as the majority of points will remain as segmentation points. Therefore, by using our algorithm we can achieve higher compression rates.

7.2 Regularity and Segmentation

We have seen that by using the cell entropies we can achieve higher compression rates compared to the method which uses the conditional entropy of Markov trajectories. In addition to compression rate, we are interested in how effective our approach is in finding the fork-points. In other words, we would like to measure the effectiveness of our approach in detecting the points that are responsible for the highest amount of irregularities within trajectories. In order to achieve this objective, we computed the Markov trajectory entropy of every trajectory of the most popular source of Cabspotting dataset. In addition, we also detected the fork-points of each trajectory using our approach. Finally, we extracted the segmentation points as returned by

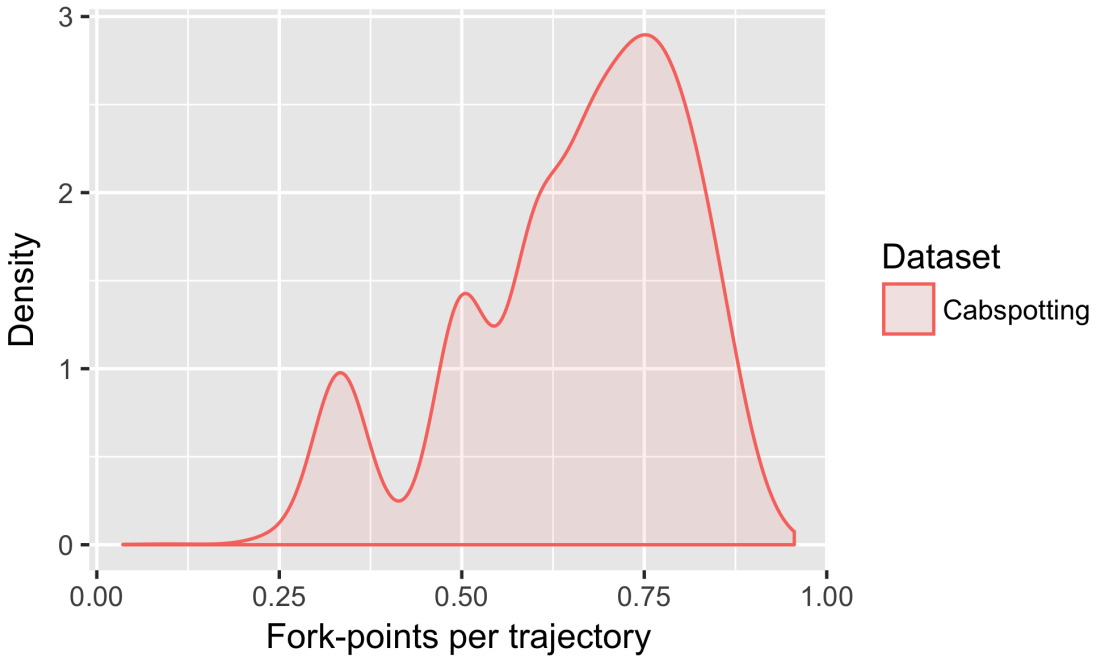

Figure 24: The density of fork-points per trajectory based on conditional entropy of Markov trajectories.

the conditional Markov trajectory entropy segmentation method.

Next, for each trajectory we only extract the points as returned by the conditional entropy method. These are the cells which will result in ratios of conditional entropies over total entropies of higher than one. Of these points, we randomly select one of our fork-points and one of the non fork-points of the trajectory. For the selected points, we compute the ratio of the conditional Markov trajectory entropy over the total entropy of the trajectory. Figure 25 illustrates the results. We can observe that our approach can effectively detect points that are responsible for higher irregularities within trajectories. Therefore, in addition to higher compression ratio our approach is also effective in finding the cells that cause the highest amount of irregularity within trajectories. In addition, our segmentation approach is computationally less expensive compared to the method that computes the conditional entropies.

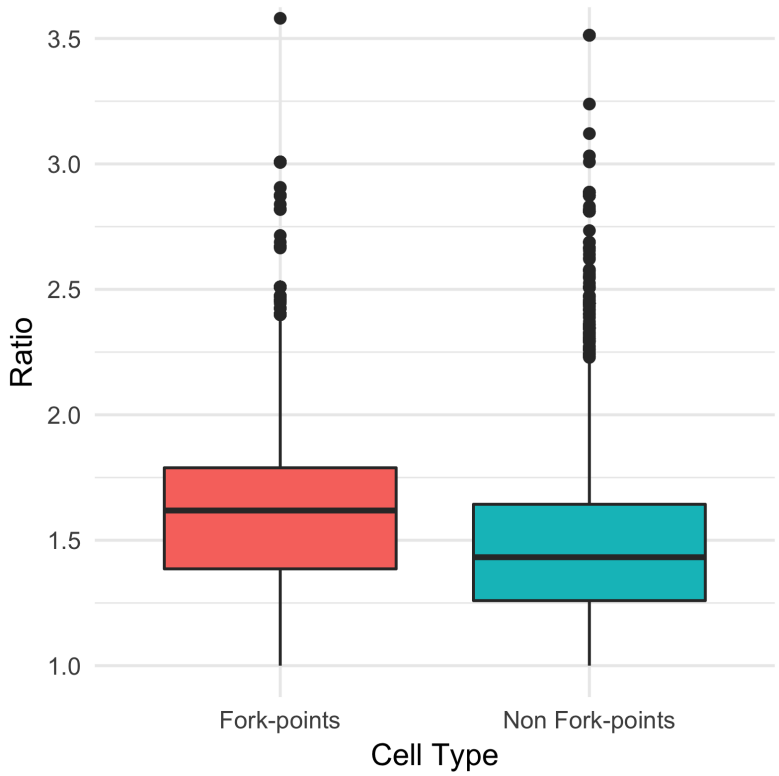

Figure 25: The effectiveness of our segmentation approach is measured by comparing it to the achieved ratios of conditional entropy over total entropy of trajectoriesf.

7.3 Destination Prediction

In the previous chapter, we have introduced five of the most popular trajectory similarity measures. In this section, we first explore and compare the effectiveness of the trajectory similarity measures to each other for predicting the destination. Afterwards, we select a smaller set of data and utilise the conditional entropy of Markov trajectories to predict the destination and see how it performs compared to the classical trajectory similarity measures.

In order to compare the similarity measures to each other we use Algorithm 6. In addition to calculating the accuracy for predicting the destination, we also compute how far the predicted destination is from the actual destination of the trajectories. Figures 26 and 27 display the results. Based on the figures, we can observe that the task of destination prediction is quite hard. In the beginning of the paths the accuracy is close to zero and as we move along the path to finally reach the destination we can only achieve an accuracy of 40% percent for Cabspotting, 60%

for Geolife, and 40% for the Taxi Service Trajectories. However, if we look into the predicted destination and compute how far they are from the actual destinations, we can see that the distance converges really fast to the minimum amount somewhere in the middle of paths. While this result is better compared to accuracy achieved by exact matching the predicted destination and the actual one, the distance error of 1000 meters is still quite large. Another interesting observation is regarding the performance of the different similarity measures. They seem to be not that different from each other. This can be due to the fact that we use discretised version of the trajectories and the their corresponding short lengths compared to GPS trajectories.

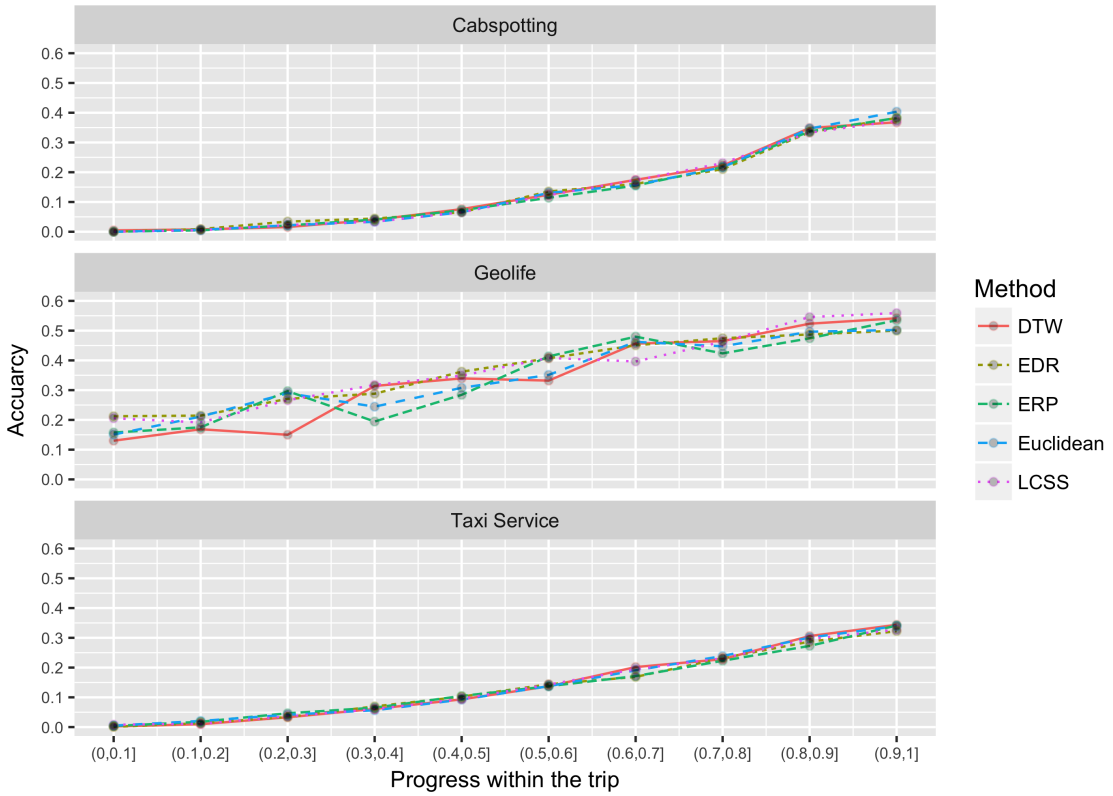

Figure 26: The accuracy of predicting the destination based on different similarity methods along the path.

In addition to the classic similarity measures for trajectories, we also used the conditional entropy of the Markov trajectories to predict the destination. Due to its high computational complexity, from the trajectories starting from the most popular source in each dataset, we only selected those that their destinations are in the top 10 most popular destinations from the most popular source to proceed.

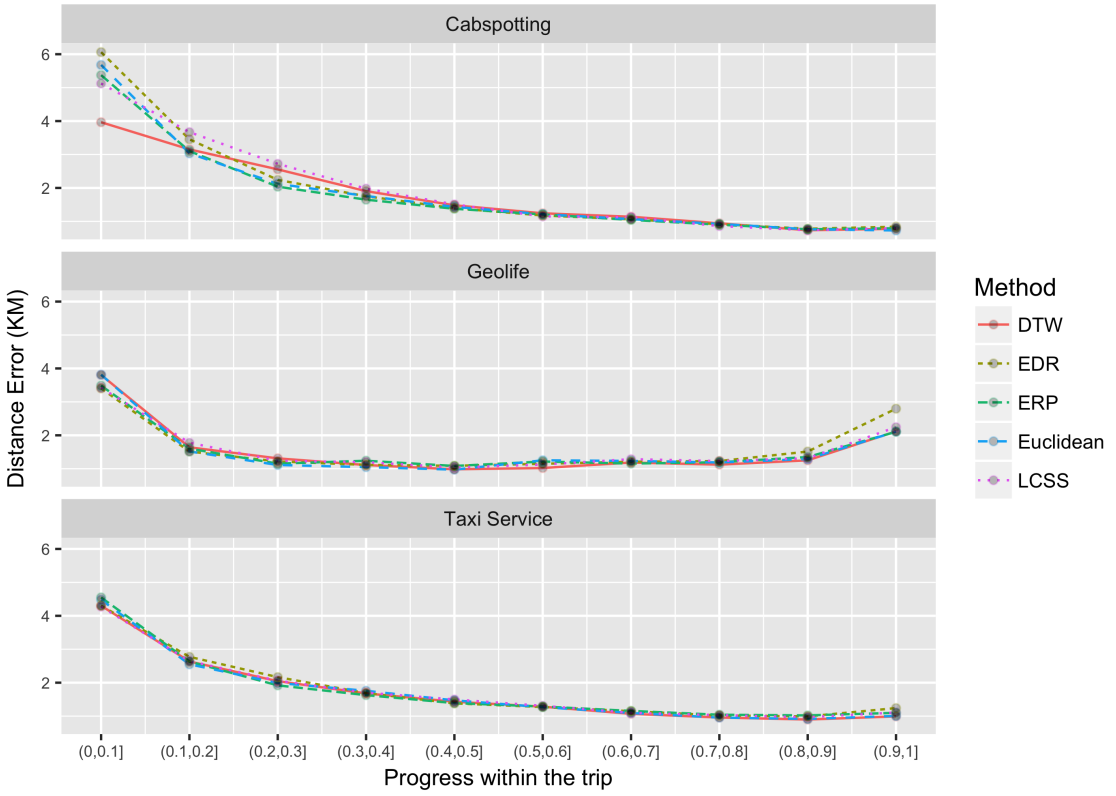

Figure 27: The distance from the predicted destination to the actual destination based on different similarity methods along the path.

At each step along the trajectories, we extract all the observed cells by then and condition on all the observed cells and compute the conditional entropy of the Markov trajectories. We select the one that has the lowest conditional entropy value for the prediction. Figures 28 and 29 display the results. We can observe that using the conditional entropy does not help the prediction task and it performs worse than state-of-the-art trajectory similarity measures. Therefore, the irregularities within the paths are not small enough for the conditional entropy to find the actual path.

7.4 Segment Prediction

In the previous sections, we explored how efficient our segmentation algorithm is for compressing the trajectories and how effective it is in finding the points that contain the highest amount of irregularities. We also looked into the task of predicting the destination of the trajectories using the trajectory similarity measures and conditional entropy values. As the last section of this chapter, we analyse how

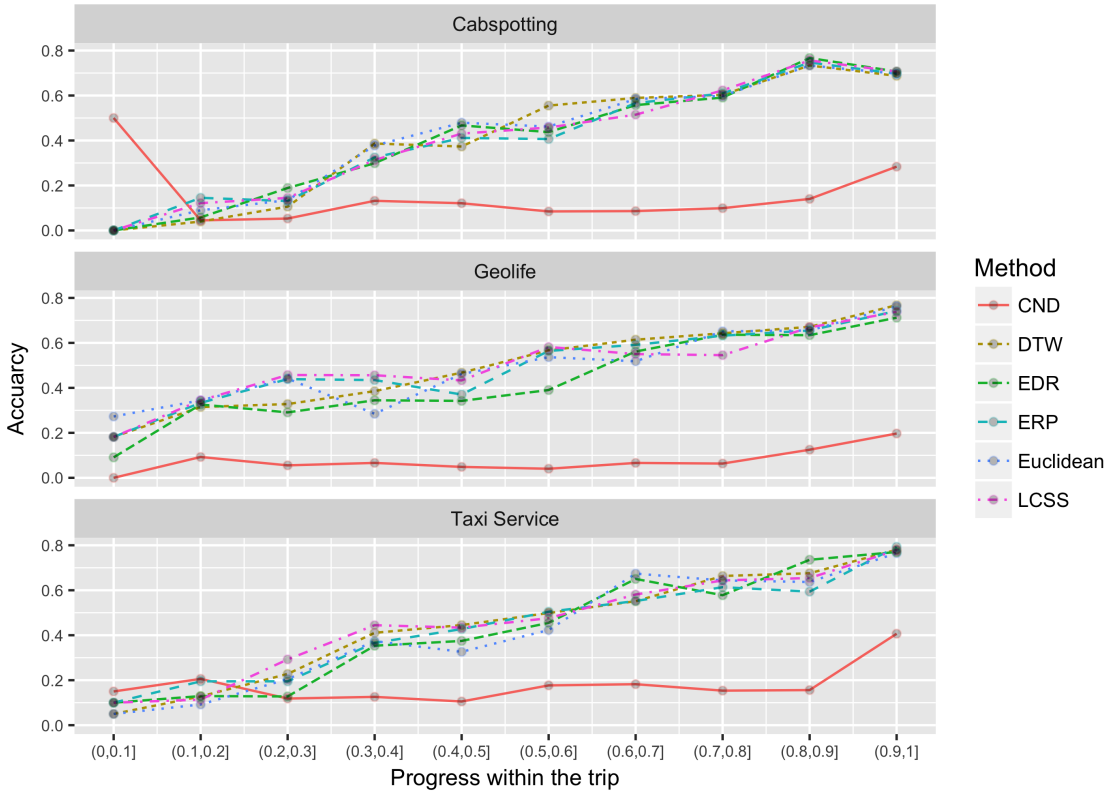

Figure 28: The accuracy of predicting the destination based on different similarity methods along the path.

segmenting the trajectory affects the prediction. Mainly, as we progress along the path we would like to predict the next fork-point on the way. In order to do this, we do the trajectory segmentation and use Algorithm 7 to predict the fork-points in a segment based manner.

Figure 30 shows the results of the fork-point predictions. We can observe that segmentation has significantly improved the accuracy compared to when we only select the most similar trajectories and return the destination of that trajectory as the prediction. As we progress along the path, the fork-point prediction accuracy increases and in general we reach an accuracy above 80%. Although the prediction accuracy is higher in the beginning compared to the destination prediction results, the accuracy is still relatively low. We can conclude that there is higher amount of uncertainty at beginning of the paths which makes the prediction difficult. This finding is line with the fact that we use a subset of data for evaluation which only contains the trajectories that start from the most popular source cell. As we progress

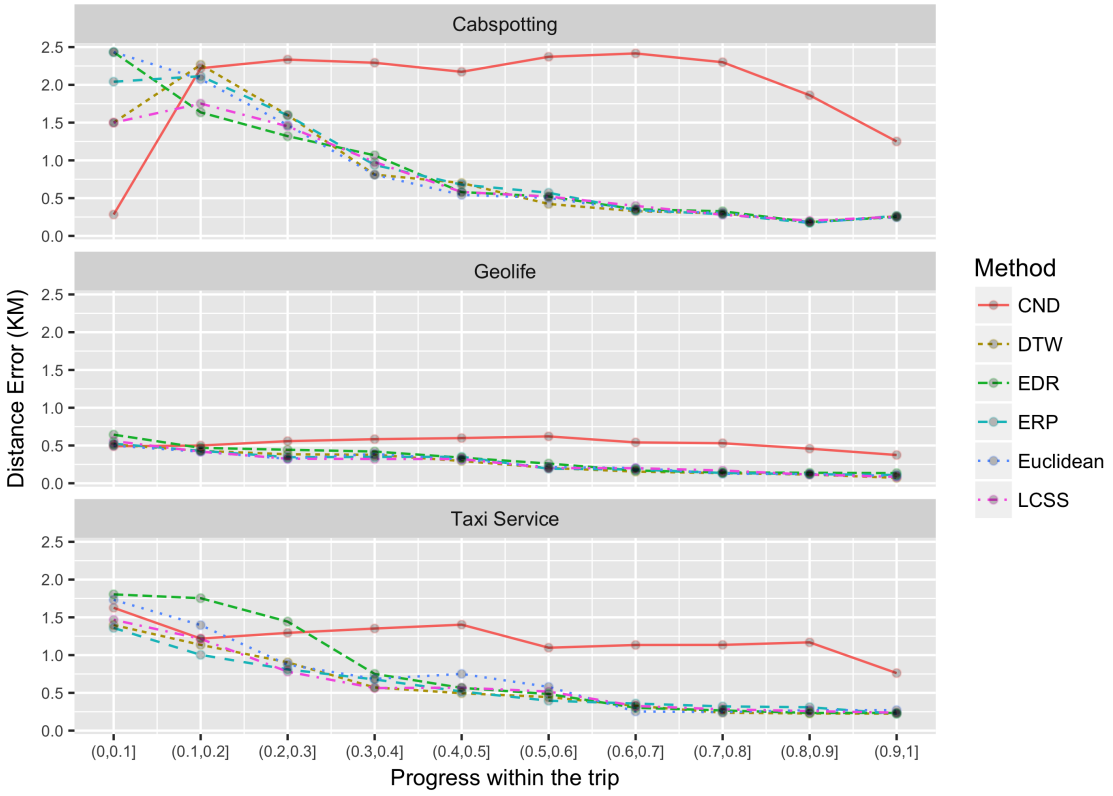

Figure 29: The distance from the predicted destination to the actual destination based on different similarity methods along the path.

along the path, there is less and less irregularity within the paths to reach the destination and that is why we get the better accuracies as we progress towards the destination. Another interesting observation is the performance of the similarity measures. In this case they all almost perform the same way. This is in line with our assumption in the last section that short lengths are most probably contribute to this behaviour.

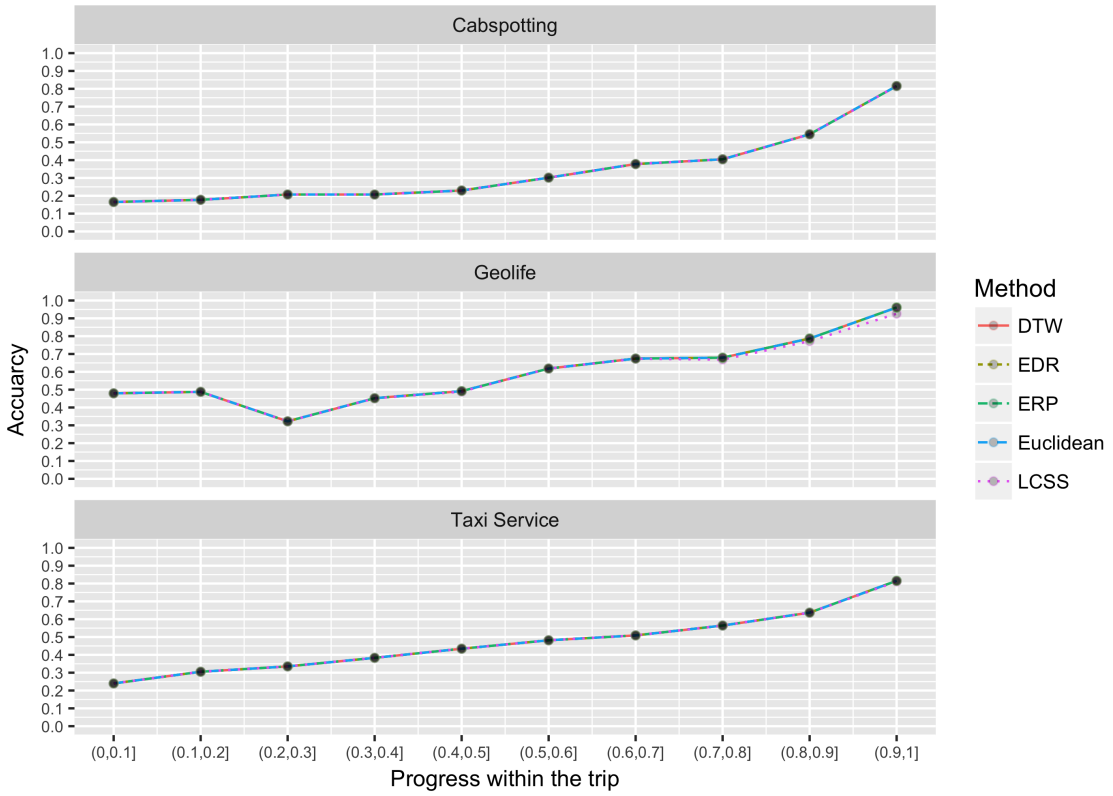

Figure 30: The accuracy of predicting the fork-points based on different similarity methods along the path.

8 Conclusion

In this thesis, we analysed the amount of regularity within trajectories and ways to utilise it for decomposing the trajectories into segments which can be used for encoding and prediction.

As our first contribution, we introduced a trajectory segmentation algorithm that utilises local entropy. Our approach only needs to compute the local entropy of cells; therefore, it has lower computational complexity compared to the approach that segments the trajectories using conditional entropy of Markov trajectories which requires costly matrix multiplications. We have shown that using our segmentation algorithm, we can encode the majority of trajectories with less than 30 percent of their points. In addition, we have shown that the extracted segmentation points are in fact the points along the trajectories that are responsible for most of the irregularity within the path.

The thesis has also investigated the predictability of trajectories. We described five popular trajectory similarity measures and showed that predicting the destination of a trajectory along the path using the similarity measures are difficult. We also have observed that the conditional entropy of Markov trajectories perform worse compared to the traditional trajectory similarity measures for predicting the destination. Finally, as our last contribution, we have shown that by decomposing the trajectories into segmentation points we can achieve better prediction accuracies. The effect of segmentation for prediction can be understood better when we compare the prediction accuracy around the ending of trajectories where for the traditional similarity measures we achieved at best an accuracy of 0.5 while using the segmentation points we achieve prediction accuracies above 0.8 for all datasets.

To evaluate our algorithms, we have selected three popular movement datasets in the human mobility research studies. These datasets are openly available which help the reproducibility of results. To be able to analyse and process trajectory data, we have covered methods for preprocessing them which are required for implementing systems that capture and store trajectories. In addition, postprocessing methods introduced to handle the erroneous and noisy observations. A limitation regarding these datasets is that two of them only contain cab traces. While the way that people move can be approximated with cab traces, they may lack some properties of general human movement. For example, the way that cabs navigate through the city may be different from how a private car would do. In addition, we do not have

any information regarding other types of movements such as walking. GeoLife is the only dataset that contains all types of transportations but compared to the other two datasets it has considerably less trajectories.

In order to use Markov models for measuring the amount of regularity, we used a grid model to discretise the trajectories and transform the area of observations into the states of a Markov chain. Markov chains enabled us to use regularity measures, specifically local entropy, entropy of the Markov trajectories, and conditional entropy of Markov trajectories. However, it should be mentioned that using a uniform resolution grid model causes some limitations. In case of big highways with multiple lanes or large traffic junctions, the mapping of the observations to the grid cells may not be accurate as the location may mistakenly be mapped to a neighbouring grid cell instead of the correct one. This issue causes lower overall regularity in the discretised system.

The proposed approach for modelling the people's movement can be used in other contexts other than the energy efficient tracking. Location Based Services (LBS) are one example where the methodology can be useful. There are two types of these services. Ones that are dependent on the location of the user in the current moment and those that require a trace of the object's movement. In either case, our approach can be used to model the movement and provide the system with predictions regarding the path and the whereabouts of the user. Spatial trajectories are also of high importance in the business analysis, city planning, and transportation.

The discussed methodology and system not only can be used in the context of movement and energy efficient tracking but also to any other real-world phenomena that can be modelled as a discrete Markov chain. Then, it is possible to use the segmentation methods to extract the irregular points.

Everyday, more mobile applications track their user's whereabouts by continuously sensing their locations. Sensing the user's location constantly is a costly action, and greatly affects the energy consumption of tracking devices. Such applications sense the locations and human movement trajectories continuously over an interval of time. As a result, more smartphone users choose to disable the tracking option in order to lower their energy consumption and save battery. On the other hand, human movement habits are regular to a high degree; therefore, continuously tracking the locations by using costly sensors is not efficient. It is possible to make use of this inherent regularity in their movement trajectories to predict path people take to reach their destinations, which will result in an energy proportional usage of

resources. After predicting the trajectory of the moving object, it is possible to reduce the usage of sensors proportionally. Therefore, accurate prediction of people's movement can result in efficient management of device resources.

As future work, we plan to look into real-time sampling techniques along individuals' trajectories as they move. More specifically, we are interested in providing an energy-efficient tracking solution that transform the trajectories into fork-points for future reference and as people move we accordingly change the sampling rate.

References

BA01 Blue, V. J. and Adler, J. L., Cellular automata microsimulation for modeling bi-directional pedestrian walkways. *Transportation Research Part B: Methodological*, 35,3(2001), pages 293 – 312. URL `http://www.sciencedirect.com/science/article/pii/S0191261599000521`.

BBKN15 Bhattacharya, S., Blunck, H., Kjærgaard, M. B. and Nurmi, P., Robust and energy-efficient trajectory tracking for mobile devices. *IEEE Transactions on Mobile Computing*, 14,2(2015), pages 430–443.

BGB11 Belik, V., Geisel, T. and Brockmann, D., Natural human mobility patterns and spatial spread of infectious diseases. *Phys. Rev. X*, 1, page 011001. URL `http://link.aps.org/doi/10.1103/PhysRevX.1.011001`.

BHG06 Brockmann, D., Hufnagel, L. and Geisel, T., The scaling laws of human travel. *Nature*, 439,7075(2006), pages 462–465. URL `http://dx.doi.org/10.1038/nature04292`.

CGS+09 Constandache, I., Gaonkar, S., Sayler, M., Choudhury, R. R. and Cox, L., Enloc: Energy-efficient localization for mobile phones. *INFOCOM 2009, IEEE*. IEEE, 2009, pages 2716–2720.

CJNP04 Civilis, A., Jensen, C. S., Nenortaite, J. and Pakalnis, S., Efficient tracking of moving objects with precision guarantees. *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*. IEEE, 2004, pages 164–173.

CN04 Chen, L. and Ng, R., On the marriage of lp-norms and edit distance. *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04. VLDB Endowment, 2004, pages 792–803, URL `http://dl.acm.org/citation.cfm?id=1316689.1316758`.

COO05 Chen, L., Özsu, M. T. and Oria, V., Robust and fast similarity search for moving object trajectories. *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, New

York, NY, USA, 2005, ACM, pages 491–502, URL http://doi.acm.org/10.1145/1066157.1066213.

DDFG01 Doucet, A., De Freitas, N. and Gordon, N., An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, Springer, 2001, pages 3–14.

DDLM13 De Domenico, M., Lima, A. and Musolesi, M., Interdependence and predictability of human mobility and social interactions. *Pervasive and Mobile Computing*, 9,6(2013), pages 798–807.

DP73 Douglas, D. H. and Peucker, T. K., Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10,2(1973), pages 112–122.

EC93 Ekroot, L. and Cover, T., The entropy of markov trajectories. *Information Theory, IEEE Transactions on*, 39,4(1993), pages 1418–1421.

FN16 Faghihi, F. and Nurmi, P., An empirical study on the regularity of route mobility. *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, UbiComp '16, New York, NY, USA, 2016, ACM, pages 1418–1425, URL http://doi.acm.org/10.1145/2968219.2968420.

FZ11 Fang, S. and Zimmermann, R., Enacq: energy-efficient gps trajectory data acquisition based on improved map matching. *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2011, pages 221–230.

Gel74 Gelb, A., *Applied optimal estimation*. MIT press, 1974.

GHB08 Gonzalez, M. C., Hidalgo, C. A. and Barabasi, A.-L., Understanding individual human mobility patterns. *Nature*, 453,7196(2008), pages 779–782. URL http://dx.doi.org/10.1038/nature06958.

GLPC12 Goh, S., Lee, K., Park, J. S. and Choi, M. Y., Modification of the gravity model and application to the metropolitan seoul subway system. *Phys. Rev. E*, 86, page 026102. URL http://link.aps.org/doi/10.1103/PhysRevE.86.026102.

Gus02 Gustavsen, R. M., Condor–an application framework for mobility-based context-aware applications. *Proceedings of the workshop on concepts and models for ubiquitous computing*, volume 39, 2002.

HNT13 Hemminki, S., Nurmi, P. and Tarkoma, S., Accelerometer-based transportation mode detection on smartphones. *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, New York, NY, USA, 2013, ACM, pages 13:1–13:14, URL `http://doi.acm.org/10.1145/2517351.2517367`.

IBC+12 Isaacman, S., Becker, R., Cáceres, R., Martonosi, M., Rowland, J., Varshavsky, A. and Willinger, W., Human mobility modeling at metropolitan scales. *Proceedings of the 10th international conference on Mobile systems, applications, and services*. Acm, 2012, pages 239–252.

JDLVDVV80 Jonker, R., De Leve, G., Van Der Velde, J. and Volgenant, A., Technical note-rounding symmetric traveling salesman problems with an asymmetric assignment problem. *Operations Research*, 28,3-part-i(1980), pages 623–627.

JWS08 Jung, W.-S., Wang, F. and Stanley, H. E., Gravity model in the korean highway. *EPL (Europhysics Letters)*, 81,4(2008), page 48005. URL `http://stacks.iop.org/0295-5075/81/i=4/a=48005`.

KBBN11 Kjærgaard, M. B., Bhattacharya, S., Blunck, H. and Nurmi, P., Energy-efficient trajectory tracking for mobile devices. *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 2011, pages 307–320.

KGT13 Kafsi, M., Grossglauser, M. and Thiran, P., The entropy of conditional markov trajectories. *Information Theory, IEEE Transactions on*, 59,9(2013), pages 5577–5583.

KGT15 Kafsi, M., Grossglauser, M. and Thiran, P., Traveling salesman in reverse: Conditional markov entropy for trajectory segmentation. *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 2015, pages 201–210.

KH90 Kearney, J. K. and Hansen, S., Stream editing for animation. Technical Report, DTIC Document, 1990.

KH06 Krumm, J. and Horvitz, E. *Predestination: Inferring Destinations from Partial Trajectories*, pages 243–260. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. URL http://dx.doi.org/10.1007/11853565_15.

KKK06 Kim, M., Kotz, D. and Kim, S., Extracting a mobility model from real user traces. *Proceedings of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Barcelona, Spain, April 2006, IEEE Computer Society Press, URL http://www.cs.dartmouth.edu/~dfk/papers/kim-mobility.pdf.

Kru06 Krumm, J., Real time destination prediction based on efficient routes. *SAE Technical Paper*. SAE International, 04 2006, URL http://dx.doi.org/10.4271/2006-01-0811.

Kru10 Krumm, J., Where will they turn: predicting turn proportions at intersections. *Personal and Ubiquitous Computing*, 14,7(2010), pages 591–599. URL http://dx.doi.org/10.1007/s00779-009-0248-1.

LDR11 Lange, R., Dürr, F. and Rothermel, K., Efficient real-time trajectory tracking. *The VLDB Journal*, 20,5(2011), pages 671–694. URL http://dx.doi.org/10.1007/s00778-011-0237-7.

LFDR09 Lange, R., Farrell, T., Durr, F. and Rothermel, K., Remote real-time trajectory simplification. *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*. IEEE, 2009, pages 1–10.

LFK07 Liao, L., Fox, D. and Kautz, H., Extracting places and activities from gps traces using hierarchical conditional random fields. *The International Journal of Robotics Research*, 26,1(2007), pages 119–134.

LHK+09 Lee, K., Hong, S., Kim, S. J., Rhee, I. and Chong, S., Slaw: A new mobility model for human walks. *IEEE INFOCOM 2009*, April 2009, pages 855–863.

LHK+12 Lee, K., Hong, S., Kim, S. J., Rhee, I. and Chong, S., Slaw: Self-similar least-action human walk. *IEEE/ACM Trans. Netw.*, 20,2(2012), pages 515–529. URL http://dx.doi.org/10.1109/TNET.2011.2172984.

LHL12 Lin, M., Hsu, W.-J. and Lee, Z. Q., Predictability of individuals' mobility with high-resolution positioning data. *Proceedings of the 2012*

ACM Conference on Ubiquitous Computing, UbiComp '12, New York, NY, USA, 2012, ACM, pages 381–390, URL http://doi.acm.org/10.1145/2370216.2370274.

LJH+14 Li, Y., Jin, D., Hui, P., Wang, Z. and Chen, S., Limits of predictability for large-scale urban vehicular mobility. *IEEE Transactions on Intelligent Transportation Systems*, 15,6(2014), pages 2671–2682.

LKLZ10 Lin, K., Kansal, A., Lymberopoulos, D. and Zhao, F., Energy-accuracy trade-off for continuous mobile device location. *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pages 285–298.

LPFK07 Liao, L., Patterson, D. J., Fox, D. and Kautz, H., Learning and inferring transportation routines. *Artificial Intelligence*, 171,5(2007), pages 311 – 331. URL http://www.sciencedirect.com/science/article/pii/S0004370207000380.

LWB+13 Lu, X., Wetter, E., Bharti, N., Tatem, A. J. and Bengtsson, L., Approaching the limit of predictability in human mobility. *Scientific reports*, 3.

LZX+08 Li, Q., Zheng, Y., Xie, X., Chen, Y., Liu, W. and Ma, W.-Y., Mining user similarity based on location history. *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '08, New York, NY, USA, 2008, ACM, pages 34:1–34:10, URL http://doi.acm.org/10.1145/1463434.1463477.

MCN11 Munjal, A., Camp, T. and Navidi, W. C., Smooth: A simple way to model human mobility. *Proceedings of the 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '11, New York, NY, USA, 2011, ACM, pages 351–360, URL http://doi.acm.org/10.1145/2068897.2068957.

MMGF+13 Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J. and Damas, L., Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems*, 14,3(2013), pages 1393–1402.

MMGF+16 Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J. and Damas, L., Time-evolving od matrix estimation using high-speed gps data streams. *Expert Systems With Applications*, 44, pages 275–288.

NB08 Nurmi, P. and Bhattacharya, S., Identifying meaningful places: The non-parametric way. *International Conference on Pervasive Computing*. Springer, 2008, pages 111–127.

NBK10 Nurmi, P., Bhattacharya, S. and Kukkonen, J., A grid-based algorithm for on-device gsm positioning. *Proceedings of the 12th ACM international conference on Ubiquitous computing*. ACM, 2010, pages 227–236.

NNS15 Nodari, A., Nurminen, J. and Siekkinen, M., Energy-efficient position tracking via trajectory modeling. *Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*. ACM, 2015, pages 33–38.

NW09 Ni, S. and Weng, W., Impact of travel patterns on epidemic dynamics in heterogeneous spatial metapopulation networks. *Phys. Rev. E*, 79, page 016111. URL `http://link.aps.org/doi/10.1103/PhysRevE.79.016111`.

PKG10 Paek, J., Kim, J. and Govindan, R., Energy-efficient rate-adaptive gps-based positioning for smartphones. *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pages 299–314.

PSDG09a Piorkowski, M., Sarafijanovic-Djukic, N. and Grossglauser, M., CRAWDAD dataset epfl/mobility (v. 2009-02-24), Downloaded from `http://crawdad.org/epfl/mobility/20090224`, February 2009.

PSDG09b Piorkowski, M., Sarafijanovic-Djukic, N. and Grossglauser, M., A parsimonious model of mobile partitioned networks with clustering. *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*. IEEE, 2009, pages 1–10.

PZCG14 Perera, C., Zaslavsky, A., Christen, P. and Georgakopoulos, D., Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials*, 16,1(2014), pages 414–454.

PZWS13 Pan, B., Zheng, Y., Wilkie, D. and Shahabi, C., Crowd sensing of traffic anomalies based on human mobility and social media. *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, New York, NY, USA, 2013, ACM, pages 344–353, URL `http://doi.acm.org/10.1145/2525314.2525343`.

QLL+11 Qi, G., Li, X., Li, S., Pan, G., Wang, Z. and Zhang, D., Measuring social functions of city regions from large-scale taxi behaviors. *2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, March 2011, pages 384–388.

RSH+11 Rhee, I., Shin, M., Hong, S., Lee, K., Kim, S. J. and Chong, S., On the levy-walk nature of human mobility. *IEEE/ACM Trans. Netw.*, 19,3(2011), pages 630–643. URL `http://dx.doi.org/10.1109/TNET.2011.2120618`.

RZL+11 Ramos, H. S., Zhang, T., Liu, J., Priyantha, N. B. and Kansal, A., LEAP: a low energy assisted GPS for trajectory-based services. *Proceedings of the 13th International Conference on Ubiquitous Computing (Ubicomp)*, 2011.

RZL+12 Rao, W., Zhao, K., Lagerspetz, E., Hui, P. and Tarkoma, S., Energy-aware keyword search on mobile phones. *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, New York, NY, USA, 2012, ACM, pages 59–64, URL `http://doi.acm.org/10.1145/2342509.2342522`.

SBBK+11 Scott, J., Bernheim Brush, A., Krumm, J., Meyers, B., Hazas, M., Hodges, S. and Villar, N., Preheat: Controlling home heating using occupancy prediction. *Proceedings of the 13th International Conference on Ubiquitous Computing*, UbiComp '11, New York, NY, USA, 2011, ACM, pages 281–290, URL `http://doi.acm.org/10.1145/2030112.2030151`.

SKJH06 Song, L., Kotz, D., Jain, R. and He, X., Evaluating next-cell predictors with extensive wi-fi mobility data. *IEEE Transactions on Mobile Computing*, 5,12(2006), pages 1633–1649.

SKWB10 Song, C., Koren, T., Wang, P. and Barabasi, A.-L., Modelling the scaling properties of human mobility. *Nat Phys*, 6,10(2010), pages 818–823. URL `http://dx.doi.org/10.1038/nphys1760`.

SMM+11 Scellato, S., Musolesi, M., Mascolo, C., Latora, V. and Campbell, A. T., Nextplace: a spatio-temporal prediction framework for pervasive systems. *International Conference on Pervasive Computing*. Springer, 2011, pages 152–169.

SQBB10 Song, C., Qu, Z., Blumm, N. and Barabási, A.-L., Limits of predictability in human mobility. *Science*, 327,5968(2010), pages 1018–1021.

SR88 Soong, F. K. and Rosenberg, A. E., On the use of instantaneous and transitional spectral information in speaker recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36,6(1988), pages 871–879.

ST94 Schilit, B. N. and Theimer, M. M., Disseminating active map information to mobile hosts. *IEEE network*, 8,5(1994), pages 22–32.

SWGB14 Smith, G., Wieser, R., Goulding, J. and Barrack, D., A refined limit on the predictability of human mobility. *Pervasive Computing and Communications (PerCom), 2014 IEEE International Conference on*. IEEE, 2014, pages 88–94.

YNLK06 Yoon, J., Noble, B. D., Liu, M. and Kim, M., Building realistic mobility models from coarse-grained traces. *Proceedings of the 4th International Conference on Mobile Systems, Applications and Services*, MobiSys '06, New York, NY, USA, 2006, ACM, pages 177–190, URL `http://doi.acm.org/10.1145/1134680.1134699`.

YZX12 Yuan, J., Zheng, Y. and Xie, X., Discovering regions of different functions in a city using human mobility and pois. *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, New York, NY, USA, 2012, ACM, pages 186–194, URL `http://doi.acm.org/10.1145/2339530.2339561`.

ZHL06 Zheng, Q., Hong, X. and Liu, J., An agenda based mobility model. *39th Annual Simulation Symposium (ANSS'06)*, April 2006, pages 8 pp.–.

ZLC+08 Zheng, Y., Li, Q., Chen, Y., Xie, X. and Ma, W.-Y., Understanding mobility based on gps data. *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 2008, pages 312–321.

ZLH13 Zheng, Y., Liu, F. and Hsieh, H.-P., U-air: When urban air quality inference meets big data. *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, New York, NY, USA, 2013, ACM, pages 1436–1444, URL `http://doi.acm.org/10.1145/2487575.2488188`.

ZLJG13 Zhang, L., Liu, J., Jiang, H. and Guan, Y., Senstrack: Energy-efficient location tracking with smartphone sensors. *IEEE sensors journal*, 13,10(2013), pages 3775–3784.

ZLYX11 Zheng, Y., Liu, Y., Yuan, J. and Xie, X., Urban computing with taxicabs. *Proceedings of the 13th International Conference on Ubiquitous Computing*, UbiComp '11, New York, NY, USA, 2011, ACM, pages 89–98, URL `http://doi.acm.org/10.1145/2030112.2030126`.

ZMH+15 Zhao, K., Musolesi, M., Hui, P., Rao, W. and Tarkoma, S., Explaining the power-law distribution of human mobility through transportation modality decomposition. *Scientific Reports*, 5, pages 9136 EP –. URL `http://dx.doi.org/10.1038/srep09136`.

ZXM10 Zheng, Y., Xie, X. and Ma, W.-Y., Geolife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.*, 33,2(2010), pages 32–39.

ZZXM09 Zheng, Y., Zhang, L., Xie, X. and Ma, W.-Y., Mining interesting locations and travel sequences from gps trajectories. *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pages 791–800.