Titre: Title:	Weakly Supervised Performance Evaluation of Trajectory Clustering
Auteur: Author:	Mohsen Rezaie
Date:	2021
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Rezaie, M. (2021). Weakly Supervised Performance Evaluation of Trajectory Clustering [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/6336/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/6336/
Directeurs de recherche: Advisors:	Nicolas Saunier
Programme: Program:	Génie civil

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Weakly Supervised Performance Evaluation of Trajectory Clustering

MOHSEN REZAIE
Département de génies Civil, Géologique et des Mines (CGM)

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie civil

Mai 2021

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Weakly Supervised Performance Evaluation of Trajectory Clustering

présenté par **Mohsen REZAIE**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Martin TRÉPANIER, président
Nicolas SAUNIER, membre et directeur de recherche
Bruno AGARD, membre

DEDICATION

To my parents,
for what you did for me is beyond words . . .

ACKNOWLEDGEMENTS

First, I would like to thank my research supervisor, Nicolas Saunier, for his guidance, patience and and above all his availability throughout this project. Also thanks to NSREC for offering Discovery Grants and Discovery Accelerator Supplements programs which funded this project.

I would also like to express my gratitude to all my friends and colleagues offering me their love and support throughout this research while away from my family and hometown. Thanks to Mahdi, my friend and now roommate, for his overwhelming support during the last few weeks of my research and always being there, figuratively and literally!

Especial thanks to Ali, my brother, best friend and mentor throughout my life, who has always been just a call away.

Finally, I express my immense gratitude to my parents, for what words cannot describe.

RÉSUMÉ

Avec l'avènement des nouvelles technologies au cours des dernières décennies, les techniques de collecte de données ont évolué et changé considérablement. Des appareils moins chers et plus récents ont facilité les tâches de collecte de données, tandis que la culture de la science ouverte a contribué à la disponibilité des ensembles de données collectées. D'autre part, les innovations technologiques et les méthodes d'analyse en constante amélioration ont facilité la tâche de traitement des données. Si ces changements technologiques contribuent à une évolution de l'analyse des données, les capacités humaines restent limitées. Par conséquent, les tâches manuelles peuvent potentiellement ralentir ce processus.

Les méthodes d'apprentissage non supervisées constituent une solution pour résoudre le problème de la gestion et de l'analyse des mégadonnées. Les algorithmes de groupement ("clustering"), comme sont appelées les méthodes d'apprentissage non supervisées qui traitent de l'assignation des données à des catégories, jouent ici un rôle clé.

Les études de circulation sont un domaine qui a largement utilisé ces algorithmes. Les trajectoires, en tant que source majeure d'informations dans les études de circulation, ont vu de nombreuses applications des méthodes de groupement. Cependant, dans la plupart des cas, l'évaluation des performances de ces méthodes a été négligé.

Dans ce contexte, ce projet de recherche se concentre sur le problème de l'évaluation non supervisée du groupement de trajectoires. L'objectif principal de ce projet de recherche était de développer un cadre pour les méthodes de groupement qui fonctionnerait entièrement sans l'étiquetage manuel des trajectoires. L'objectif secondaire était de procéder à une comparaison entre mesures de distance pour trajectoires et entre algorithmes de groupement.

Pour répondre à l'objectif principal, un cadre, avec une combinaison de mesures de performance à la fois non supervisées et supervisées, est développé pour comparer les configurations de groupement. En outre, une méthode est proposée pour générer automatiquement des groupes de trajectoire de référence en fonction de leurs points d'origine et de destination qui sont utilisés par des mesures de performance supervisées. L'ensemble de la procédure fonctionne sans supervision à la fois dans les étapes de groupement et d'évaluation.

Enfin, une tâche de comparaison est menée entre les mesures de distance populaires, les algorithmes de clustering et les mesures d'évaluation. La comparaison se compose de six mesures populaires de distance pour les trajectoires (c'est-à-dire "Dynamic Time Warping" (DTW), "Longest Common Sub Sequence" (LCSS), "Edit Distance on Real sequence" (EDR),

Piciarelli-Foresti (PF), Hausdorff et “Symmetric Segment-Path Distance” (SSPD)), dont trois ont été essayées avec six valeurs de paramètres, six algorithmes de groupement (c’est-à-dire “k-Medoids”, “k-Means”, “agglomerative hierarchical”, “spectral”, DBSCAN et OPTICS), ainsi que différents nombres de groupes allant de 2 à 30. Sept mesures de performance sont prises en compte dans la comparaison, à savoir l’exhaustivité, l’homogénéité, la validité, l’“Adjusted Mutual Information” (AMI), l’“Adjusted Rand Index” (ARI), le “Fowlkes-Mallows Index” (FMI) et la Silhouette. Les comparaisons indiquées sont basées sur les données de sept intersections dans deux villes différentes.

Les résultats montrent qu’il n’y a pas de configurations de groupement spécifique et même de mesures de distance ou d’algorithmes de groupement spécifique qui surpassent les autres systématiquement ou du moins systématiquement apparaît parmi les dix meilleures configurations pour toutes les intersections. Les résultats montrent qu’il y a des gagnants clairs pour des sites spécifiques. Chaque site a un, et jusqu’à trois, algorithmes ou distances surpassant les autres. Cependant, il n’y a pas de tendance claire dans l’ensemble des sites.

ABSTRACT

With the advent of new technologies in the past few decades, data collection techniques have evolved and changed significantly. Cheaper and newer devices have made data collection tasks easier, while open source culture has contributed to the availability of the collected datasets. On the other hand, technological innovations and ever-improving analytical methods have facilitated the data processing task. While these technological changes are contributing to an evolution in the data analysis, human capabilities remain limited. Therefore, manual tasks can potentially slow down this process.

One solution to address the problem of managing and analyzing big data is unsupervised learning methods. Clustering algorithms, as unsupervised learning methods which deal with categorical assignment of data are called, play a key role here.

Traffic studies is an area that has made extensive use of these algorithms. Trajectories, as a major source of information in traffic studies, have been the target of many studies that considered the application of clustering. However, in most cases the performance evaluation aspect of clustering has been overlooked.

In this context, this research project focuses on the unsupervised evaluation of trajectory clustering. The primary objective of this research project was to develop a clustering framework that would work entirely without the manual labeling of trajectories. The secondary objective was to conduct a comparison between popular trajectory distance measures and clustering algorithms.

To address the primary objective, a framework, with a combination of both unsupervised and supervised performance measures is developed that compares the clustering setups. Also, a method is proposed to automatically generate reference clusters based on their origin and destination points to be used by supervised performance measures. The entire procedure is unsupervised both in the clustering and evaluation steps.

Finally, a comparison is conducted between popular distance measures, clustering algorithms and evaluation measures. The comparison consists of six popular trajectory distance measures (i.e. Dynamic Time Warping (DTW), Longest Common Sub-Sequence (LCSS), Edit Distance on Real sequence (EDR), Piciarelli-Foresti (PF), Hausdorff and Symmetric Segment-Path Distance (SSPD)), three of them tried with six parameter values, six clustering algorithms (i.e k-Medoids, k-Means, agglomerative hierarchical, spectral, DBSCAN and OPTICS), as well as different numbers of clusters ranging from 2 to 30. Seven performance metrics are

considered in the comparison, namely, completeness, homogeneity, validity, Adjusted Mutual Information (AMI), Adjusted Rand Index (ARI), Fowlkes-Mallows Index (FMI) and Silhouette. The noted comparisons are done based on data from seven intersections in two different cities.

The results show that there is no single clustering setup and not even a specific distance measure or clustering algorithm that outperforms the others systematically or at least consistently appears among the top ten best clustering setups for all intersections. The results show that there are clear winners for specific sites. Each site has one, and up to three, algorithms or distances outperforming the others. However, there is no clear pattern in these observations across the sites.

TABLE OF CONTENTS

DEDICATION iii

ACKNOWLEDGEMENTS iv

RÉSUMÉ v

ABSTRACT vii

TABLE OF CONTENTS ix

LIST OF TABLES xii

LIST OF FIGURES xiii

LIST OF SYMBOLS AND ACRONYMS xv

CHAPTER 1 INTRODUCTION 1
1.1 Background 1
1.2 Thesis Objectives 2
1.3 Thesis Organization 2

CHAPTER 2 LITERATURE REVIEW 4
2.1 Video Surveillance 4
2.2 Pattern Recognition 5
2.2.1 Learning Types 5
2.2.2 Supervision Types 5
2.2.3 Time-series Analysis 6
2.3 Trajectory Clustering 7
2.3.1 Distance Measures 7
2.3.2 Clustering Algorithms 11
2.3.3 Unsupervised Performance Measures 12
2.3.4 Supervised Performance Measures 13
2.3.5 Previous Trajectory Clustering Studies 15

CHAPTER 3 RESEARCH PROJECT APPROACH 18
3.1 Framework 18

3.2 Data 20
3.2.1 NGSIM Dataset 20
3.2.2 inD Dataset 21
3.2.3 Data Quality 26

CHAPTER 4 Article 1: Trajectory Clustering Performance Evaluation: If We Know The Answer, It's Not Clustering 28
4.1 Introduction 28
4.2 Related Work 30
4.3 Methodology 31
4.3.1 Clustering Setup 31
4.3.2 Unsupervised Performance Measure 34
4.3.3 Weak Trajectory Clustering Supervision using Origin-Destination . . 35
4.4 Experimental Results 39
4.4.1 Datasets 39
4.4.2 Performance Plots 41
4.4.3 Performance Measure Combination 42
4.5 Conclusion 43

CHAPTER 5 METHODOLOGICAL ASPECTS AND COMPLEMENTARY RESULTS 48
5.1 Some Details On Implementation 48
5.1.1 Database Management System 48
5.1.2 Low-level Compilation And Parallelization 49
5.1.3 Caching The Objects 49
5.2 Further Results 50
5.2.1 Correlation Of The Distance Measures 50
5.2.2 Sensitivity To Data Permutation 50
5.3 Extra Experiments Using Centroid-based Algorithms And Performance Measures 52

CHAPTER 6 GENERAL DISCUSSION 56
6.1 Trajectory Clustering Framework 56
6.2 Comparison Of Clustering Setups 57

CHAPTER 7 CONCLUSION 58
7.1 Summary Of Works 58
7.2 Limitations 59

7.3 Future Research 60

REFERENCES 61

APPENDICES 68

LIST OF TABLES

3.1	Parameter values of DBSCAN for different distance measures	19
4.1	Similarity measures	32
4.2	Clustering algorithms	33
4.3	Evaluation metrics	39
4.4	Site summary with their characteristics (multiple synced cameras were mounted on buildings for the NGSIM datasets).	41

LIST OF FIGURES

3.1 The overview of the framework 18
3.2 Locations of data collection sites for NGSIM data in Atlanta, Georgia 21
3.3 NGSIM data collection sites 22
3.4 Locations of data collection sites for NGSIM data in Aachen, Germany 23
3.5 Aerial photo of Bendplatz intersection (inD 1), Aachen, Germany . . 24
3.6 Aerial photo of Frankenburg intersection (inD 2), Aachen, Germany . 25
3.7 Aerial photo of Frankenburg intersection (inD 3), Aachen, Germany . 26
4.1 Example of trajectories D in an intersection after clustering. Clusters are represented by different colors (in some cases the colors assigned to different clusters are similar due to the limit on number of colors while maintaining the contrast high enough to keep them visually distinctive). 33
4.2 Example of error in clustered trajectories and their silhouette values . 35
4.3 Average distances for the datasets of origins (left) and destinations (right) with respect to the number of clusters. The mean plus and minus two standard deviations are computed over several replications of the clustering algorithm for each number of clusters. The graphs are produced based on data from an intersection in NGSIM dataset using the agglomerative hierarchical clustering algorithm with average linkage as the merging criterion. 36
4.4 Trajectory origin (left) and destination (right) clusters with $k_O = 8$, $k_D = 4$ and agglomerative hierarchical clustering algorithm with average linkage. 37
4.5 An example of intersection for each of the two datasets 45
4.6 Performance results for the *NGSIM 1* intersection based on EDR similarity measure with parameter $w = 7$ m 46
4.7 Correlation of the performance measures for the *NGSIM 1* intersection 46
4.8 Top performing similarity measures (left) and trajectory clustering algorithms (right) on different sites 47
5.1 Correlation between distances measures with different parameter values 51
5.2 Performance variation of agglomerative clustering paired with LCSS ($d_z = 5\ m$) on NGSIM 2 52
5.3 Performance variation of agglomerative clustering paired with LCSS ($d_z = 5\ m$) on inD 1 53

5.4 Top performing similarity measures (left) and trajectory clustering algorithms (right) on different sites 55
A.1 Correlations between different performance measures for each intersection 68

LIST OF SYMBOLS AND ACRONYMS

TIS	Systèmes de Transport Intelligents
ITS	Intelligent Traffic Systems
GNSS	Global Navigation Satellite System
LiDAR	Light Detection and Ranging
SSD	Solid-State Drive
NPU	Neural Processing Unit
ANPR	Automatic Number Plate Recognition
ALPR	Automatic License Plate Recognition
DISSIM	Dissimilarity metric
ED	Euclidean Distance
DTW	Dynamic Time Warping
LCSS	Longest Common Sub-Sequence
EPR	Edit distance with Real Penalty
EDR	Edit Distance on Real sequence
SWAle	Sequence Weighted Alignment model
FCM	Fuzzy C-Means
RI	Rand Index
DP	Douglas-Peucker
DBSCAN	Density-Based Clustering of Application with Noise
LSTM	Longest Common Sub-Sequences
WAPE	Weighted Average Percentage Error
GMM	Gaussian Mixture Models
DBI	Davies-Bouldin Index
CHI	Calinski-Harabasz Index
SQL	Structured Query Language
RAM	Random Access Memory
PF	Piciarelli-Foresti
SSPD	Symmetric Segment-Path Distance
OPTICS	Ordering Points To Identify The Clustering Structure
FMI	Fowlkes-Mallows Index
AMI	Adjusted Mutual Information
ARI	Adjusted Rand Index
NGSIM	Next Generation SIMulation

US DOT	United States Department of Transportation
FHWA	Federal Highway Administration
inD	intersection Drone
VRU	Vulnerable Road User
RDBMS	Relational DataBase Management System

CHAPTER 1 INTRODUCTION

1.1 Background

Measuring and data collection is the basis of quantitative studies in different transportation tasks, such as transportation planning, design and construction, transportation management, as well as traffic related studies. Traffic studies range from congestion management, safety analysis, calculating road capacity and level of service, traffic monitoring to road design and all of these fields require knowledge on movement of road users in road network. Even law enforcement tasks can benefit from road user movement data to automate or enhance the law enforcement procedures. A popular method of collecting this data is recording time-stamped locations of road users during the span of data collection and generating time-ordered sequences of them that are called trajectories.

Especially with the advent of new technologies, user movement data collection is becoming more and more easy and accessible. Global Navigation Satellite System (GNSS) devices, smartphones, regular and infrared cameras, Light Detection and Ranging (LiDAR) and even communication networks such as Bluetooth, WiFi and cellular networks can be used to collect trajectory data (though the latter might no be as high quality as the rest of them). With the advancement and improvement of these technologies quality and reliability of data collection improves [1]. In addition, some problems that are present in active data collection can be avoided through passive and non-intrusive data collection [1, 2].

The evolution of technology also results into the availability of these devices in locations that could not easily be used before and leads to an ever-growing coverage of transport networks. It is also worth mentioning that these technological improvements when added to the growing popularity of open-source culture, results into massive amounts of data becoming accessible to researchers (and the public).

Advancement in technologies has also improved the processing tasks and increased the speeds through faster devices, raising the limits, introducing new devices, such as Solid-State Drive (SSD) and Neural Processing Unit (NPU). Furthermore, it has reduced the installation complexity and costs of the devices. This provides the opportunity to perform data analyses task easier and better than any time before.

On the other hand, we as human beings are limited in our capabilities. We may find new ways to manage these capabilities to push our limits and improve our performance and our skills, but we cannot escape these limits in the same way that we do with other technological

objects through changes that sometimes may change their essence. Therefore, in order to keep up with the pace of changes in data collection and analysis, we need to lower the required supervision and interaction of human operators, even if we might not be able to completely eliminate it. In addition to speeding up the analysis procedure, it will also lower the labor burden and cost.

A crucial step towards decreasing the need for human supervision in data analysis is using unsupervised learning algorithms: "unsupervised (learning) algorithms are those that experience only features but not a supervision signal", also called label or target [3]. In many cases, the problem can be pictured in the form of splitting objects or observations into different groups. Such unsupervised tasks are referred as clustering.

Even though the application of clustering methods in trajectory analysis is well studied (discussed in Chapter 2), avoiding use of labels for performance evaluations is overlooked.

1.2 Thesis Objectives

The proposed research project aims to investigate the problem of trajectory clustering evaluation. Specifically, the focus will be on performance evaluation with an unsupervised approach. It is divided more specifically into a primary and secondary objective:

Primary objective: developing a framework to evaluate and compare distance measures and clustering methods without requiring manually-labeled trajectories. The sub-objectives are:

- The framework should evaluate and compare a desired set of candidate clustering setups (i.e. combination of trajectory distances, clustering methods and their parameters) without the need for manual data labeling, and return the top performing methods;
- In addition to regular unsupervised performance measure(s), which do not require reference labels, a method should be proposed and implemented to extract reference labels from trajectories to be used by supervised performance measures.

Secondary objective: exploring the performance of popular clustering setups on different datasets. The comparison should be carried out with different distances and clustering algorithms with a range of parameter values.

1.3 Thesis Organization

The remainder of this thesis is organized as follows:

- In Chapter 2, we briefly introduce different learning approaches. Then we introduce different components of trajectory clustering and review prior literature on trajectory clustering.
- In Chapter 3, we explain the outline of the workand present an overview about the two different datasets used for the experiments.
- Chapter 4 contains the main body of the study in the form of a paper submitted for publication in the journal IEEE Transactions on Intelligent Transportation Systems.
- In Chapter 5, we further discuss the details of the method and implementation. It also contains some extra experiments and results.
- In Chapter 6, we further investigate and discuss the results.
- Finally, in the last chapter we summarize the study, discuss the limitations of the work and possible suggestions for future studies.

CHAPTER 2 LITERATURE REVIEW

This chapter is dedicated to reviewing the previous related research. At the beginning, an overview about traffic data collection is presented. Video-based object detection as a major source of traffic data collection is then explained with its applications, such as traffic event monitoring, parking monitoring and law enforcement. Then we will review how different pattern recognition approaches have been used to extract knowledge from traffic trajectories.

2.1 Video Surveillance

Whether the data collection be through fixed or moving sensors, e.g. on vehicles or pedestrians, such as pole or drone mounted regular or infrared cameras, radars, LiDARs and GNSS devices [4–7], various kinds of sensors provide user trajectories, i.e. the series of their positions over time. Video based traffic studies is especially emerging fast due to increasing number of cameras, growing coverage of the road network [8] not requiring active participation of road users.

Video-based object detection and pattern recognition has been deployed for different traffic problems in urban environments such as vehicle counting, speed and movement detection (e.g. for traffic violation detection), road user interaction analysis and automatic incident detection. In several cases, a reliable approach is through the recognition of license plates, i.e. Automatic Number Plate Recognition (ANPR) or Automatic License Plate Recognition (ALPR) [8].

Vehicle detection is a well-known application of computer vision that is deployed in different environments with vision challenges and occlusions [9–11]. Comparing the performances in highways vs. rural roads, it seems that detection and classification in urban environment is more challenging compared to highways [8].

Traffic event monitoring is studied in several researches [12–14]. Li and Porikli [12] developed an algorithm to detect several types of traffic events such as heavy congestion, high vehicle density with high speed and vacancy in different weather situations. Video-based automatic incident detection is also used to monitor the traffic in locations where temporary road lane changes for increase in road capacity is practiced [15, 16].

Parking lot monitoring and management is another application of video surveillance related to traffic management. Unauthorized parking detection and vehicle identification (in case of blockages) are among the issues that are addressed in some researches [15, 17]. Law

enforcement is another area where video-based traffic monitoring can be useful. Traffic violations such as illegal movements, speeding, running red lights, vehicle driving in the wrong direction or in a bus lane can be automatically detected in real time [8, 15, 18]. The output of video records in traffic surveys are usually stored as trajectories which are essentially time-series that represent the positions of objects over time.

2.2 Pattern Recognition

As the name suggests, pattern recognition is the task of learning patterns and regularities within data through algorithms [19]. Statistical modelling, machine learning and deep learning are some of the popular terms in the nomenclature of pattern recognition. They refer to different types of learning methods with different algorithms (explained in the following sub-section), yet with the same objective, which is reaching high accuracy by minimizing some loss function(s) [20].

2.2.1 Learning Types

Statistical methods are traditional learning methods that are based on statistics and are well studied theoretically regarding the optimality of the results in different situations. On the other hand, machine learning algorithms rely less on statistical assumptions (often non-parametric methods) and have relatively more emphasis on prediction performance as the main concerns, even if the optimality of the results is not mathematically proven.

Most statistical and machine learning methods consist of two main steps: a feature engineering step to extract features from raw data to be used as the input for the methods, and a training part to fit a model to those features using the algorithms.

2.2.2 Supervision Types

A dataset consists of the recorded values for several variables and may or may not include the response variable. If present, the response variable is sought to be predicted based on the other variables. Based on the presence/use of response variable, also called label or target, learning methods are divided into supervised, unsupervised and semi-supervised algorithms. If labels are available and used by a method, it is considered supervised while unsupervised methods do not use any label. Semi-supervised methods use a combination of labeled and unlabeled data. The labels can be either continuous (numeric) or discrete (nominal) variables.

Clustering is a category of unsupervised learning methods which are often used in the absence

of prior knowledge on the labels, but also sometimes used while the labels are known in the dataset as a pre-processing step to give insight about data for further analysis [21]. Given a similarity measure, clusters are made by putting similar objects/observations in the same groups or clusters.

2.2.3 Time-series Analysis

Time-series data consists of single or multiple sequence(s) of time-ordered records. The learning task can be either at the shape level or structure level. The former is when performing on multiple sequences to study the similarity/dis-similarity among them, while the latter tries to find similar sub-sequences within a single or multiple sequence(s) [22].

There are three types of learning depending on the input type used [21, 22]:

- **Temporal-proximity based, shape based or raw-data based:** when the learning method takes raw time-series as input with no need for pre-processing steps on data, except for those steps needed to extract the time-series themselves from the data.
- **Representation based or feature based:** when the data should be processed to extract features from trajectories and those features are the input of the learning method.
- **Model based:** these methods assume that the data comes from models/distributions and tries to split the data into groups of observations where each group is assigned to one of the models.

A crucial part of clustering problems is selecting a suitable measure of similarity. Among the similarity measures used in time-series analysis are lock-step measures [23, 24] such as Dissimilarity metric (DISSIM), Bhattacharyya distance, Euclidean Distance (ED) and more generally the L_p norms that are based on comparing fixed (one-to-one) pairs of elements from input time series of the same length [24]. On the other hand, elastic measures such as Dynamic Time Warping (DTW), distances based on Longest Common Sub-Sequence (LCSS), Edit distance with Real Penalty (ERP), Edit Distance on Real sequence (EDR) and Sequence Weighted Alignment model (SWAle) can handle time-series of different lengths. However, this flexibility in input length usually comes at the cost of higher computational burden. For instance, the computational burden for the ED and Bhattacharyya distances is $O(n)$ while computing several elastic measures including DTW and LCSS are of $O(mn)$ where m and n are the lengths of the time-series [25].

Studies including time series data analysis can be categorized in two different ways: based on the characteristics of the data or of the method [26]. Regarding the dataset, it is possible

to train models on raw time-series. But this is not practical to extract complex relations from large real-world datasets because of the need for more complex models and thus higher computational cost. Also, in some cases, the algorithms require the input series to have the same length. Therefore, data pre-processing is usually needed before training the model.

A popular type of time-series data pre-processing is transforming the data. Simple curves can be fitted to complete or subsets of time-series to represent the sequences. Discrete Fourier Transformation is another method to extract features from time-series. The data can also be mapped into lower-dimension subspace by orthogonal transformations [25].

According to the analysis approach, the three main characteristics of time series analysis are distance measure, classification/clustering algorithm used and evaluation method. Many algorithms used to compare time series are generally the same as those for stationary (non-time series) data, only with the information being summarized in a set small number of features that are inputs of the algorithms [26].

2.3 Trajectory Clustering

Various problems, such as road user counting, speed measurement and traffic violation detection are solved/eased using trajectory-based road user detection and clustering [8, 27–29].

In the following sections, popular methods used in the literature for the computation of the distances between trajectories, clustering and evaluation of clustering performance are explained. As explained later in the article in Chapter 4.3, we define a clustering setup ω that consists of a distance d, distance parameters α, a clustering algorithm A, algorithm parameters β and the number of clusters k. Also D denotes the set of n_D trajectories M.

2.3.1 Distance Measures

As mentioned earlier, distance measurement is the first step of trajectory clustering. Although none-elastic distances such as ED, Bhattacharyya and DISSIM which use one-to-one mapping technique for distance measurement are more time-efficient, they are not suitable for trajectories with unequal lengths [21]. But real trajectories are usually different in duration and length. Therefore, only elastic distance methods are considered in this study as they can handle trajectory pairs with unequal lengths.

Some of the most popular distance measurement methods in the trajectory clustering literature are as follows [21, 25, 30]:

- **Dynamic Time Warping (DTW)** First introduced by Sakoe at early 70s for speech

recognition [31], this method is among the most popular techniques for calculating the dissimilarity of trajectories.

Given the trajectories $M^1 = (p_1^1, ..., p_m^1$ and $M^2 = (p_1^2, ..., p_n^2$ and starting from the start points of both trajectories p_1^1 and p_1^2, it maps the points from M^1 to the points from M_2 while preserving the temporal order and minimizing the distance of the matched pairs of points. After finding the optimal mapping, the distance is the sum of distances between all pairs. It can be formulated as [32]:

$$DTW(M^1, M^2) = \begin{cases} 0 & \text{if } |M^1| = |M^2| = 0 \\ \infty & \text{if } |M^1| = 0 \text{ } or \text{ } |M^2| = 0 \\ d(p_i^1, p_j^2) & \\ \quad +min\Big(DTW\Big(H(M^1), M^2\Big), & \\ \quad DTW\Big(M^1, H(M^2)\Big), & \\ \quad DTW\Big(H(M^1), H(M^2)\Big)\Big) & \text{otherwise} \end{cases}$$

Where $H(M)$ is the trajectory M with its last point removed. The complexity for the pair of trajectories M^1 and M^2 is of $O(mn)$. It is also worth mentioning that the mappings $M^1 \rightarrow M^2$ and $M^2 \rightarrow M^1$ are similar and the distance is thus symmetrical.

- **Longest Common Sub-Sequence (LCSS)** is also among the most popular measures in trajectory analysis. When mapping trajectories $M^1 \rightarrow M^2$ it allows points from either trajectories to remain unmatched. The measure is calculated as below [33]:

$$LCSS_\epsilon(M^1, M^2) = \begin{cases} 0 & \text{if } m = 0 \text{ } or \text{ } n = 0 \\ 1 + LCSS_\epsilon\Big(H(M^1), H(M^2)\Big) & \text{if } |p_m - p_n| \leq \epsilon \\ \max\Big(LCSS_\epsilon\Big(H(M^1), M^2\Big), LCSS_\epsilon\Big(M^1, H(M^2)\Big)\Big) & \text{otherwise} \end{cases}$$

Where ϵ is the matching threshold, that is, the maximum distance between points to be considered as similar. The LCSS returns the length of the longest uninterrupted sub-sequence of the two trajectories where the points are close enough (regarding ϵ) to the other trajectory. The choice of ϵ depends on the characteristics of the road and road users (e.g. road/lane width, vehicle width and driving behavior).

Also as suggested by Vlachos et al. [34] to turn this similarity measure into a distance (to be used with the algorithms), we convert the measure as in Equation 2.1:

$$d_{LCSS_\epsilon}(M^1, M^2) = \frac{1 - LCSS\epsilon(M^1, M^2)}{min(m, n)} \tag{2.1}$$

Where $LCSS(M^1, M^2)$ is the LCSS similarity between trajectories M^1 and M^2, i.e. the lengths of the longest common subsequence between them. The LCSS similarities and distances are also symmetric. Finally, the complexity of this method for calculating the similarity for a pair of trajectories is $O(mn)$.

- **Edit Distance on Real sequence (EDR)** is also widely used in the literature. This measure represents the minimum number of points needed to be changed so that the two trajectories would be similar. The formula to calculate the measure is as below:

$$EDR_\epsilon(M^1, M^2) = \begin{cases} n & \text{if } m = 0 \\ m & \text{if } n = 0 \\ min\Big(EDR_\epsilon\Big(R(M^1), R(M^2)\Big) + subcost, & \\ \quad EDR_\epsilon\Big(R(M^1), M^2\Big) + 1, & \\ \quad EDR_\epsilon\Big(M^1, R(M^2)\Big) + 1\Big) & \text{otherwise} \end{cases}$$

Where ϵ is the matching threshold, $R(M)$ is the trajectory M with the start point being removed and $subcost = 0$ if the start points of the two trajectories are close enough (i.e. $|p_1^1, p_1^2| \leq \epsilon$), otherwise it is 1. The EDR is also a symmetric distance and its calculation complexity is $O(mn)$. Also the trajectories do not need to be of equal lengths, though unequal lengths of the two trajectories automatically inflates the measure [33].

- **Piciarelli-Foresti(PF)**: starting from start points p_1^1 and p_1^2, the method tries to match every point p_i^1 from the first trajectory M^1 to a point p_j^2 within a time window around a similar point (in order) from M^2 (i.e. $p_i^2 \in M^2$) where the time window grows with time. The distance is calculated as the average distance of the paired points:

$$PF(M^1, M^2) = \frac{1}{m} \sum_{i=1}^{m} d_{PF}(p_i^1, M^2) \tag{2.2}$$

Where:

$$d_{PF}(p_i^1, M^2) = \min_\tau \left(\frac{|p_i^1, p_\tau^2|}{c_\tau} \right), \quad \tau \in \{\lfloor (1-\delta)i \rfloor \dots \lceil (1+\delta)i \rceil\} \tag{2.3}$$

In an study to compare distance and clustering methods for trajectory cclustering, Morris and Trivedi [35] suggest to set the normalization constant $c_\tau = 1$.

- Informally, the **Hausdorff** distance represents the maximum of all the distances from a point in one set to the closest point in the other set. It is formally defined as [30]:

$$d_{Hausdorff}(M^1, M^2) = \max\left\{ \max_{i\in[1,\dots,|M^1|],j\in[1,\dots,|M^2-1|]} d_{ps}(p_i^1, s_j^2), \max_{i\in[1,\dots,|M^1-1|],j\in[1,\dots,|M^2|]} d_{ps}(p_j^1, s_i^2) \right\} \tag{2.4}$$

Where segment s_j^a is the straight line between two successive points $p_j^a, p_{j+1}^a \in M^a$, and point-to-segment distance $d_{ps}(p_i^1, s_j^2)$ is defined as:

$$d_{ps}(p_i^1, s_j^2) = \begin{cases} |p_i^1, p^{*1}_i| & \text{if } p^{*1}_i \in s_j^2 \\ \min\left(|p_i^1, q_j^2|, |p_i^1, q_{j+1}^2|\right) & \text{otherwise} \end{cases}$$

Where p^{*1}_i is the orthogonal projection of p_i^1 on the line including segment s_j^2.

The computation complexity of the Hausdorff distance is $O(n^2)$ [36].

- **Symmetric Segment-Path Distance (SSPD)** has been proposed in [30, 36] as:

$$d_{SPD}(M^1, M^2) = \frac{1}{|M^1|}\sum_{i=1}^{|M^1|} d_{pt}(p_i^1, M^2) \tag{2.5}$$

$$d_{SSPD}(M^1, M^2) = \frac{d_{SPD}(M^1, M^2) + d_{SPD}(M^2, M^1)}{2} \tag{2.6}$$

Where $d_{ps}(p_i^1, s_j^2)$ is point-to-segment distance (defined in Equation 2.3.1), and point-to-trajectory distance $d_{pt}(p_i^1, M^2)$ is deined as:

$$d_{pt}(p_i^1, M^2) = \min_{j\in[o,\dots,|M^2|]} d_{ps}(p_i^1, s_j^2) \tag{2.7}$$

The computation complexity of the SSPD is $O(n^2)$ [36].

- **Fréchet distance** Originally introduced by Fréchet [37], this measure is used in the literature for trajectory analysis and was initially included in this work. But it can be very time consuming, having a worst case complexity of $O(n^2 \log(n^2))$ [38] . This was reflected in the long run-times (compared to other distances) when calculating the

distance matrix based on this measure. Also sometimes the available implementation had crashes. Therefore, we decided to exclude it from the experiments.

2.3.2 Clustering Algorithms

After calculating distances between trajectories, the next step is to use a clustering method to split the set of trajectories into groups of similar objects based on distance values.

Some of clustering methods such as k-Means and Gaussian Mixture Model (GMM) rely on calculation of centroid for sets of objects which cannot be defined for objects of time-series type. Therefore, these methods are not applicable unless trajectories are mapped into a proper space where each trajectory is represented with a single point [39].

The following are some popular clustering methods which do not rely on calculation of centroid points [21, 25]:

- **k-Medoids** is very similar to k-Means but tries to find k points from the dataset (rather than the feasible space) to represent the center of each cluster. Given a set of initial cluster centers, in every step it updates each cluster center with a point from the corresponding cluster while trying to minimize the sum of the distances of each observation to cluster centers. The complexity of the method is of $O(n_D k)$ [40].

- the **Hierarchical clustering** approach generates a dendogram which splits the observations into clusters while showing "the nested grouping of patterns" [41]. The approach consists either in splitting a set into subsets in each step (divisive), or in merging sets of observations into a superset (agglomerative), or in a mix of the two (hybrid).

  According to the study by Morris and Trivedi [35], these algorithms have similar performances in trajectory clustering. Beside the number of clusters k, there is another parameter called *linkage* for this algorithm that defines the criterion used to merge the subsets. There are four values for this parameter:

  1. **Single linkage** is based on the minimum distance between objects from the two sets.
  2. **Complete or maximum linkage** is based on the maximum distance between objects from the two sets.
  3. **Average linkage** takes the average of distances over all objects in the two sets.
  4. **Ward linkage** is based on variance of objects in the two sets. This method was not used in the final results presented in this paper due to very poor results.

- the **Spectral clustering** algorithm takes a distance matrix T, decomposes it to its eigenvalues and eigenvectors and picks the eigenvectors corresponding the k smallest eigenvalues. These k eigenvectors are used as a low-dimensional embeddings of the dataset to apply a regular clustering method (such as k-Means) [42, 43].
- **Density-based spatial clustering of applications with noise (DBSCAN)** finds points associated with high density (core points) and expands the clusters by connecting the core points that are close enough to each other. Its parameters are the zone radius r_z (for density check around each point) and minimum number of points around a point to be considered a core point n_z^{min}.
- **Ordering Points To Identify The Clustering Structure (OPTICS)** is another density based method similar to DBSCAN which relaxes the requirement of setting the parameters n_z and r_z to deal with varying density environment. Giving boundaries for these two parameters is optional and reduces the complexity of the calculations.

2.3.3 Unsupervised Performance Measures

To evaluate clustering performance in the absence of ground truth, unsupervised performance measures (also known as internal indices) are used. Some of the most popular unsupervised performance measures are Silhouette (S) [44], Calinski-Harabasz Index (CHI) [45] and Davies-Bouldin Index (DBI) [46]. These measures all use the distance d_α to check the performance of the clustering setup ω.

- **Silhouette** is a very popular internal performance metric which does not need ground truth for evaluation. It is defined for each object i as [44]:

$$s_i = \frac{b_i - a_i}{max(a_i, b_i)} \tag{2.8}$$

  Where a_i is the mean distance between the object i and all other objects within the same cluster, and b_i is the mean distance between the object i and all the objects within the closest cluster to i. The total clustering performance based on Silhouette S is then calculated as the average silhouette coefficient over all objects.

- the **Calinski-Harabasz Index (CHI)** is based on the dispersion of objects within and between clusters. For the dataset D of size n_D we have [45]:

$$CHI = \frac{tr(B)}{tr(W)} \times \frac{n_D - k}{k - 1} \tag{2.9}$$

Where $tr(B_k)$ and $tr(W_k)$ are traces of between cluster and within-cluster dispersion matrices, respectively, and are calculated as below:

$$W_K = \sum_{k=1}^{K} \sum_{x \in C_q} (x - c_q)(x - c_q)^T \tag{2.10}$$

$$B_k = \sum_{k=1}^{K} n_q (c_q - c_D)(c_q - c_D)^T \tag{2.11}$$

Where n_q is the number of objects in cluster q, and c_q and c_D are the centers of cluster q and dataset D, respectively.

- the **Davies-Bouldin Index (DBI)** was proposed in [46].It is formally defined as:

$$DBI = \frac{1}{k} \sum_{q=1}^{k} \max_{q \neq q'} R_{q,q'} \tag{2.12}$$

Where:

$$R_{q,q'} = \frac{d_q + d_{q'}}{d_{q,q'}} \tag{2.13}$$

d_q is the average distance of objects in cluster from its centroid, and $d_{q,q'}$ is the distance between centroids of the clusters q and q'.

These performance measures are unsupervised and are used widely in the literature. They are mostly suitable for comparison between clustering setups with same algorithms since they make assumptions about cluster structure [21].

However, unsupervised performance measures cannot reflect results expected for specific applications. If an expected, even partial, reference clustering is known, generating a result that matches this clustering may not yield the best unsupervised performance measure.

Also in the case of CHI and DBI, these two measures require definition of centroid for clusters, which is not naturally defined for a set of trajectories. Therefore, only Silhouette is applicable for trajectory clustering purposes.

2.3.4 Supervised Performance Measures

When reference labels are available, supervised performance measures can be used to evaluate clustering performance. In the following, some of popular supervised performance measures

in the literature are explained:

- **Completeness, homogeneity and V**: according to [47], for the reference labels C and a cluster assignment K, we define entropy of C and K as:

$$H(C) = -\sum_{c=1}^{|C|} \frac{n_c}{n}.\log\frac{n_c}{n} \tag{2.14}$$

$$H(K) = -\sum_{k=1}^{|K|} \frac{n_k}{n}.\log\frac{n_k}{n} \tag{2.15}$$

Also we have conditional entropies of C given K $H(C|K)$ and K given C $H(K|C)$ as:

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|} \frac{n_{c,k}}{n}.\log\frac{n_{c,k}}{n_k} \tag{2.16}$$

$$H(K|C) = -\sum_{k=1}^{|K|}\sum_{C=1}^{|C|} \frac{n_{c,k}}{n}.\log\frac{n_{c,k}}{n_c} \tag{2.17}$$

Then we define performance metrics completeness (c) and homogeneity (h):

$$c = 1 - \frac{H(K|C)}{H(K)} h = 1 - \frac{H(C|K)}{H(C)} \tag{2.18}$$

Finally, based on these two metrics, we can calculate V measure:

$$v = \frac{(1+\beta)\times h \times c}{\beta \times h + c} \tag{2.19}$$

We use $\beta = 1$ to give equal weights to completeness and homogeneity indices.

- **Adjusted Mutual Information (AMI)**: with the same definition of entropy as above, defining $P(i) = \frac{C_i}{n}$ of a random picked object feom dataset would belong to reference set C_i and defining $P(i,j) = \frac{|C_i \cap K_i|}{n}$, we have Mutual Information (MI) [48]:

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} P(i,j)\log\frac{P(i,j)}{P(i)P'(j)} \tag{2.20}$$

Since MI index is more in favor of clustering with higher number of clusters k, we use Adjusted Mutual Information (AMI) instead:

$$AMI = \frac{MI - E[MI]}{max(H(U), H(V)) - E[MI]} \tag{2.21}$$

Where *E[MI]* is the expected value of MI index.

- **Adjusted Rand Index (ARI)**: Rand Index (RI) is a simple, yet popular index used as an external clustering performance. It is defined as [49]:

$$RI = \frac{a + b}{C_2^n} \tag{2.22}$$

Where a is the number of object-pairs that are in same group in both C and K, b is the number of object-pairs that are in different groups in both C and K, and $C_2^n = \frac{n(n-1)}{2}$ is the total number of object-pairs in the dataset. For a random label assignment, MI is not necessarily zero. Instead, we use Adjusted Rand Index which fixes this using expected value of RI:

$$ARI = \frac{RI - E[RI]}{max(RI) - E[RI]} \tag{2.23}$$

- **Fowlkes-Mallows Index (FMI)**: is also based on the classification of the object pairs, considering two objects being in the same clusters in both assignments as the positive class and two objects being different ones as the negative class. For such a binary classification problem, the numbers of True Positives (TP), False Positives (FP) and False Negatives (FN) can be counted and the Fowlkes-Mallows Index (FMI) is calculated as the geometric mean of precision and recall [50]:

$$FMI = \frac{TP}{\sqrt{(TP + FP)(TP + FN)}} \tag{2.24}$$

2.3.5 Previous Trajectory Clustering Studies

Choong et al. used K-means and Fuzzy C-Means (FCM) for trajectory clustering for the traffic surveillance at intersections [51]. They used LCSS as their similarity measure and then the Gaussian kernel function to convert LCSS similarity into distance. Finally, they compared the results against ground-truth through the Rand Index (RI).

In [52], the researchers used the average pairwise distance of the first N points of trajectories to represent the dissimilarity of each pair of trajectories, where N is the length of the shorter

trajectory. They also used two layer clustering based on FCM, Spectral and hierarchical clustering algorithms and compared them against each other.

Zhao et al. [53] used Douglas-Peucker (DP) based compression for re-sampling marine trajectory data, together with the DTW for calculating the distances between the trajectories. Finally they used a Density-Based Clustering of Application with Noise (DBSCAN) algorithm for clustering and evaluated them with a custom performance metric based on ground-truth labels.

To address scalability of trajectory distance calculations and lower the run-time, Kumar et al. proposed a method based on DTW distance which is suitable for large datasets of overlapping trajectories in dense road networks. They also proposed a four step Visual Assessment of (cluster) Tendency (VAT) based clustering method. This method tries to order the distance matrix based on distance of the objects and provides visual assistance based on an image representing the oredered distance matrix. This visual assisstance is used as an assessment to find suitable number of clusters. Finally they compared their proposed methods with DBSCAN, OPTICS, NETSCAN and NEAT using two unsupervised performance metrics, namely Silhouette and Dunn measures.

The automatically counting vehicle movements at intersections was considered in [54]. Their approach consists of vehicle trajectory detection/tracking and then trajectory clustering. The LCSS was employed as a similarity measure in a customized clustering algorithm similar to K-means. Finally, they used traffic turning count as ground truth to calculate the Weighted Average Percentage Error (WAPE) , for the optimization and evaluation purposes.

In a recent work, Wang et al. first extracted a representation of the network [39]. They used it to learn the embedding vectors of the vehicles. They used Gaussian Mixture Models (GMM), K-means and K-medoids clustering algorithms. To investigate the performance of these clustering algorithms with different hyper-parameter values, they used three common internal performance indices, namely the Silhouette coefficient, the Davies-Bouldin Index (DBI) and the Calinski-Harabasz Index (CHI).

Xue et al. [55] proposed a method for source-destination clustering to produce candidate ODs paths. Then they used these trajecory paths for a classification task using Longest Common Sub-Sequences (LSTM) based methods.

Even though different methods of similarity measurement and clustering algorithms are applied in trajectory analysis and novel methods have been proposed in the literature, unsupervised performance measurement is mostly overlooked.

As mentioned, unsupervised performance measures use the same distance d_α used in the

clustering procedure to check the performance of the clustering setup ω. This can be an issue especially when comparing distance measurement methods.

On the other hand, a large body of literature uses supervised performance measures with manually labeled data. However, given the heuristic nature of clustering approaches and the concern about their sensitivity to different data characteristics, errors in data and also in judgement [56], it may be necessary to systematically evaluate the clustering performance on each dataset. Manual data labeling would cost extra time and effort for such evaluation.

In this study, the focus is on providing a method for using a combination of both unsupervised and supervised performance measures without needing manually labeled data. The proposed method automatically generates trajectory reference clusters based on origin and destination points of trajectories to be used by supervised performance measures. Therefore, the entire procedure remains unsupervised both in the clustering and evaluation steps.

CHAPTER 3 RESEARCH PROJECT APPROACH

In this chapter, the outline of the work is discussed. First, a brief description of the components of the framework is provided and the choices of programming software is discussed. Then, the distance methods and clustering algorithms used in the comparison are explained. In the end, the characteristics of the datasets used for comparison of the methods are detailed.

3.1 Framework

To achieve the objectives of the research project that were set out in section 1.2, a framework is developed and coded in a programming language. Figure 3.1 shows the overview of the framework.

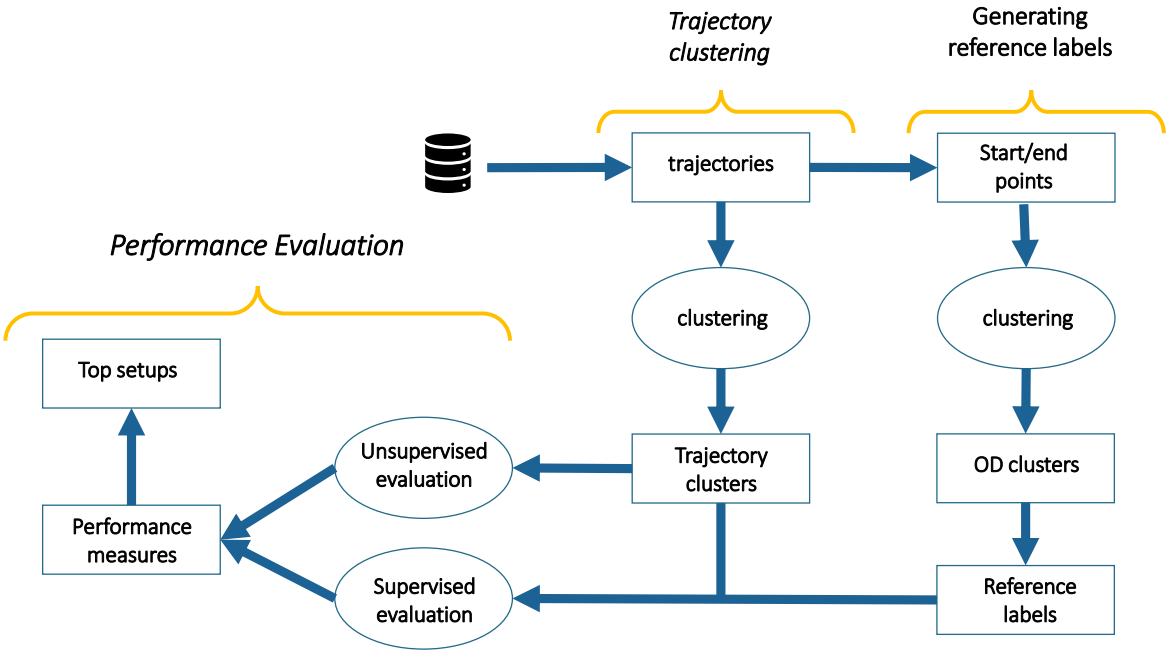


Figure 3.1 The overview of the framework

The framework is explained in detail in Section 4.3, Algorithms 1 and 2. It consists of three main steps, namely, automatic reference label generation, trajectory clustering and performance evaluation.

The distance measures with their parameter values used in this study for trajectory clustering are presented in Table 4.1. As mentioned earlier, the time complexity of Fréchet distance is of

$O(n^2 \log(n^2))$. This was reflected in the long run-times (compared to other distances) in early tests when calculating the distance matrix. Also sometimes the available implementation had crashes. Therefore, it was excluded from the final comparison experiment.

Table 4.2 shows clustering algorithms used for trajectory clustering and parameter values. Since the two parameters of DBSCAN are highly sensitive to the choice of the distance measure, its range and the distribution of the distances in the dataset, different parameter values are picked manually by trial and error for every distance measure, illustrated in Table 3.1.

Distance measure	n_z	r_z
DTW	8	150
LCSS	6	0.41
PF	5	205
EDR	2	0.02
Hausdorff	10	4.7
SSPD	5	2

Table 3.1 Parameter values of DBSCAN for different distance measures

For the Spectral clustering method, the heat kernel is used to convert distances into similarity values [57]:

$$s = e^{-\beta \frac{d}{\sigma}} \tag{3.1}$$

Where σ is the standard deviation of the distance values and $\beta = 1$.

The performance measures used in this study are presented in Table 4.3.

Given the diverse tasks in the framework and to make it easy and convenient to use by the public, *Python* was used as the main programming platform: it is an open-source, general-purpose programming language which is frequently used in scientific researches as scripting language [58, 59].

Out of different options for data management (discussed in section 5.1), Structured Query Language (SQL) was used to keep the run times low while avoiding overload on Random Access Memory (RAM). Although there are different free and open source SQL softwares with built-in functions to support spatial and geographical calculations, SQLite was chosen. SQLite is an embedded database software available for many programming languages. It comes bundled with Python as a module, therefore no further software installation or requirement is needed here. This adds to the convenience of use of the code developed here for the end user.

3.2 Data

To investigate the sensitivity of distance measures and clustering algorithms to different data specifications (e.g data collection method, road network type and regulations, road user behaviors, data noise and quality), the comparisons are conducted on data from different sites. We use seven sets of trajectories from two different datasets, with different characteristics. To keep the run-times reasonable (less than a day per intersection), 500 trajectories are sampled from the set of trajectories of each dataset for for clustering.

3.2.1 NGSIM Dataset

The first dataset used in this work is originally from a dataset made publicly available by the Next Generation SIMulation (NGSIM) program [60]. The program was initiated by the United States Department of Transportation (US DOT) Federal Highway Administration (FHWA) in the early 2000's. It is a popular dataset used among traffic researchers as a reference for benchmarking their empirical studies [61].

The data was collected through a network of synchronized digital video cameras. A customized software application called NGVIDEO was developed for this program. NGVIDEO extracted the vehicle trajectory data from video. The output trajectory data contains location of each vehicle within the study area at time every one-tenth of a second [62].

One dataset from the NGSIM was collected on a segment of Peachtree Street, in downtown Atlanta, Georgia (shown in Figure 3.2) on November 8 and 9, 2006. The street is an arterial with primarily north-south direction and the speed limit of 35 miles/hr. The study considers the intersections of this arterial road with 5 streets namely, 10th, 11th, 12th, 13th and 14th streets.

The videos were recorded using 8 cameras being mounted on a 30 floors building. Figure 3.3a shows the segments of the arterial with the camera coverage, as well as the location of cameras. Figure 3.3b represents a schematic of the study area with unique identification numbers for intersections and lanes. Overall these segments cover almost 2,100 feet long, and two or three arterial through lanes at each intersection.

There are five intersections with intersection 1 as the southernmost, where Peachtree and 10th street meet and intersection 5 at the northernmost section of the study area, where 14th cuts Peachtree. Intersections 1, 2, 3, and 5 are signalized while intersection 4 is unsignalized and is controlled by stop signs. In this study we only considered signalized intersections. Figure 3.3b presents the street lanes with incremental numbers being started at the closest lane to the median, with the exception of left-turn bays along Peachtree street, which are

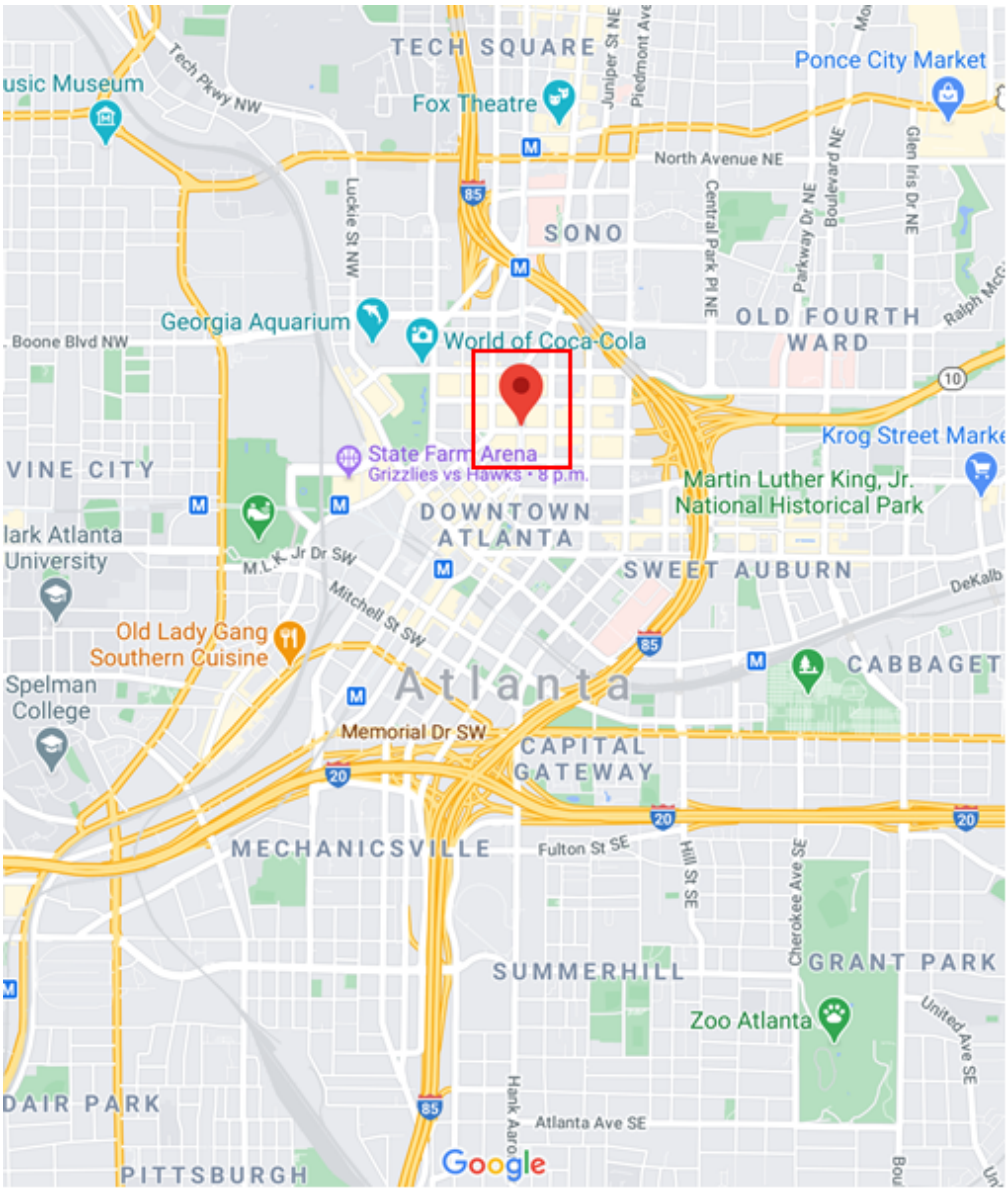
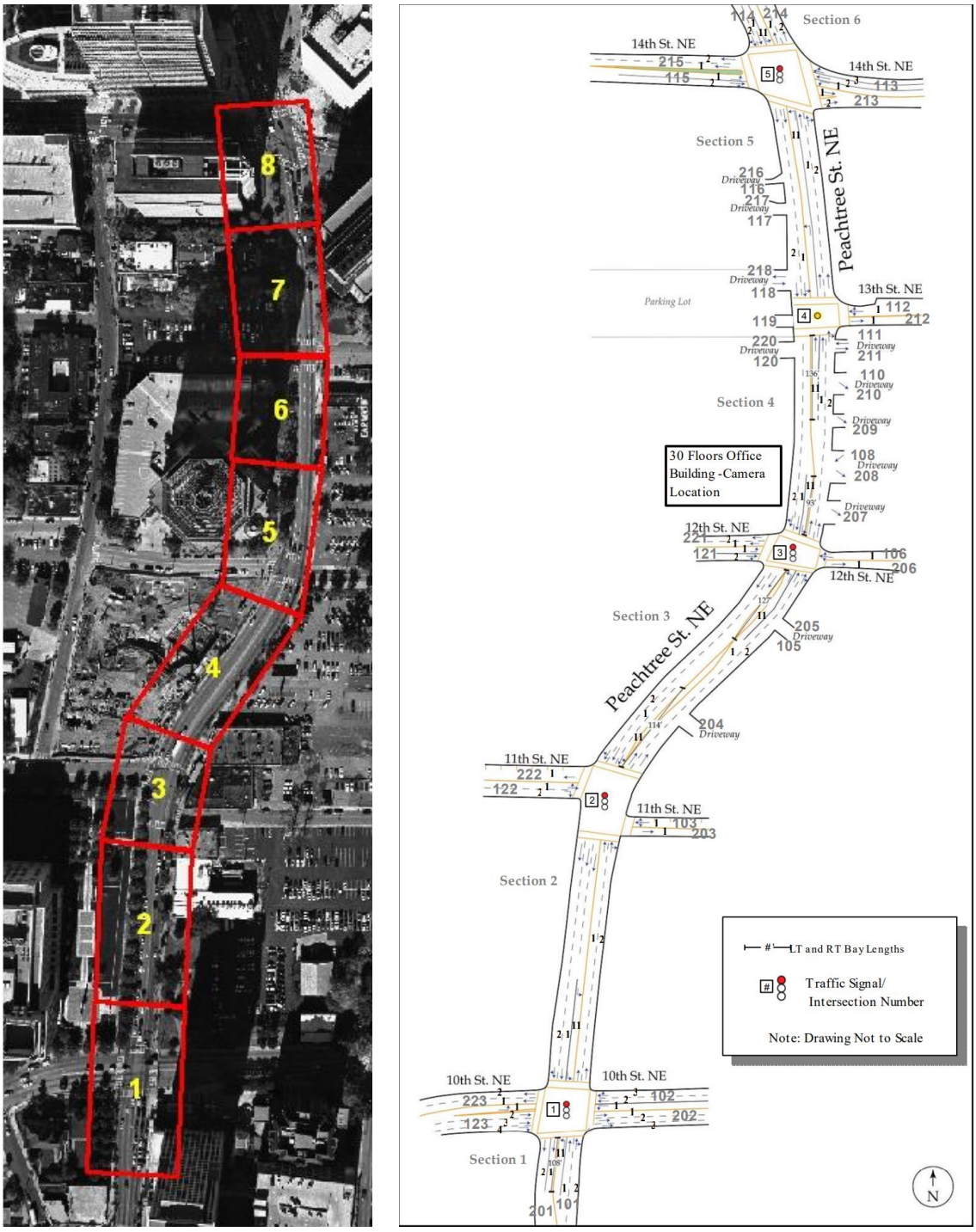


Figure 3.2 Locations of data collection sites for NGSIM data in Atlanta, Georgia

numbered 11.

3.2.2 inD Dataset

The second set of trajectories is from the intersection Drone (inD) project. This project used the drone technology to collect and create a trajectory dataset in Aachen, Germany from 2017 to 2019 [63]. The data collections sites are from different locations in the city to cover different traffic rules and densities as well as road layout (shown in Figure 3.4).



(a) Aerial photo of NGSIM sites used in this project

(b) Site map of NGSIM data used in this project

Figure 3.3 NGSIM data collection sites

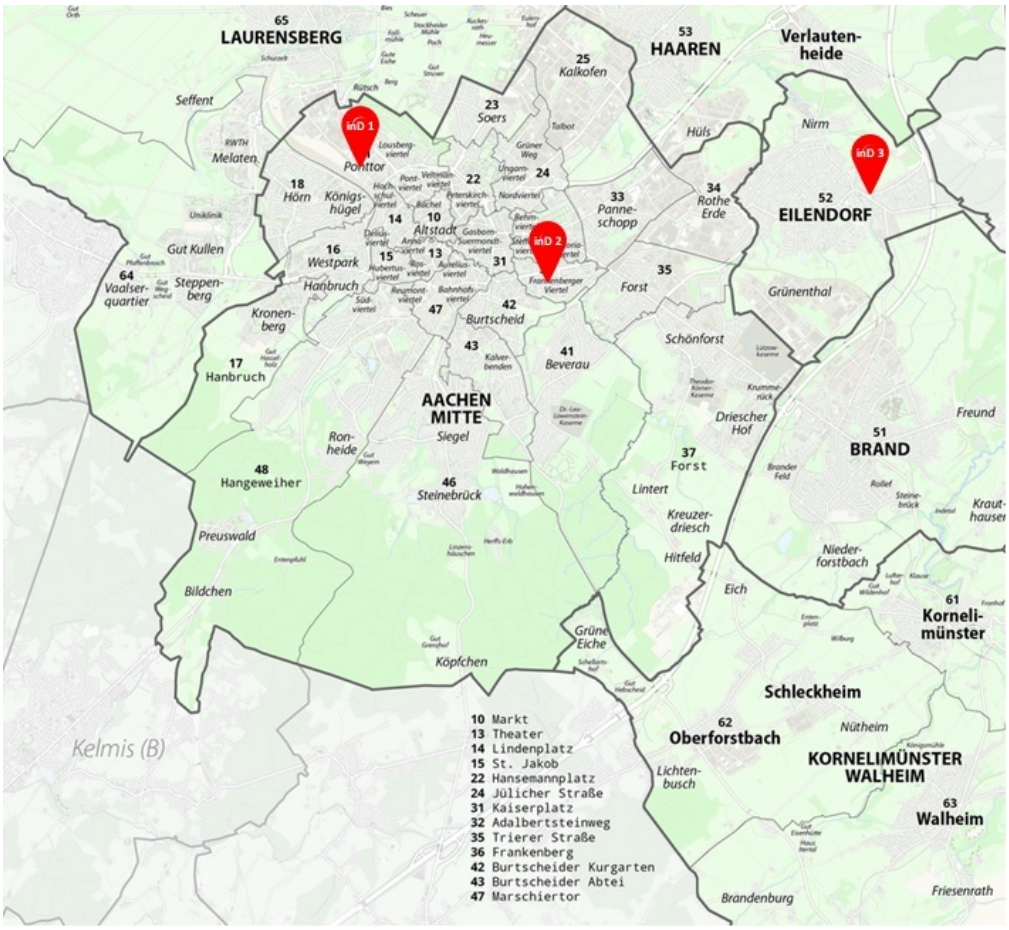


Figure 3.4 Locations of data collection sites for NGSIM data in Aachen, Germany

It contains tracked trajectories with a high level of accuracy and precision (positioning error less than 0.1 meters). For video recording the drones were flying to a predefined position at an altitude up to 100 meters above the ground. Each video sequence included 20-22 minutes video recording with DJI Phantom 4 Pro at 25 fps in 4K (4096 × 2160 pixel) resolution.

To produce the trajectories, first a computer vision algorithm based on deep neural networks was used to detect and classify the positions and types of the road users in each frame. Then the detections between consecutive frames were compared and matched by their distances and a Bayesian smoothing approach and a constant acceleration method were employed to refine trajectories based on information related to all detections of the corresponding track.

Generally, the dataset represents trajectories of 13,599 road users including 5366 Vulnerable Road Users (VRUs) (e.g. pedestrians and bicyclists) as well as 8233 cars, trucks and busses The trajectory sets from inD data used in this study are collected from three intersections:

- Bendplatz intersection (**inD 1**) is a four-way intersection near a university and railway

depot, where Kühlwetterstraße and Süsterfeldstraße meet. The intersection is unsignalized, but Süsterfeldstraße (North-West/South-East) is the major road. It has one left turn lane on each side, however the pedestrian crossings are not regulated. The intersection has a high number of cyclists, buses and pedestrians, due to the closeness to the university. Therefore, it seems there are interactions between pedestrians and cyclists and the turning vehicles at this intersection. Figure 3.5 shows an aerial photo of the intersection.

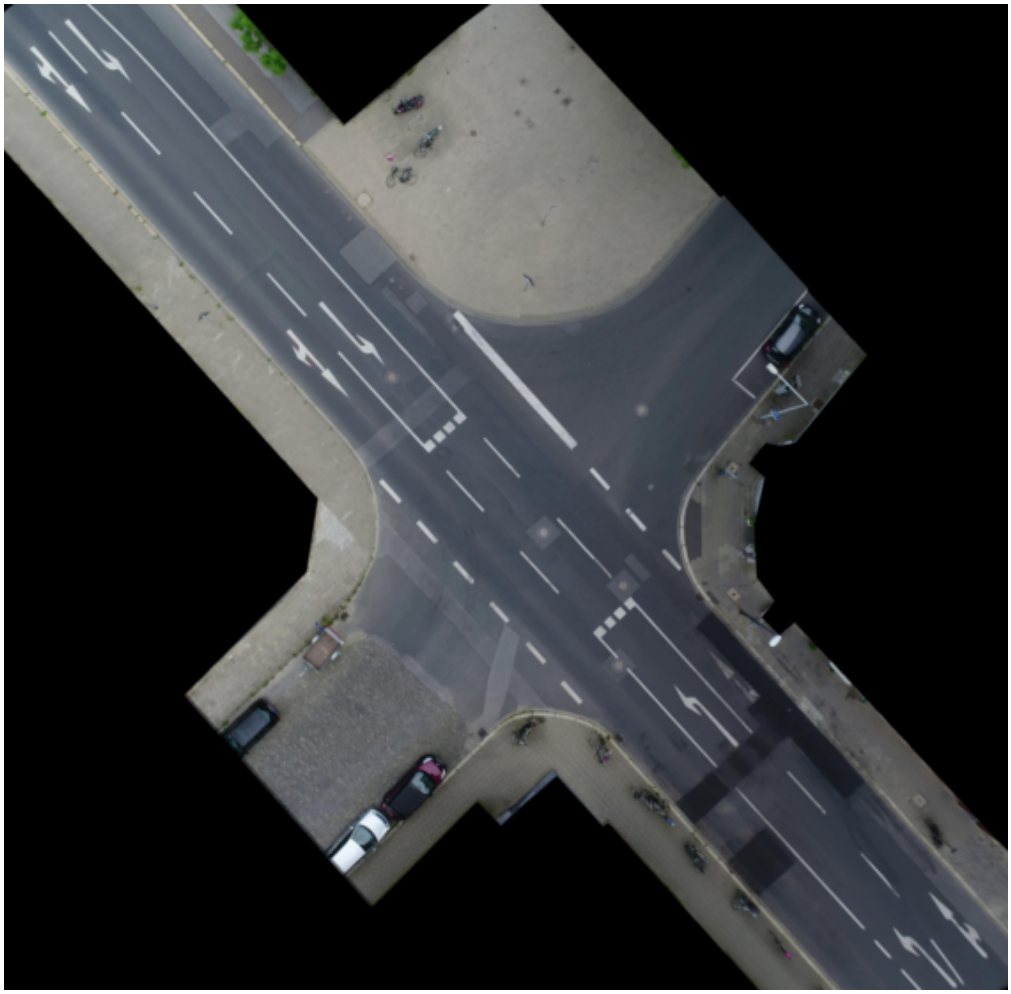


Figure 3.5 Aerial photo of Bendplatz intersection (inD 1), Aachen, Germany

- Frankenburg (**inD 2**) is a four-way intersection near the city centre (shown in Figure 3.6). It is the meeting point of Bismarckstraße and Schlossstraße. There is a park on the south-east corner of the intersection. The Intersection is not symmetric as the Schlossstraße is not perfectly aligned on the sides of the intersection. In addition to the regular parking lots on each street side there are extra parking lots just east of the

intersection on the Bismarckstraße. The vehicles moving in or out of the parking lot affect the traffic flow.

At this intersection the right-hand rule applies. Due to the proximity to a residential area and also a park at the corner of intersection, there is an increased frequency of cyclists and pedestrians. This makes a lot of interactions between the vehicles and cyclists at the intersection. Also, there are a considerable number of pedestrians, especially at the zebra crossing on the east of the intersection.

Figure 3.6 Aerial photo of Frankenburg intersection (inD 2), Aachen, Germany

- Heckstrasse (**inD 3**) is an unsignalized T-intersection joining Heckstraße and Von-Coels-Straße with Von-Coels-Straße being the priority road. This intersection is located in a suburban area far from the city core compared to the other intersections. There are cycle paths along the main road and also there is a traffic island in the middle for the pedestrian crossing. Therefore, there are interactions between the vehicles that are turning and the cyclists. Figure 3.7 shows an aerial photo of the intersection.

Figure 3.7 Aerial photo of Frankenburg intersection (inD 3), Aachen, Germany

3.2.3 Data Quality

Even though *inD* data was collected by drone-mounted camera, we did not find noise or irregular patterns in our visual investigations among the trajectories. The trajectories and intersection boundaries were neat and consistent across different recordings.

In the other hand, the intersection boundaries in *NGSIM* data were not fixed (in different recordings) and irregular patterns were observed (probably the same identifier assigned to two different road users). Above all, road users entering the survey zone were not detected from the very beginning of their presence within the boundaries. The latter especially created severe issues when we tried to cluster the start points.

Regardless of these data quality issues, we refrained from dataset-specific data cleaning to remain loyal to the main purpose of this study, which is lowering human intervention in data analysis. This way, we sought to find a method that would be robust and work in real-life

situations when facing uncleaned data without (or with minimum) human supervision of any kind.

CHAPTER 4 Article 1: Trajectory Clustering Performance Evaluation: If We Know The Answer, It's Not Clustering

Authors: Mohsen Rezaie[1], Nicolas Saunier[1]

[1]Polytechnique Montréal, Montreal, Canada

Submitted to IEEE Transactions on Intelligent Transportation Systems, April 2021.

Abstract

Advancements in Intelligent Traffic Systems (ITS) have made huge amounts of traffic data available through automatic data collection. A big part of this data is stored as trajectories of moving objects and road users. Automatic analysis of this data with minimizing required interference an operator would both lower the costs and eliminate subjectivity of the analysis.

Trajectory clustering is an unsupervised task. It is an effort to extract patterns without knowing labels which otherwise should be generated by a human operator. Trajectory clustering requires a measure to calculate similarity of the trajectories, a clustering algorithm to extract the patterns and a measure to evaluate its performance.

In this paper, we perform a comprehensive comparison of similarity measures, clustering algorithms and evaluation measures using trajectory data from seven intersections. We also propose a method to automatically generate trajectory reference labels based on their origin and destination points to be used for label-based evaluation measures. Therefore, the entire procedure remains man-made-label-free both in clustering and evaluation levels. Finally, we use a combination of evaluation measures to find the top performing similarity measures and clustering algorithms for each intersection.

4.1 Introduction

Data collection and analysis is the basis for transport planning and traffic management. Better serving users while minimizing the negative impacts of transportation systems requires better data. From automatic traffic counting, surveys, incident detection, road network state recognition to law enforcement tasks, transport management tasks all benefit from increasing amounts of data with good temporal and spatial coverage collected thanks to the advancements in hardware and software technology [15]. Exploiting the full potential of the massive data collected every day by Intelligent Transportation Systems (ITS) requires

automatic data management and analysis with minimum intervention from human operators.

Whether the data collection be through fixed or moving sensors, e.g. on vehicles or pedestrians, such as pole or drone mounted regular or infrared cameras, radars, LIDARs and Global Navigation Satellite System (GNSS) devices [4–7], various kinds of sensors provide user trajectories, i.e. the series of their positions over time. These trajectories are then processed in various ways for many transport applications, as noted above.

In some data analysis tasks such as motion prediction, the expected answer of the model for each trajectory is known, either retrospectively (for example in the case of time series prediction) or through expert annotation. Such a task is therefore supervised, i.e. a label is available for each trajectory. If the label is a discrete variable, it is a classification task, while if it is continuous, it is regression task. However, this is typically not the case for many other tasks such as automatically learning OD matrices, identifying trip patterns in the population, anomaly detection for example to spot incidents or law violations [28,64–68]. Such tasks consist in putting together similar trajectories into subsets of the whole dataset using a similarity measure or distance, also called clustering. These tasks are naturally unsupervised, as a good partition of the dataset or clustering depends on the application or purpose. Even when possible, annotation is time consuming, costly, error-prone and can be subjective or ambiguous.

We refer to the combination of clustering algorithm, distance or similarity metric and their parameters as the clustering setup. Finally, performance measures are used to evaluate how well the clustering method is working.

Even though clustering algorithms are unsupervised by definition, a large body of literature use labeled data to evaluate the performance of these models. Since the performance of a given clustering setup may depend on each dataset it is applied to, this approach means that some data must be labeled every time to measure the performance on new data, which is impractical and negates to some extent the need for clustering in the first place. This motivates our exploration of unsupervised performance measures for trajectory clustering.

In this paper we propose a framework for a comprehensive search of suitable trajectory-clustering setups with common unsupervised evaluation measures. We also propose a method to generate reference clusters for trajectories based on their origin and destination points to be used in supervised evaluation measures. We then use the results from the performance measures (both unsupervised and supervised) and combine them to generate a metric to compare several popular trajectory distances and clustering algorithms. The remainder of this paper is organized as follows. Section 4.2 outlines the prior literature related to trajectory clustering and discusses how this research builds on the prior work. Section 4.3 quickly

introduces the similarity measures, clustering algorithms and evaluation methods and then explains the steps of the proposed comprehensive comparison and evaluation method. Section 4.4 discusses the results and insights from testing the clustering algorithm on real-world data. Section 4.5 describes the conclusions of this research and the proposed future work.

4.2 Related Work

Trajectory-based road user detection, classification and tracking has been deployed for different traffic problems in various environments such as road user counting, speed measurement and movement detection (e.g. traffic violation detection), road user interaction analysis and automatic incident detection [8, 27–29].

A crucial part of time-series data clustering problems is choosing a suitable measure of similarity or distance. There are two categories of similarity measures used in time-series analysis: lock-step and elastic measures [23,24]. "Lock-step measures compare fixed (one-to-one) pairs of elements" [24] from input time series of the same length: well-known measures are the Bhattacharyya distance, the Euclidean Distance (ED) and more generally the distances derived from the L_p norms. On the other hand, elastic measures such as Dynamic Time Warping (DTW), distances based on Longest Common Sub-Sequence (LCSS), Edit distance with Real Penalty (ERP), Edit Distance on Real sequence (EDR) and Symmetric Segment-Path Distance (SSPD) can handle time-series of different lengths. Among these similarity measures, LCSS and DTW are very popular among researchers for their ability to compare trajectories with different lengths [21, 35, 36, 69]. However, this flexibility in input length usually comes at the cost of higher computational burden. For instance, the computational burden for the ED and Bhattacharyya distances is $O(n)$ while computing several elastic measures including DTW and LCSS are of $O(mn)$ where m and n is the lengths of the input time-series [25].

A body of literature focuses on trajectory clustering, using different Machine Learning (ML) approaches. Choong et al. used trajectory clustering for the traffic surveillance at intersections [70]. They used Longest Common Sub-Sequence (LCSS) for measuring similarity and then used Gaussian kernel function to convert LCSS similarity into distance. K-means and Fuzzy C-Means (FCM) are used for clustering and the results were compared against ground-truth through the Rand Index (RI).

Fu et al. used average pair-wise distance of first N points of each pair of trajectories to represent the dissimilarity of them, where N is the length of the shorter trajectory [52]. Then they used two layer clustering based on FCM, Spectral and hierarchical clustering algorithms and compared them against each other. For evaluation of performances they used

Tightness and Separation Criterion (TSC) which is based on within and between cluster dispersion of objects.

Zhao et.al [53] used Douglas-Peucker (DP) based compression to re-sample marine trajectory data. DTW is then used to calculate distances between the trajectories. Finally Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm is used for clustering. A custom performance metric based on ground-truth labels was used for evaluation of the method.

Belisle et al. worked on automatically counting vehicle movements at intersections [54]. Their method consists of vehicle detection and tracking, then trajectory clustering with a focus on parameter optimization. They used LCSS as similarity measure and a customized clustering algorithm similar to K-means. The algorithm picks trajectory prototypes to represent each cluster. For optimization and evaluation purposes, traffic turning counts were used as ground truth to calculate the Weighted Average Percentage Error (WAPE).

Even though different methods of similarity measurement and clustering algorithms are applied in trajectory analysis and novel methods have been purposed, unsupervised performance measurement is mostly overlooked. Given the heuristic nature of clustering approaches and the concern about their sensitivity to different data characteristics, errors in data and also in judgement [56], reiterating evaluation measurements for each site could be essential.

In this study, we focus on providing a framework to use several internal and external performance indices without ground-truth labels to facilitate easier clustering evaluations. More specifically, we propose a method to generate reference labels for supervised performance metrics and use a combination of supervised and unsupervised performance measures.

4.3 Methodology

4.3.1 Clustering Setup

For each site of interest, let D be the set of road user trajectories within the boundaries of the site. Let each trajectory $M_i \in D$ be a time-ordered sequence of m_i time-stamped positions of road user i as (x_j^i, y_j^i, t_j^i) where x_j^i and y_j^i represent the coordinates of the road user at timestamp t_j^i, $1 \leq j \leq m_i$.

Clustering algorithms split the dataset into groups with similar objects based on a similarity or distance measure. According to the literature, we use the terms "distance" and "dissimilarity" in their broad meaning to mention a method that measures unlikeness of a pair of trajectories. Value of a distance measure is essentially non-negative and the more alike a pair

of trajectories are, the lower is the distance value. On the other hand, a similarity measure represents a method of comparing trajectories which assigns (still non-negative) higher values when two trajectories are more alike [71].

(Dis)similarity Measures	Parameters	Experimental Values
DTW	-	-
LCSS	r_b	1, 2, 3, 5, 7, 10 m
EDR	r_b	1, 2, 3, 5, 7, 10 m
PF	w	0.01, 0.05, 0.1, 0.2, 0.3, 0.5
Hausdorff	-	-
SSPD	-	-

Table 4.1 Similarity measures

Table 4.1 shows an overview of the (dis)similarity measures used in this study and their parameters. For all the measures in Table 4.1, the Euclidean distance was used for calculating point-to-point distances. LCSS and EDR both have a parameter r_b which is the threshold on point distances to decide if two points are close enough and may be part of their similar subsequence.

Since LCSS provides a similarity measure, we convert it to the distance d_{LCSS} as suggested by Vlachos et al. [34]:

$$d_{LCSS}(M_i, M_j) = \frac{1 - LCSS(M_i, M_j)}{min(m_i, m_j)} \tag{4.1}$$

where $LCSS(M_i, M_j)$ is the LCSS similarity between trajectories M_i and M_j, i.e. the lengths of the longest common subsequence between them.

Given a distance measure d_α with parameters α, all distances $d_{i,j}$ between all pairs of trajectories $< M_i, M_j >$ in D are calculated and stored in matrix T_{D,d_α}. The distance matrix T_{D,d_α} is then used by a clustering algorithm A^1 with parameters β to split D into k groups. We define the clustering setup as $\omega = (s, \alpha, A^1, \beta, k)$. Even though there are algorithms which do not use the whole matrix but rather some of its elements (e.g. DBSCAN and spectral clustering), pre-computing the matrix saves time since it is used repeatedly in our experiments for different clustering algorithms A^1 and number of clusters k.

In this study, we used six popular clustering algorithms used for trajectory clustering, namely, k-Medoids [40], agglomerative hierarchical clustering [41], spectral clustering [42], Density-based spatial clustering of applications with noise (DBSCAN) and Ordering Points To Identify The Clustering Structure (OPTICS).

It is worth mentioning that DBSCAN and OPTICS do not take into account a number of clusters as input. In these two cases, the number of clusters k is a result of the algorithm and cannot be controlled directly. The table 4.2 summarizes the clustering algorithms used in this study with their corresponding parameters (in addition to the number of clusters when possible).

Clustering Algorithm	Parameters
k-medoids	-
agglomerative clustering	$linkage \in \{complete, average, single\}$
spectral	-
DBSCAN	n_z^{min}, d_z
OPTICS	n_z^{min}

Table 4.2 Clustering algorithms

Figure 4.1 shows an example of trajectories recorded from an intersection in Atlanta, Georgia, after trajectory clustering.

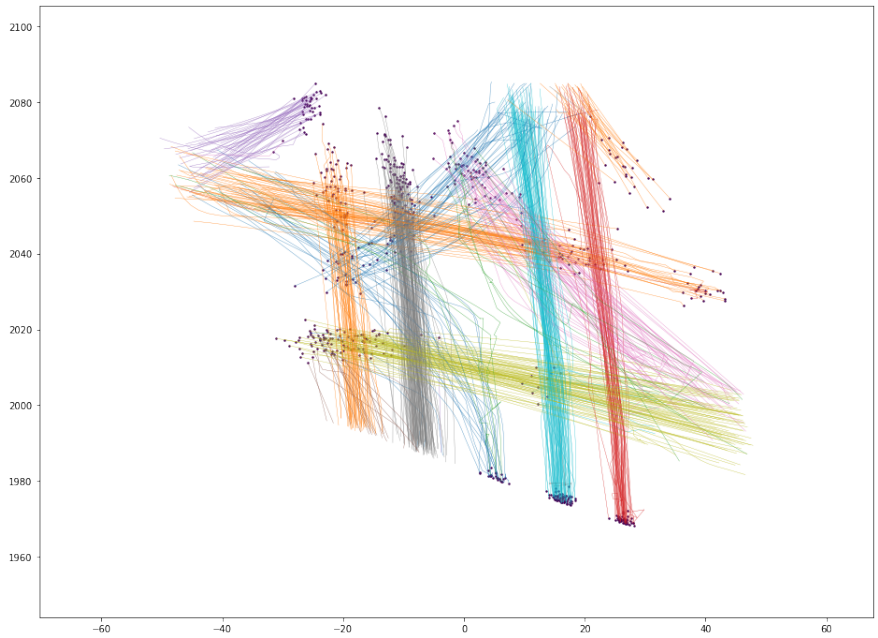


Figure 4.1 Example of trajectories D in an intersection after clustering. Clusters are represented by different colors (in some cases the colors assigned to different clusters are similar due to the limit on number of colors while maintaining the contrast high enough to keep them visually distinctive).

For the reasons highlighted in the introduction of cost, time and subjectivity of manually labelling trajectory clusters, the focus of this work is on the unsupervised evaluation of the

performance of methods for trajectory clustering. Besides, there is an inherent contradiction to the use of labels for clustering: if labels can be unambiguously assigned, it means the task can be supervised and is not a clustering task.

Thus, there are no reference to check the clustering results presented in Figure 4.1. While we may visually confirm that the generated clusters are meaningful and can be considered as representatives of different manoeuvres in the intersection, this is not always the case. The clusters presented in Figure 4.1 result from a clustering setup ω picked based on the results of this work. Indeed, it is not trivial to make the choice of ω and the best clustering setup may be different from one location to another because of different characteristics of the road configuration and the road user trajectories. The next sections introduce the performance metrics used in this work and how supervised performance metrics can be used without ground truth.

4.3.2 Unsupervised Performance Measure

There are several unsupervised performance metrics for clustering evaluation. These metrics only rely on distance values d_α to evaluate the clustering setup ω. The silhouette (S) [44] is one of these metrics which is widely used in the literature for trajectory clustering evaluation [21]. It is defined for each object i as [44]:

$$s_i = \frac{b_i - a_i}{max(a_i, b_i)} \tag{4.2}$$

Where a_i is the mean distance between the object i and all other objects within the same cluster, and b_i is the smallest of the mean distances between the object i and the objects in another cluster (different from the one object i belongs to. The total clustering performance based on the Silhouette S is then calculated as the average silhouette coefficient over all objects.

Other unsupervised performance measures like he Calinski-Harabasz Index (CHI) [45] and the Davies-Bouldin Index (DBI) [46] cannot be used as they rely on the computation of cluster centers as averages of all objects in each cluster, which cannot be done simply for variable length vectors such as trajectories.

However, unsupervised performance measures such as Silhouette are mostly suitable for comparison between clustering setups with the same algorithms since they make assumptions about cluster structure [21]. Also, unsupervised performance measures cannot reflect results expected for specific applications. If an expected, even partial, reference clustering is

known, generating a result that matches this clustering may not yield the best unsupervised performance measure.

We have observed that if two trajectories are mistakenly considered similar by A^1 based on distance d, an unsupervised performance measure based on d may make the same mistake and confirm A^1's decision. Figure 4.2 shows an example case in which the silhouette method mistakenly confirms the clustering results. As one can see, the cluster 1 should be split in two groups, each being merged with either cluster 6 or 20. Thus, while we expect several negative silhouette values for the trajectories in the first cluster, they are mostly positive and confirm the clustering mistakes.

(a) Clustered trajectories with colors showing the closest cluster for each trajectory, computed as the mean distance to all elements in that cluster.

(b) Silhouette values for trajectories in the clusters shown in the left

Figure 4.2 Example of error in clustered trajectories and their silhouette values

4.3.3 Weak Trajectory Clustering Supervision using Origin-Destination

To overcome this problem, we suggest that beside using common unsupervised evaluation measures, a method independent of the distance d and trajectory clustering algorithm A^1 be used to validate the result of trajectory clustering. The method that we propose here is generating trajectory "reference" clusters based on clustering simply the first and last points of the trajectories (origins and destinations) and using these reference clusters with

supervised evaluation measures.

To do so, we extract the origins and destinations of the trajectories in D put them in two sets, respectively D_O and D_D. The points in each set are clustered using the Euclidean distance and the clustering algorithm A^2, which may not be the same as A^1, with the parameters β_O and β_D and number of clusters k_O and k_D, respectively. The steps are provided in detail in Algorithm 1.

The elbow method is used to make good choices for k_O and k_D, since it is a popular technique in point clustering [72]. Figure 4.3 illustrates the average distances $\bar{d}$ with respect to k_O and k_D: in this case we picked $k_O = 8$ and $k_D = 4$. Figure 4.4 depicts an example of the origin and destination clusters for the trajectories in the same intersection of the NGSIM dataset (which will be introduced in section 4.4.1). If the proportion of trajectories starting from origin cluster ϕ_O and destination cluster ϕ_D is more than a threshold $\epsilon = 0.01$, i.e. over 1 % of all trajectories, the origin-destination pair is deemed significant and we assign label ϕ_{OD} to the corresponding trajectories.

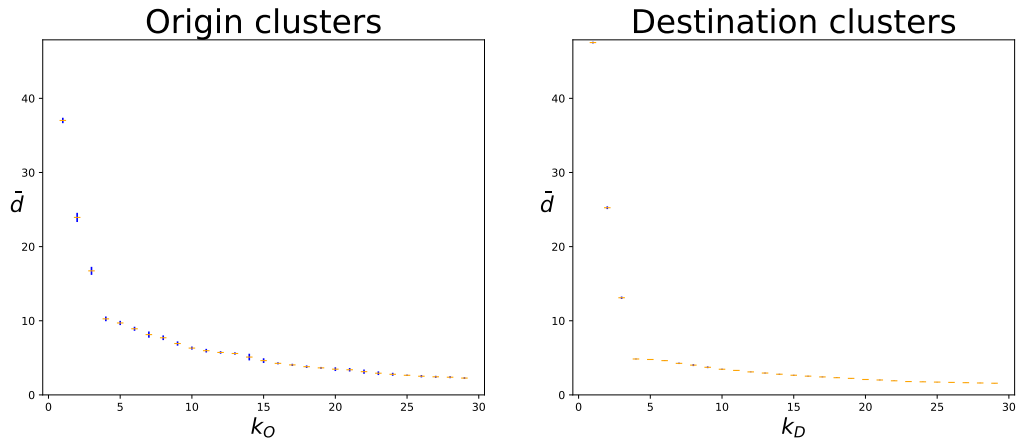


Figure 4.3 Average distances for the datasets of origins (left) and destinations (right) with respect to the number of clusters. The mean plus and minus two standard deviations are computed over several replications of the clustering algorithm for each number of clusters. The graphs are produced based on data from an intersection in NGSIM dataset using the agglomerative hierarchical clustering algorithm with average linkage as the merging criterion.

Using these reference clusters, we calculate six additional performance measures, namely, the completeness, homogeneity and Validity (V) measures [47], the Adjusted Rand Index (ARI) [49], Adjusted Mutual Information (AMI) [48] and Fowlkes-Mallows Index (FMI) [50]. These measures represent the agreement or similarity between the reference clusters and the clusters generated by the given setup ω, ignoring permutations of the clusters. In particular,

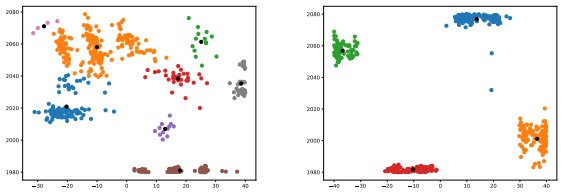


Figure 4.4 Trajectory origin (left) and destination (right) clusters with $k_O = 8$, $k_D = 4$ and agglomerative hierarchical clustering algorithm with average linkage.

for ARI and FMI, the measures rely on the joint classifications of object pairs, whether they are both in the same or different clusters in the reference and result of ω.

- **Completeness, homogeneity and V** [47]: for a cluster assignment K, we define entropy of K:

$$H(K) = -\sum_{i=1}^{k} \frac{|C_i|}{n} . \log \frac{|C_i|}{n} \tag{4.3}$$

The entropy $H(Ref)$ of the reference clusters Ref is defined accordingly. Let k_{Ref} be the number of reference clusters. The conditional entropy is also defined, for example of Ref given K $H(Ref|K)$:

$$H(Ref|K) = -\sum_{r=1}^{k_{Ref}} \sum_{i=1}^{k} \frac{n_{r,i}}{n} . \log \frac{n_{r,i}}{n_i} \tag{4.4}$$

Where $n_{r,i}$ is the number of objects belonging to reference cluster r and cluster i. The conditional entropy $H(K|Ref)$ of K given Ref is defined in a symmetric manner. We can then finally define the performance measures completeness (c) and homogeneity (h), and V as their harmonic mean:

$$c = 1 - \frac{H(K|Ref)}{H(K)} \tag{4.5}$$

$$h = 1 - \frac{H(Ref|K)}{H(Ref)} \tag{4.6}$$

$$V = \frac{(1+\beta) \times h \times c}{\beta \times h + c} \tag{4.7}$$

We use $\beta = 1$ to give equal weights to the completeness and homogeneity measures.

- **Adjusted Mutual Information (AMI)**: we first define the probabilities $P(r) = \frac{|Ref_r|}{n}$ and $P'(i) = \frac{|C_i|}{n}$ of a randomly picked object to belong respectively to reference cluster r and cluster i, and $P(r,i) = \frac{n_{r,i}}{n}$, we define the Mutual Information (MI) [48]:

$$MI(Ref, K) = \sum_{r=1}^{k_{Ref}} \sum_{i=1}^{k} P(r,i) \log \frac{P(r,i)}{P(r)P'(i)} \tag{4.8}$$

  Since the MI favors clustering with a higher number of clusters k, we use the Adjusted Mutual Information (AMI) instead:

$$AMI = \frac{MI - E[MI]}{mean(H(Ref), H(K)) - E[MI]} \tag{4.9}$$

  Where $E[MI]$ is the expected value of the MI index [48].

- the **Adjusted Rand Index (ARI)** is a simple, yet popular index used for external clustering performance. It is defined as [49]:

$$RI = \frac{a+b}{C_2^n} \tag{4.10}$$

  Where a is the number of object pairs that are in same cluster in both Ref and K, b is the number of object pairs that are in different clusters in both Ref and K, and $C_2^n = \frac{n(n-1)}{2}$ is the total number of object pairs in the dataset. For a random label assignment, RI is not necessarily zero. Instead, we use the Adjusted Rand Index (ARI) which fixes this using the expected value of RI $E[RI]$:

$$ARI = \frac{RI - E[RI]}{max(RI) - E[RI]} \tag{4.11}$$

- the **Fowlkes-Mallows Index (FMI)** is also based on the classification of the object pairs, considering two objects being in the same clusters in both assignments as the positive class and two objects being different ones as the negative class. For such a binary classification problem, the numbers of True Positives (TP), False Positives (FP) and False Negatives (FN) can be counted and the Fowlkes-Mallows Index (FMI) is calculated as the geometric mean of precision and recall: [50]:

$$FMI = \frac{TP}{\sqrt{(TP+FP)(TP+FN)}} \tag{4.12}$$

Evaluation Measures	Need for Reference Clusters	Higher-the-better	Range
S	✗	✓	$[-1, 1]$
completeness	✓	✓	$[0, 1]$
homogeneity	✓	✓	$[0, 1]$
V	✓	✓	$[0, 1]$
AMI	✓	✓	$[-1, 1]$
ARI	✓	✓	$[-1, 1]$
FMI	✓	✓	$[0, 1]$

Table 4.3 Evaluation metrics

Table 4.3 provides an overview of the performance measures used in this work.

The trajectory dataset was permuted $n = 10$ times in order to make the results more robust for each clustering setup ω. For each clustering setup, the mean and standard deviation are computed for each performance measure, to compute intervals of the performance measures as the mean plus or minus two standard deviations which are plotted in the result section.

Finally, we use a combination of these evaluation measures to identify top performing clustering setup ωs. For each performance measure v and each ω we calculate the average performance $\bar{v_\omega}$, and, assuming t-student distributions (unknown standard deviation), we calculate the lower bound of the confidence interval for 95 % confidence level v_ω^{lower}:

$$v_\omega^{lower} = \bar{v}_\omega - t_{95\%,9} * \frac{\hat{\sigma}_{v,\omega}}{\sqrt{n}} \tag{4.13}$$

Then for a given evaluation measure v, we use v_ω^{lower} to order ω and assign rank rv, ω to each setup ω. Finally we use average of these rankings over different evaluation measures v to calculate combined performance of the clustering each clustering setup $\bar{r_\omega}$. A formal description of the computation of the clustering performance is presented in Algorithm 2.

4.4 Experimental Results

All the codes developed in this work to produce the results are available online under an open source license in (this GitHub repository).

4.4.1 Datasets

We have have tested our framework on seven sites from two datasets:

Algorithm 1: Clustering trajectories based on their origins and destinations to produce reference clusters

input : Sets of trajectories D with derived sets of origins D_O and destinations D_D, minimum and maximum number of clusters k_{min} and k_{max}, cluster size threshold ϵ, clustering algorithm A^2

output: Set Φ_{OD} of trajectory clusters ϕ_{OD}

for $k_O \in [k_{min}, k_{max})$ **do**

  Cluster trajectories based on their origins using A^2 into k_O clusters;

  Let Φ_O be the set of all clusters ϕ_{O_i} with average center c_{O_i};

  Calculate the average within-cluster sum of squares or average distance $\bar{d}_{k_O} = \frac{\sum_{i=1}^{k_O} \sum_{p \in \phi_{O_i}} ||p - c_{O_i}||}{|D|}$ using the L_2 norm $||\ ||$;

Draw the graph of $\bar{d}_{k_O}$ values against k_O;

Pick the $\tilde{k}_O$ where an unusual drop in the $\bar{d}$ values occurs;

Repeat all the steps above for D_D and pick $\tilde{k}_D$;

Split D into $\tilde{k}_{OD} = \tilde{k}_O * \tilde{k}_D$ clusters depending on the cluster of the origin and destination of each trajectory;

return the set Φ_{OD} of all trajectory clusters ϕ_{OD} containing at least ϵ trajectories;

NGSIM

The first dataset used in this work is extracted from a dataset made publicly available by the Next Generation SIMulation (NGSIM) program [60]. It is a popular dataset containing trajectory data collected at four sites in the U.S. and used among traffic researchers for benchmarking their works. It consists of positions of vehicles moving in or crossing the five lane Peachtree street in Atlanta, Georgia . The data is recorded at 10 Hz during both the morning and afternoon rush-hours from multiple synced cameras mounted on tall buildings to cover the roads over several hundreds meters. The (sub-)trajectories within the boundaries of four intersections were extracted from this dataset. Figure 4.5a shows an aerial photo of intersection NGSIM 1.

inD

The second set of trajectories is from the intersection Drone (inD) dataset, collected from three intersections in Aachen, Germany from 2017 to 2019 [63]. The trajectories are extracted from video recordings of around 20 minutes recorded via a drone-mounted camera at 25 Hz frame rate. Figure 4.5b shows an aerial photo of intersection inD 2.

The table 4.4 gives a summary of sites used for the experimental results.

Algorithm 2: Computing the clustering performance measures with permutation of the trajectory dataset

input : Sets of unsupervised and supervised performance evaluation measures V^u and V^s, number of iterations n, subset of trajectories $\tilde{D} = \cup_{\phi_{OD} \in \Phi_{OD}} \phi_{OD}$

output: Average performance rank for the clustering setups $\bar{r_\omega}$

for *each candidate* d_α **do**
  Calculate distance matrix T_{D,d_α} with elements $d_{i,j} = d_\alpha(M_i, M_j)$;
  for $l \in [1, n]$ **do**
    Permute randomly $\tilde{D}$;
    for *each candidate clustering algorithms* A^1*, parameters* β *and number of clusters* k **do**
      Cluster $\tilde{D}$ with the setup $\omega = (d, \alpha, A^1, \beta, k)$;
      Let Φ_l be the set of resulting clusters;
      For all $v \in V^u$ calculate performance measure $v_{\omega,l} = v(\Phi_l)$;
      For all $v \in V^s$ calculate performance measure $v_{\omega,l} = v(\Phi_l, \Phi_{OD})$;
for $v \in V^u \cup V^s$ **do**
  for *each* $\omega = (s, \alpha, A^1, \beta, k) \in \Omega$ **do**
    Calculate average performance measure $\bar{v}_\omega = \frac{\sum_{l=1}^{n} v_{\omega,l(\Phi_i)}}{n_{iter}}$;
    Calculate lower bound $v_\omega^{lower} = \bar{v}_\omega - t_{95\%,9} * \hat{\sigma}_{v,\omega}$;
  Order all clustering setups ω based on v_ω^{lower} and assign rank $r_{v,\omega}$;
for $\omega \in \Omega$ **do**
  Calculate average rank $\bar{r}_\omega = \frac{\sum_{v \in V^u \cup V^s} r_{v,\omega}}{|V^u \cup V^s|}$

Site	Camera	Intersection Types	Traffic Lights
inD 1	drone	4-way	✗
inD 2	drone	4-way	✗
inD 3	drone	T-intersection	✗
NGSIM 1	building	4-way	✓
NGSIM 2	building	4-way	✓
NGSIM 3	building	4-way	✓
NGSIM 4	building	4-way	✓

Table 4.4 Site summary with their characteristics (multiple synced cameras were mounted on buildings for the NGSIM datasets).

4.4.2 Performance Plots

Figure 4.6 shows the performance results for *NGSIM 1* intersection based on the EDR similarity measure with parameter $w = 7$. Each of the nine plots in the figure shows one of the seven performance measures as a function of the number of clusters k used in the clustering.

Given seven intersections studied in this work, three distances d with no parameters and distances having a parameter with six tested parameter values α, there are $7 * 21 = 147$ figures similar to Figure 4.6. Given that the general patterns observed here in Figure 4.6 is not necessarily shared in the other 146 figures, analyzing such enormous amount of data and finding the best clustering setup $\omega = (d, \alpha, A^1, \beta, n_{A^1}, k)$ is not feasible through visual comparison of those 147 figures.

4.4.3 Performance Measure Combination

To summarize the information and ease the decision process on picking the best setup, we want to combine the performance results. However, some evaluation measures may be correlated with each other and including all of them regardless of the correlations would overemphasize the performance features corresponding to these correlated measures. Figure 4.7 shows the correlations between the seven performance measures for an *NGSIM 1* intersection[1].

Based on correlation plots from the seven intersections, evaluation measures V, homogeneity, FMI and S are all highly correlated together with absolute correlation values between them above 0.75 for all pairs of measures in all intersections. Therefore, we only use S, Completeness, Homogeneity, ARI and AMI for the combined metric.

The ten best setups for each site are picked based on the average rank $\bar{r}_\omega$. The Figure 4.8 shows the top performing distances and clustering algorithms for each site, more precisely the proportion of the 10 best setups in which each distance and algorithm appears. The first observation is that there is no one clustering setup and not even a specific distance measure or clustering algorithm that outperforms systematically the others or at least always appears among the top ten best clustering setups for all sites. Hierarchical clustering is the most common clustering algorithms appearing in the top ten for all sites. Though not similarly frequent, Spectral algorithm is also showing some consistency in appearing among the top performers in most, although not all, the sites. For distances, SSPD and Hausdorff appear frequently, especially for the NGSIM datasets, while the LCSS appears in the top ten for two inD datasets.

The second observation is that there are clear winners for specific sites. Each site has one, up to three, algorithms or distances outperforming the others (in some cases, LCSS or hierarchical clustering appear several times in the top with different parameters). However, there is no clear pattern in these observations across sites, whether some dataset characteristics

[1]Given the limits of space in a paper, the correlation results were provided for a single intersection. In this thesis version, the correlation results for all intersections are provided in Appendix A

may explain to some extent the heterogeneity or whether trajectory clustering is intrinsically sensitive to the input data.

4.5 Conclusion

In this paper, we proposed a way to combine unsupervised and supervised clustering performance metrics without any manual annotation of the trajectories. Namely we tried six distances, three of them each tried with six parameter values, i.e. a total of 21 distances, six clustering algorithms, different numbers of clusters when used as input by some the clustering algorithms ranging from 2 to 30 and used nine performance measures. We did the experiments on seven intersections from two different datasets, i.e. collected in different countries, different road types and through different data collection methods.

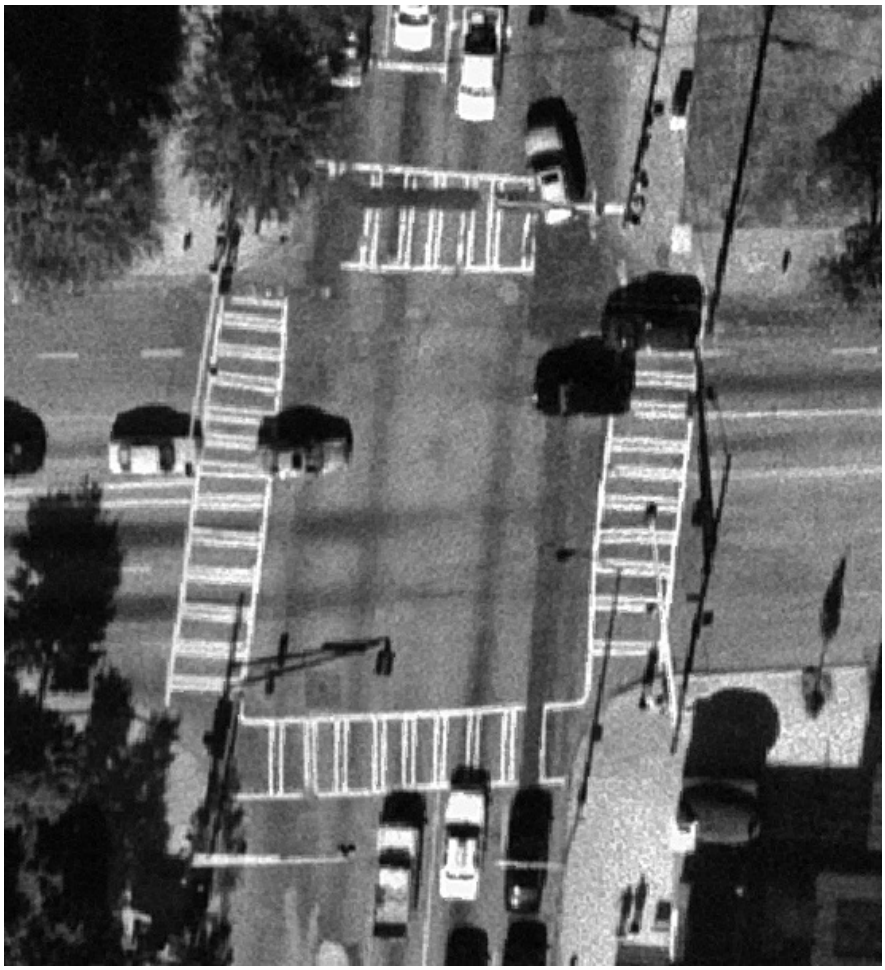
Even though usually useful, unsupervised performance measures use the same distance d used in the clustering, and their results often align with the clustering results As complementary performance measures, we proposed a method to generate reference clusters based on the origins and destinations of the trajectories to be used in supervised performance measures.

Finally, using a metric based on a combination of performance measures, we picked the top performing distances and clustering algorithms. The results show that there is no single combination of distance and clustering algorithm that is always among the top ten clustering setups. Even looking at them independently, the finding was similar. There are distances that performed better compared to the others in specific intersections, but none of them are present among the top performers for every intersection. Among the clustering algorithms, agglomerative hierarchical clustering was the only one that was among the top performers in all seven intersections, though not always with the same linkage type. Also the plots do not show a visual dominance of it over other clustering algorithms based on frequency of its appearance among the top performers for every intersection.

Based on these findings, we believe that the choice of similarity measure and clustering algorithm greatly depends on the intersection type, environment and road user behaviors. Therefore, we suggest that similar comparison procedures be followed for new sites among the candidate setups. To facilitate this, we have made all the essential codes developed in this work available online under an open source license (GitHub repository).

A systematic search of the parameters of the clustering algorithms and distances to determine whether some results depend on the choice of parameters. Further work is necessary to better characterize clustering performance, e.g. based on domain specific tasks like origin-destination clustering, and apply these to more datasets and better characterize the factors that may

determine clustering performance.



(a) Site map of intersection NGSIM 1 (one out of four)

(b) Site map of intersection inD 2 (one out of three)

Figure 4.5 An example of intersection for each of the two datasets

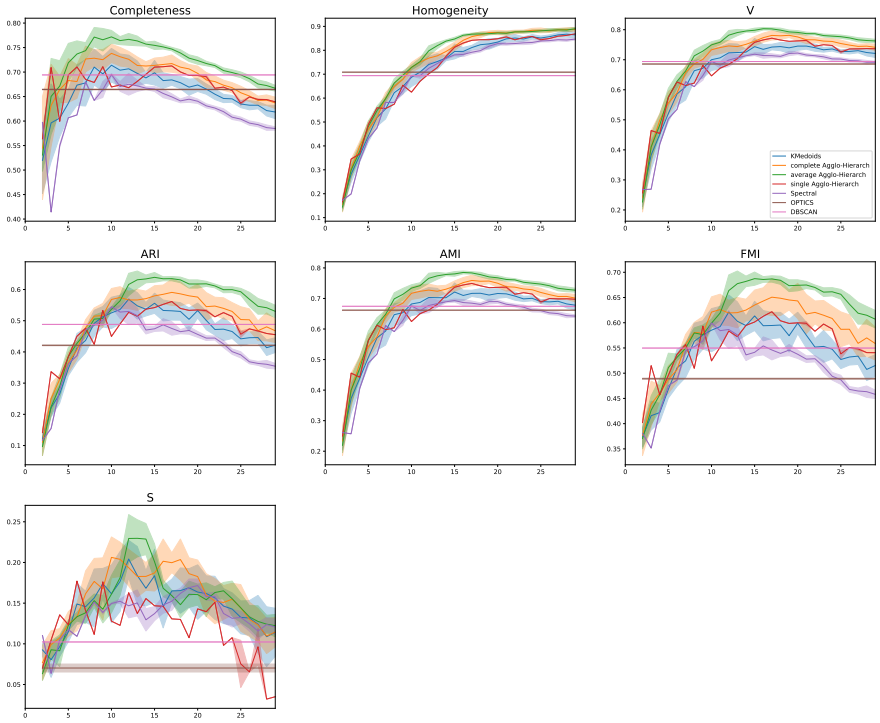


Figure 4.6 Performance results for the *NGSIM 1* intersection based on EDR similarity measure with parameter $w = 7$ m

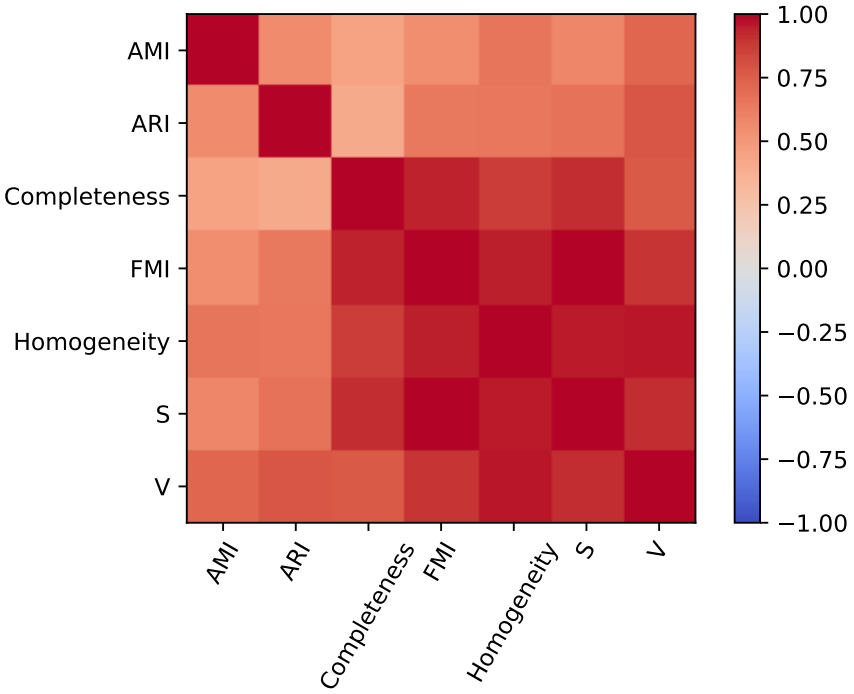


Figure 4.7 Correlation of the performance measures for the *NGSIM 1* intersection

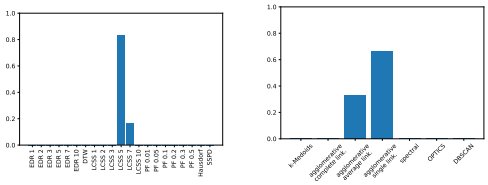


Figure 4.8 inD intersection 1

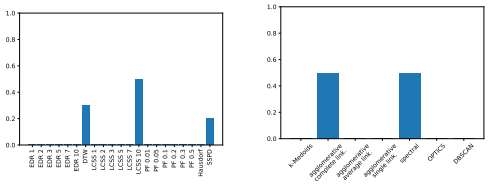


Figure 4.8 inD intersection 2

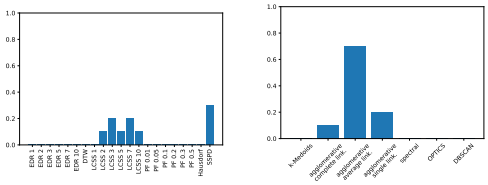


Figure 4.8 inD intersection 3

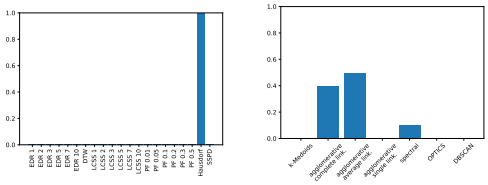


Figure 4.8 NGSIM intersection 1

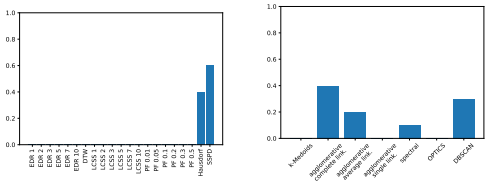


Figure 4.8 NGSIM intersection 2

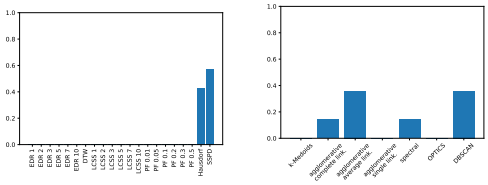


Figure 4.8 NGSIM intersection 3

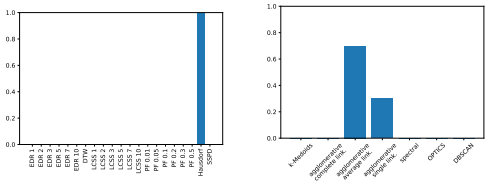


Figure 4.8 NGSIM intersection 4

CHAPTER 5 METHODOLOGICAL ASPECTS AND COMPLEMENTARY RESULTS

5.1 Some Details On Implementation

The entire procedure of trajectory clustering and performance evaluation explained in this work involved different tasks: loading data, creating trajectories, calculating distance matrices, trajectory clustering, origin/destination clustering and performance evaluation. We performed over 36,000 clustering for each of the seven sites. To keep the total run-time reasonable, we had to take different measures to ensure light and fast performance of the code. Here we explain three of those main measures that we took to boost the speed.

5.1.1 Database Management System

Given the diverse type of tasks in our method that needed to be coded, *Python* was used as the main framework for programming. *Python* is an open-source, general-purpose programming language which is frequently used in scientific researches as scripting language [58, 59]. With all its capabilities, running simple data queries is not one of *Python*'s strengths. Also to work with data, first it should load it on RAM. This method is not ideal for large datasets. To address these limits and keep the framework scalable, the use of a database management system was essential.

Candidates were Structure Query Language (SQL) based software which are solutions designed for managing data held in Relational Database Management System (RDBMS). Some of this software even supports some spatial functions out of the box. However, installing a second software would have imposed an extra burden on the operator. Also it would have prevented the full integration and interruption-free flow of the framework.

Another candidate was Pandas which is a module provided for Python. This is a popular tool for data manipulation and analysis. However, its speed is not on par with SQL-based software (specially in *SELECT* queries) [73]. Also, it loads the entire data on RAM which can be quite demanding for large datsets and limit scalability.

Finally, SQLite was picked for data management in the framework which is an embedded database software available for many programming languages. Since it comes bundled with Python as a module, the framework remains interruption-free and no further software installation by operator is required. Despite the lack of built-in spatial functions and visualization (which was especially a disadvantage in the coding phase of the project), similar to other

SQL-based software, it works fast and is not demanding for RAM.

5.1.2 Low-level Compilation And Parallelization

Another area that was considered to save run-time was distance-matrix calculation. Given that such calculation for a set of n_D trajectories is of complexity $O(n_D^2)$ (also distance calculation for a pair of trajectories with lengths m^1 and m^2 is of complexity $O(m^1, m^2)$), lowering the run-time of this step of the framework was crucial for scalability.

To boost the speed, the possibility of compiling down the Python codes to native machine code (within Python) and also parallelization was investigated. The two popular options are the Cython and Numba packages. Both methods support both low-level compilation and parallelization [74], although Numba may be faster in some situations [75]. It is worth mentioning that these benefits come at the cost of severe restrictions in form of losing access to many tools available in Python (such as Scipy package and some basic functions from Numpy package). Also, functions compiled using Numba can only take limited input types (that should be strictly pre-defined). Thus, functions and classes cannot be used as input for such functions.

Both solutions were tried. For Numba, the method was implemented in the code, and for Cython, a package for calculating popular trajectory distances developed by Guillouet [76]. Although neither methods could be run on GPU, Numba could still run on CPU cores simultaneously, thus running several times faster. Therefore, Numba method was used as the main solution in the framework, except for Hausdorff and SSPD distances which required passing functions as input to the main distance functions.

5.1.3 Caching The Objects

When conducting a comparison among clustering setups, naturally the run-time grows with an increase in the number of setups. As a reference, in the comparison carried out in this study (explained in Chapter 3) with over 3600 setups and 10 iterations for each setup (using the final/tuned version of the code), about 85 % of the total run-time was consumed for the trajectory clustering and evaluation tasks while less than 15 % of the total run-time was used for trajectory extraction, calculation of distance matrices and extraction of OD reference labels, altogether.

Therefore, even though the trajectory clustering and evaluation tasks are complex and use of Numba and Cython methods are either not feasible or very hard to implement because of the limits of these two methods, given the significant portion of run-time consumed by this

part of the framework in comprehensive comparisons, even a marginal boost in speed can save considerable amount of time.

To do so, a simple, yet effective approach is caching the objects that are frequently called in the code. In this case, both sets of clustering algorithms and performance measures were loaded and cached on the memory early in the for-loops so that the call-time would be saved. To do so, each clustering algorithm was loaded and cached with different parameter values. Also the input for both clustering algorithms and performance metrics were standardized to require minimum manipulation of objects later in the code.

5.2 Further Results

5.2.1 Correlation Of The Distance Measures

Figure 5.1 represents the correlations between the distance measures with different parameter values. While the correlation values are positive in most of the cases, we can see interesting patterns there.

First of all, the distance measures between variants of the same method with different parameter values show the highest correlations values. Considering different distance methods, SSPD and Hausdorff are highly correlated, which is expected as their calculation method is very similar. It is also consistent with our observation that they both appear frequently among the top performing distance measures (Figure 4.8).

PF has the highest correlation with DTW. This could also be expected given the similarity of their calculation methods: both reflect average point-to-point distances with points compared in order.

Another observation is that the smallest correlation coefficients (in some cases even negative) are between EDR and three other distance methods, namely, DTW, SSPD and Hausdorff. This is also in line with the findings of our proposed evaluation method. As we can see in Figure 4.8, contrary to DTW, SSPD and Hausdorff, EDR does not appear among distances with the highest average rank in any of the sites.

5.2.2 Sensitivity To Data Permutation

Although changes in performance measures based on change in sample data are expected, the sensitivity varies among different setups. Even the same algorithm may show different performance values when paired with different distance measures.

The Figure 5.2 shows the performance of different clustering algorithms based on the LCSS

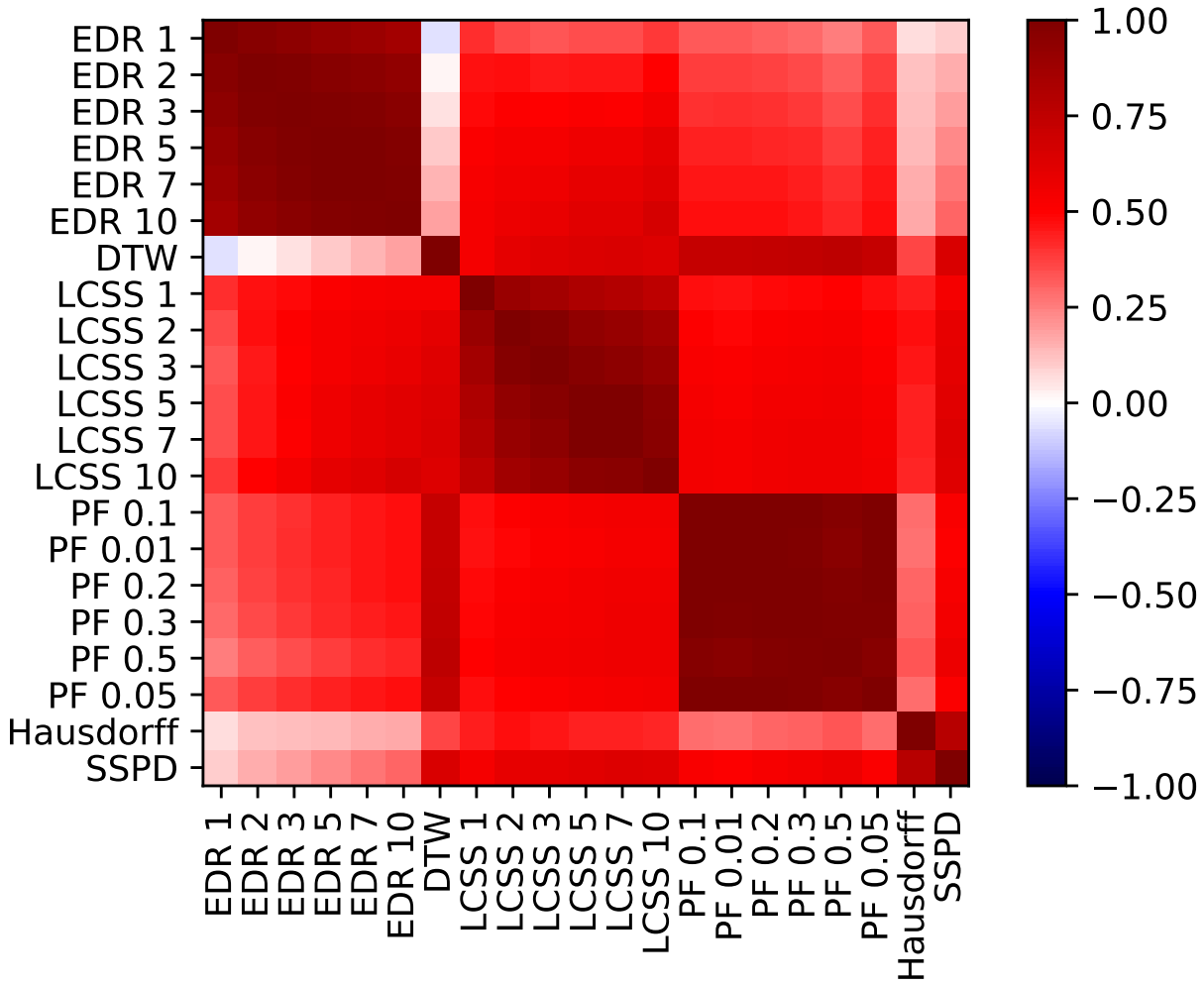


Figure 5.1 Correlation between distances measures with different parameter values

distance with $d_z = 5$ m for intersection NGSIM 2. We can see that performance variation is considerably higher for agglomerative hierarchical clustering. However, such high sensitivity is not observed for the same pair of algorithm and distance on every other datasets. Figure 5.3 shows the performance of the algorithms with the same distance measure on intersection inD 1. We can see that agglomerative clustering has a performance variation similar to the other algorithms.

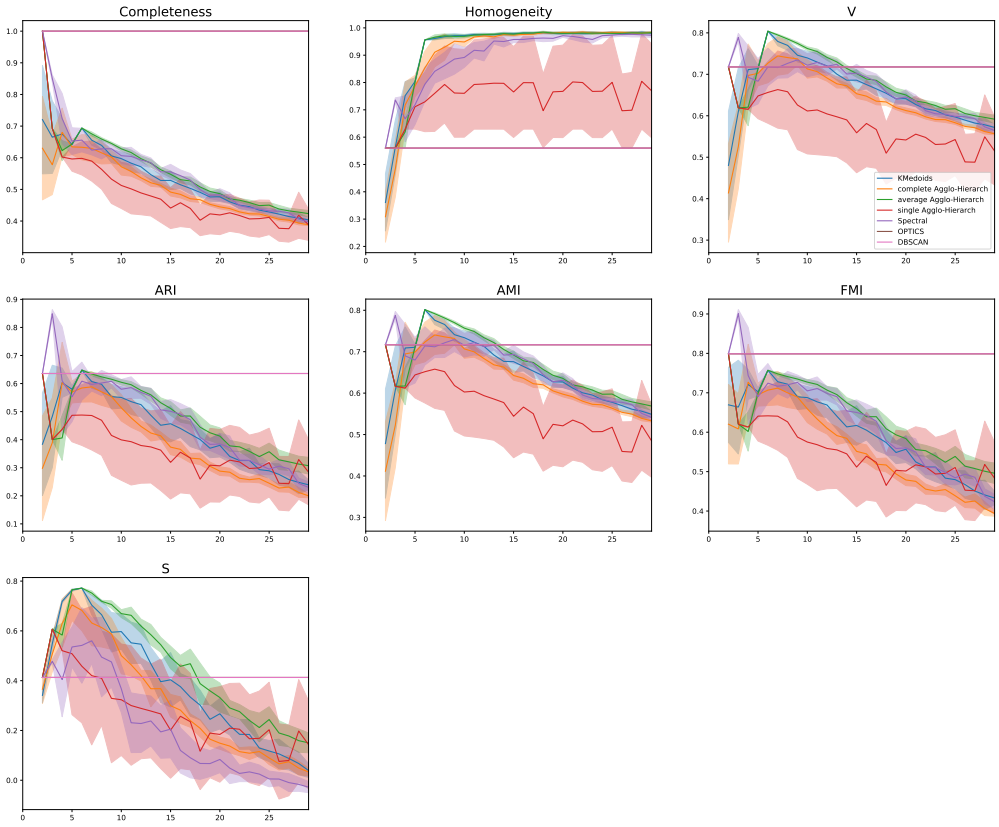


Figure 5.2 Performance variation of agglomerative clustering paired with LCSS ($d_z = 5\ m$) on NGSIM 2

5.3 Extra Experiments Using Centroid-based Algorithms And Performance Measures

k-Means is claimed to be "by far the most popular clustering algorithm used in scientific and industrial applications" [77]. This clustering approach looks for k arbitrary points from the feasible space representing the cluster centers that would minimize the sum of distances of all data points from their corresponding closest cluster centers [78]. A simple algorithm suggested by Lloyd [79] suggests picking k arbitrary initial cluster centers and assigning observations in the dataset into their closest cluster center. Once the clusters are formed, the cluster centers are calculated by averaging over the cluster points, and the procedure is repeated until the clusters are stable [78]. This algorithm has the complexity of $O(nk)$ [40].

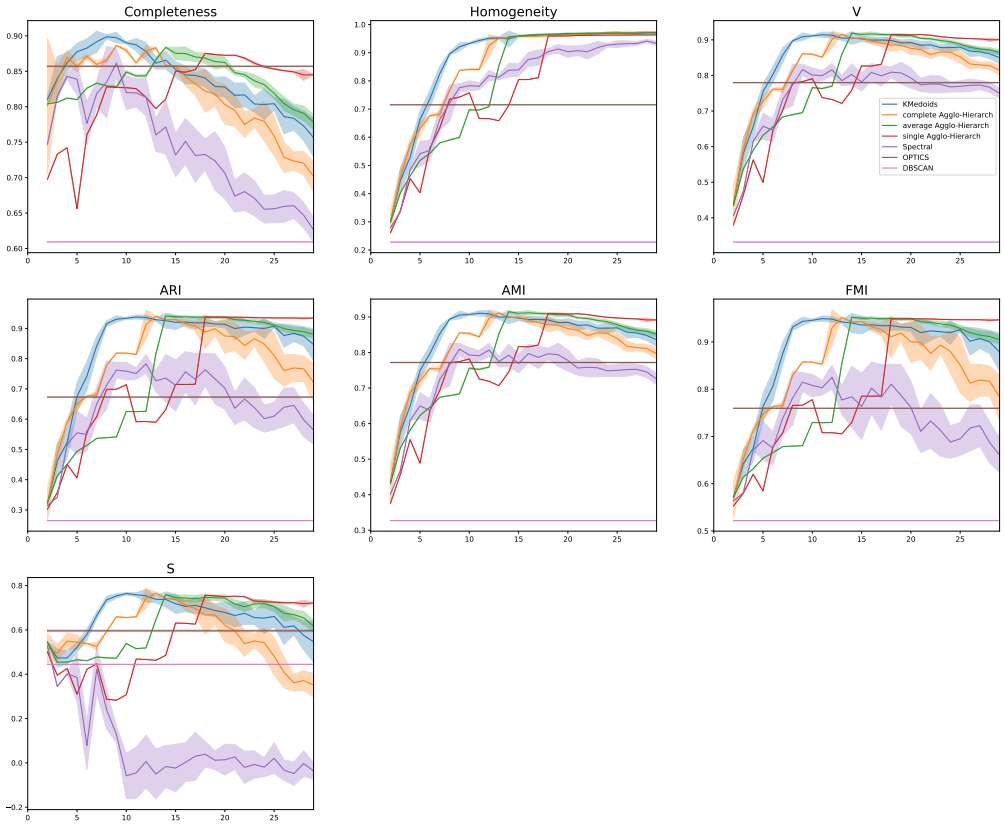


Figure 5.3 Performance variation of agglomerative clustering paired with LCSS ($d_z = 5\ m$) on inD 1

Despite the simplicity and popularity of k-Means, the use of centroids imposes limitation on its application in time-series clustering. Also, because of the nature of trajectories, the common definition of centroid is generally not meaningful for a set of trajectories. Therefore, in order to use k-Means in such settings, trajectories should first be mapped into an embedding space. There are several sophisticated embedding techniques used in the literature in coordination with k-Means [39].

As an experiment, we experimented with the distance matrices as the input for the k-Means algorithm. This means that we mapped each trajectory into an n-dimensional embedding space where each object i is associated with a trajectory M^i and represents the distances to all the trajectories.

Figure 5.4 shows the distance and clustering algorithms associated with highest rank according to our proposed combined evaluation measure.

As we can see, the results are generally similar to Figure 4.8, but k-Means also appears among the top-ten clustering algorithms with highest rank of combined evaluation measure. Indeed, even though it is not dominant in any of the sites, it appears in most cases. On the other hand, this minor change (compared to Figure 4.8) had no effect on the distance measures associated with the top-ten clustering setups. There is the only a change in distance parameter value of only one (out of ten) clustering setup for the inD 2 dataset.

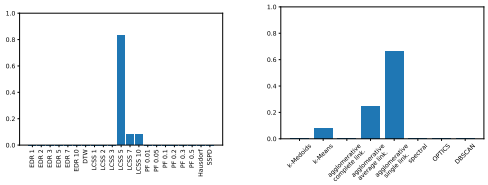


Figure 5.4 inD intersection 1

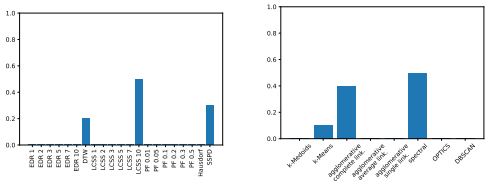


Figure 5.4 inD intersection 2

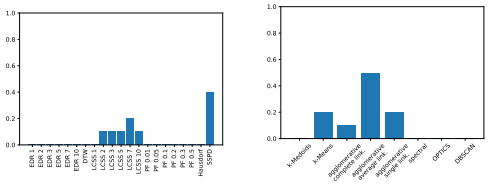


Figure 5.4 inD intersection 3

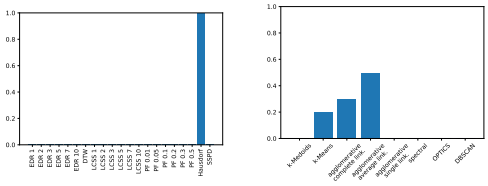


Figure 5.4 NGSIM intersection 1

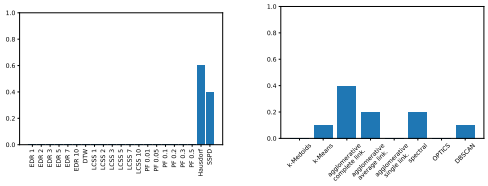


Figure 5.4 NGSIM intersection 2

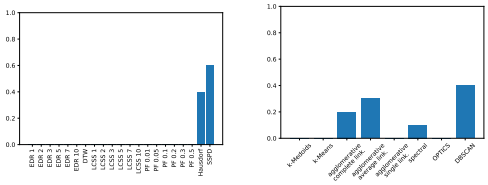


Figure 5.4 NGSIM intersection 3

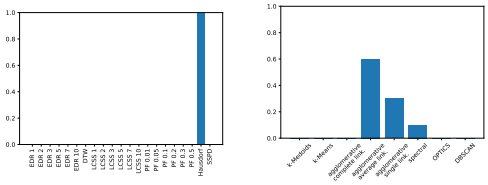


Figure 5.4 NGSIM intersection 4

CHAPTER 6 GENERAL DISCUSSION

The approach for achieving the objectives of the research project was explained in Chapter 3, 4 and 5. This chapter provides a summary of the results, as well as a critical analysis of the methodology and techniques used. Also, it discusses whether they were appropriate to achieve the objectives.

As a reminder, the primary objective of the work presented here was to investigate the problem of trajectory clustering evaluation in the absence of ground-truth labels. We aimed to develop a framework to find suitable setups for trajectory clustering that do not require human intervention. There was also a second objective to compare different popular trajectory distance measures and clustering algorithms with a range of parameters using the developed evaluation method.

The following subsections provide a summary of the methodology used, their results, the decision made in this regard, as well as the evaluation of these decisions.

6.1 Trajectory Clustering Framework

To address the primary objective of the research, a framework was developed for comparing clustering algorithms and trajectory distance measures. The trajectory clustering framework and its components were covered in the article presented in chapter 4.3.

The approach is unique in its emphasis on unsupervised performance evaluation of trajectory clustering. The framework developed in this work does not require the intensive human labor for labeling observations or providing reference clusters. Therefore, it can be even used to find suitable site-specific clustering setups on road networks with numerous intersections where generating manually labeled samples for every site would be labor intensive (beside the subjectivity of the labels).

As it was initially aimed in the objectives, the entire framework was coded in open source platforms using a single programming language and it does not need manual trajectory labeling. It is available online in a GitHub repository to encourage the use and replication of this work.

The steps are completely integrated and they work together without any essential intervention required from the operator. They also work without requiring any expert supervision. Once the dataset is defined and the desired distance measures and clustering algorithms are picked, it does a comprehensive comparison across different clustering setups and returns the best

performing setups according to the combined evaluation measure.

The only remaining human intervention in the process, which is optional, would be the choice of k_O and k_D, the number of origin and destination clusters. To facilitate this task, an option is included to visually help the operator for making the decision through using elbow method on origin/destination clustering performance results. This can also be avoided by pre-defining the values if the operator is willing to do so and has enough confidence in his/her choice.

Generally, trajectory clustering deals with objects of time-series type. This adds to the complexity of the clustering problem (compared to regular point clustering). Yet in this study, with the simple proposed method of using the two extracted features, this complexity was turned to an opportunity.

6.2 Comparison Of Clustering Setups

A comparison of clustering setups with different popular distances and clustering algorithms was provided in the sections 4.4 and 5.2. Six trajectory distance measures, three of the distances each with six different parameter values, and seven popular clustering algorithms, one of them with three different parameter values, were used for the comparison.

Overall, the results show that there is no specific distance measure or clustering algorithm that systematically outperforms the others or at least always appears among the top ten best clustering setups for all intersections. On the other hand, there are clear winners for specific, though not every, intersections. These findings are in line with the concerns mentioned in the literature about sensitivity of clustering with respect to data-specific characteristics [56].

As mentioned in Chapter 2, most studies in the trajectory clustering domain assume that their clustering method/setup has robust performance across different datasets. However, the experiments used in this work and the provided examples show that the choice of clustering setup may greatly depend on data specifications. Therefore, the results of the comparison of clustering setup presented at the end of chapter 4.4 show that the superiority of performance of a distance measure or clustering algorithm is not necessarily valid across different datasets.

Indeed, the sample observation presented in section 5.2.2 shows that even if the performance of a clustering algorithm has low variance on a dataset, suggesting reliable performance values, this may not hold true with the same distance measure on another dataset. Therefore, checking site-specific suitability of algorithms and also reiterating the evaluation is essential.

CHAPTER 7 CONCLUSION

In the previous chapter, the objectives of the research project were reiterated. A critical analysis of the methodology was also carried out to determine whether it was appropriate to achieve the objectives. The current chapter closes the study by presenting a synthesis of the work, its contributions, the limitations of the approach and the perspectives for future research work.

7.1 Summary Of Works

The principal objective of this work was to develop a framework for the performance evaluation of trajectory clustering. This framework was supposed to work without using manually labeled trajectories or reference clusters. This makes it suitable for finding site-specific clustering setups while avoiding extensive human labor. To address this objective, a framework, with a combination of both unsupervised and supervised performance measures, was developed to automatically compare the clustering setups. A method was proposed to automatically generate trajectory reference clusters based on their origin and destination points to be used by supervised performance measures. Therefore, the entire procedure remains unsupervised both in the clustering and evaluation stages.

A comprehensive comparison of distance measures, clustering algorithms and evaluation measures was also carried out using trajectory data from seven intersections. The clustering setups used in the comparison task include six popular trajectory distance measures, three of them were tried with six parameter values, i.e. a total of 21 distances, six clustering algorithms, different numbers of clusters when used as input by some of the clustering algorithms ranging from 2 to 30 and seven performance measures were tested. The experiment was done on seven intersections from two different datasets, i.e. collected in different countries, different road category types and also different data collection methods.

Finally, after using a metric based on a combination of performance measures, we picked the top performing distances and clustering algorithms. The results show that there is no single combination of distance and clustering algorithm that is always among the top ten clustering setups. Even looking at them independently, the finding was similar. There are distances that performed better compared to the others in specific intersections, but none of them are present among the top performers for every intersection. Between all the clustering algorithms, agglomerative hierarchical clustering was the only one that was in the

top performers in all seven intersections, though not always with the same linkage type. Also the plots do not show a visual dominance of this approach over other clustering algorithms based on frequency of its appearance in the top performers for each intersection.

Based on these findings, we believe that the choices of similarity measure and clustering algorithm may depend on the intersection type, environment as well as road user behaviors. Therefore, we suggest that similar comparison procedures to be followed for new sites among the candidate setups. To facilitate this, we have made all the essential codes developed in this work available online under an open source license (GitHub repository).

7.2 Limitations

The limitations can be split into data as well as methodological limitations. When considering the methodological aspect, one may note the following issues:

- The choice of parameters of DBSCAN and OPTICS is highly dependent on the range and distribution of distance values. However, the choice of distance measure in trajectory clustering is a problem of its own and many distance measures with a range of parameter values are involved in the comparison. Therefore, one should either use fixed (or a limited number of parameter values) for these algorithms, or increase the run-time to the level that considering a vast number of parameter values would not be an issue.
- As mentioned in Chapter 6, even though the steps of the framework are fully integrated and work with almost no intervention required by the operator, the choice of k_O and k_D still needs the operator to make a decision.

Regarding the data, the following points can be considered:

- Although NGSIM data is a de facto standard for benchmarking empirical researches in traffic studies, it was collected about two decades ago. This implies that the driving behaviour represented in the data might not be a perfect representation of the actual driver behaviour.
- Both the NGSIM and inD datasets are related to a very short time period (e.g. two 15 minutes intervals within one day in the case of NGSIM). Thus, they represent a subset of the traffic pattern distributions over a day/week/year.

7.3 Future Research

A systematic search of the clustering algorithm parameters and distances to determine whether some results depend on the choice of parameters is suggested.

By combining supervised and unsupervised performance metrics, this method is versatile, and thus easy to apply to more sites and datasets. However, as all the noted algorithms rely on comparing all the trajectories by computing their distances, the time consumption for creating distance matrices is of $O(n^2)$. Using methods such as recurrent neural networks (RNN) and variational auto-encoders (VAE) [3] may help and decrease the complexity to $O(n)$.

Further work is necessary to better characterize clustering performance, e.g. based on domain specific tasks like origin-destination clustering. Also, it is recommended to apply these methods to more diverse datasets and better verify the factors that may determine clustering performance.

REFERENCES

[1] X. Yu and P. D. Prevedouros, "Performance and challenges in utilizing non-intrusive sensors for traffic data collection," *Advances in Remote Sensing*, vol. 2, pp. 45–50, 2013.

[2] V. Mayer-Schönberger and K. Cukier, *Big data: A revolution that will transform how we live, work, and think*. Houghton Mifflin Harcourt, 2013.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[4] E. Barmpounakis and N. Geroliminis, "On the new era of urban traffic monitoring with massive drone data: The pneuma large-scale field experiment," *Transportation research part C: emerging technologies*, vol. 111, pp. 50–71, 2020.

[5] C. Antoniou, R. Balakrishna, and H. N. Koutsopoulos, "A synthesis of emerging data collection technologies and their impact on traffic management applications," *European Transport Research Review*, vol. 3, no. 3, pp. 139–148, 2011.

[6] A. Haghani *et al.*, "Freeway travel time ground truth data collection using bluetooth sensors," *Transportation Research Record, Journal of Transportation Research Board*, vol. 2160, pp. 60–68, 2010.

[7] M. Bernas *et al.*, "A survey and comparison of low-cost sensing technologies for road traffic monitoring," *Sensors*, vol. 18, no. 10, p. 3243, 2018.

[8] N. Buch, S. A. Velastin, and J. Orwell, "A review of computer vision techniques for the analysis of urban traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 920–939, 2011.

[9] F. Bardet and T. Chateau, "Mcmc particle filter for real-time visual tracking of vehicles," in *2008 11th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2008, pp. 539–544.

[10] P. Guha, A. Mukerjee, and K. Venkatesh, "Appearance based multiple agent tracking under complex occlusions," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2006, pp. 593–602.

[11] N. K. Kanhere and S. T. Birchfield, "Real-time incremental segmentation and tracking of vehicles at low camera angles using stable features," *IEEE transactions on intelligent transportation systems*, vol. 9, no. 1, pp. 148–160, 2008.

[12] X. Li and F. M. Porikli, "A hidden markov model framework for traffic event detection using video features," in *2004 International Conference on Image Processing, 2004. ICIP'04.*, vol. 5. IEEE, 2004, pp. 2901–2904.

[13] M. Trajkovic and S. Gutta, "Vision-based method and apparatus for monitoring vehicular traffic events," Aug. 27 2002, uS Patent 6,442,474.

[14] B. Morris and M. Trivedi, "Real-time video based highway traffic measurement and performance monitoring," in *2007 IEEE Intelligent Transportation Systems Conference*. IEEE, 2007, pp. 59–64.

[15] M. Bommes *et al.*, "Video based intelligent transportation systems–state of the art and future development," *Transportation Research Procedia*, vol. 14, pp. 4495–4504, 2016.

[16] M. Aron, S. Cohen, and R. Seidowsky, "Two french hard-shoulder running operations: Some comments on effectiveness and safety," in *13th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2010, pp. 230–236.

[17] M. Y. Aalsalem, W. Z. Khan, and K. M. Dhabbah, "An automated vehicle parking monitoring and management system using anpr cameras," in *2015 17th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2015, pp. 706–710.

[18] X. Wang *et al.*, "A video-based traffic violation detection system," in *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*. IEEE, 2013, pp. 1191–1194.

[19] C. M. Bishop, *Pattern recognition and machine learning*. Springer Science+ Business Media, 2006.

[20] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward," *PloS one*, vol. 13, no. 3, p. e0194889, 2018.

[21] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah, "Time-series clustering–a decade review," *Information Systems*, vol. 53, pp. 16–38, 2015.

[22] S. Rani and G. Sikka, "Recent techniques of clustering of time series data: a survey," *International Journal of Computer Applications*, vol. 52, no. 15, 2012.

[23] H. Ding *et al.*, "Querying and mining of time series data: experimental comparison of representations and distance measures," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1542–1552, 2008.

[24] L. N. Ferreira and L. Zhao, "Time series clustering via community detection in networks," *Information Sciences*, vol. 326, pp. 227–242, 2016.

[25] J. Bian *et al.*, "A survey on trajectory clustering analysis," *arXiv preprint arXiv:1802.06971*, 2018.

[26] T. W. Liao, "Clustering of time series data—a survey," *Pattern recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[27] Z. Kan *et al.*, "Traffic congestion analysis at the turn level using taxis' gps trajectory data," *Computers, Environment and Urban Systems*, vol. 74, pp. 229–243, 2019.

[28] M. Rezaie *et al.*, "Semi-supervised travel mode detection from smartphone data," in *2017 International Smart Cities Conference (ISC2)*. IEEE, 2017, pp. 1–8.

[29] J. Wang *et al.*, "Automatic intersection and traffic rule detection by mining motor-vehicle gps trajectories," *Computers, Environment and Urban Systems*, vol. 64, pp. 19–29, 2017.

[30] P. Besse *et al.*, "Review and perspective for distance based trajectory clustering," *arXiv preprint arXiv:1508.04904*, 2015.

[31] C. Myers, L. Rabiner, and A. Rosenberg, "Performance tradeoffs in dynamic time warping algorithms for isolated word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 6, pp. 623–635, 1980.

[32] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, USA:, 1994, pp. 359–370.

[33] K. Toohey and M. Duckham, "Trajectory similarity measures," *Sigspatial Special*, vol. 7, no. 1, pp. 43–50, 2015.

[34] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proceedings 18th international conference on data engineering*. IEEE, 2002, pp. 673–684.

[35] B. Morris and M. Trivedi, "Learning trajectory patterns by clustering: Experimental studies and comparative evaluation," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 312–319.

[36] P. C. Besse *et al.*, "Review and perspective for distance-based clustering of vehicle trajectories," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3306–3317, 2016.

[37] M. M. Fréchet, "Sur quelques points du calcul fonctionnel," *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, vol. 22, no. 1, pp. 1–72, 1906.

[38] R. Arfa, R. Yusof, and P. Shabanzadeh, "A dissimilarity measure estimation for analyzing trajectory data," *Journal of Advanced Simulation in Science and Engineering*, vol. 6, no. 2, pp. 367–385, 2019.

[39] W. Wang *et al.*, "Vehicle trajectory clustering based on dynamic representation learning of internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[40] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert systems with applications*, vol. 36, no. 2, pp. 3336–3341, 2009.

[41] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[42] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[43] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[44] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[45] U. Maulik and S. Bandyopadhyay, "Performance evaluation of some clustering algorithms and validity indices," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 12, pp. 1650–1654, 2002.

[46] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE transactions on pattern analysis and machine intelligence*, no. 2, pp. 224–227, 1979.

[47] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007, pp. 410–420.

[48] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *The Journal of Machine Learning Research*, vol. 11, pp. 2837–2854, 2010.

[49] J. M. Santos and M. Embrechts, "On the use of the adjusted rand index as a metric for evaluating supervised classification," in *International conference on artificial neural networks*. Springer, 2009, pp. 175–184.

[50] B. Desgraupes, "Clustering indices," *University of Paris Ouest-Lab Modal'X*, vol. 1, p. 34, 2013.

[51] M. Y. Choong *et al.*, "Modeling of vehicle trajectory clustering based on lcss for traffic pattern extraction," in *2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS)*. IEEE, 2017, pp. 74–79.

[52] Z. Fu, W. Hu, and T. Tan, "Similarity based vehicle trajectory clustering and anomaly detection," in *IEEE International Conference on Image Processing 2005*, vol. 2. Ieee, 2005, pp. II–602.

[53] L. Zhao and G. Shi, "A trajectory clustering method based on douglas-peucker compression and density for marine traffic pattern recognition," *Ocean Engineering*, vol. 172, pp. 456–467, 2019.

[54] F. Bélisle *et al.*, "Optimized video tracking for automated vehicle turning movement counts," *Transportation Research Record*, vol. 2645, no. 1, pp. 104–112, 2017.

[55] H. Xue, D. Q. Huynh, and M. Reynolds, "Poppl: Pedestrian trajectory prediction by lstm with automatic route class clustering," *IEEE transactions on neural networks and learning systems*, 2020.

[56] G. W. Milligan and M. C. Cooper, "Methodology review: Clustering methods," *Applied psychological measurement*, vol. 11, no. 4, pp. 329–354, 1987.

[57] D. Cai and X. Chen, "Large scale spectral clustering via landmark-based sparse representation," *IEEE transactions on cybernetics*, vol. 45, no. 8, pp. 1669–1680, 2014.

[58] G. Van Rossum *et al.*, "Python programming language." in *USENIX annual technical conference*, vol. 41, 2007, p. 36.

[59] S. Shell, "An introduction to python for scientific computing," *PDF). Retrieved*, vol. 3, 2019.

[60] J. Halkias and J. Colyar, "Next generation simulation fact sheet," *US Department of Transportation: Federal Highway Administration*, 2006.

[61] B. Coifman and L. Li, "A critical evaluation of the next generation simulation (ngsim) vehicle trajectory dataset," *Transportation Research Part B: Methodological*, vol. 105, pp. 362–377, 2017.

[62] V. Alexiadis, "Video-based vehicle trajectory data collection," in *Transportation Research Board 86th Annual Meeting*. Citeseer, 2006.

[63] J. Bock *et al.*, "The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1929–1934.

[64] L. Moreira-Matias *et al.*, "Time-evolving od matrix estimation using high-speed gps data streams," *Expert systems with Applications*, vol. 44, pp. 275–288, 2016.

[65] T. Bandaragoda *et al.*, "Trajectory clustering of road traffic in urban environments using incremental machine learning in combination with hyperdimensional computing," in *2019 IEEE intelligent transportation systems conference (ITSC)*. IEEE, 2019, pp. 1664–1670.

[66] E. Kwon *et al.*, "Scene modeling-based anomaly detection for intelligent transport system," in *2013 4th International Conference on Intelligent Systems, Modelling and Simulation*. IEEE, 2013, pp. 252–257.

[67] R. Zhen *et al.*, "Maritime anomaly detection within coastal waters based on vessel trajectory clustering and naïve bayes classifier," *The Journal of Navigation*, vol. 70, no. 3, p. 648, 2017.

[68] C. Katrakazas *et al.*, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, 2015.

[69] N. Saunier, T. Sayed, and C. Lim, "Probabilistic collision prediction for vision-based automated road safety analysis," in *2007 IEEE Intelligent Transportation Systems Conference*. IEEE, 2007, pp. 872–878.

[70] M. Y. Choong *et al.*, "Vehicle trajectory clustering for traffic intersection surveillance," in *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*. IEEE, 2016, pp. 1–4.

[71] S. Chen, B. Ma, and K. Zhang, "On the similarity metric and the distance metric," *Theoretical Computer Science*, vol. 410, no. 24-25, pp. 2365–2376, 2009.

[72] T. M. Kodinariya and P. R. Makwana, "Review on determining number of cluster in k-means clustering," *International Journal*, vol. 1, no. 6, pp. 90–95, 2013.

[73] Paul Paczuski. (2018) Sqlite vs pandas: Performance benchmarks. [Online]. Available: https://blog.thedataincubator.com/2018/05/sqlite-vs-pandas-performance-benchmarks/

[74] Artem Golubin. (2018) How numba and cython speed up python code. [Online]. Available: https://rushter.com/blog/numba-cython-python-optimization/

[75] PickupBrain. (2020) Speed up python up to 1 million times: Cython vs numba. [Online]. Available: https://www.pickupbrain.com/python/speed-up-python-up-to-1-million-times-cython-vs-numba/

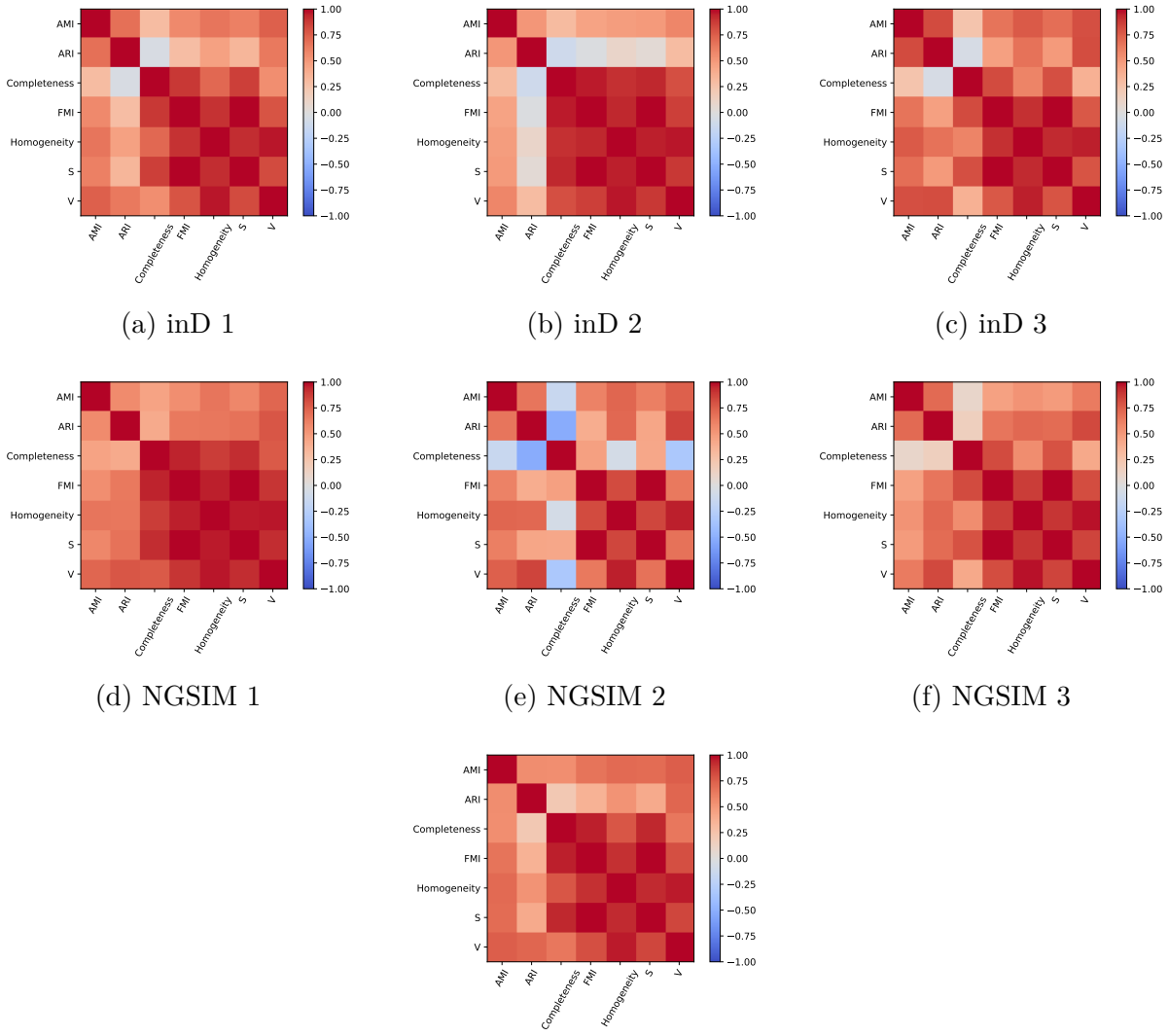
[76] B. GUILLOUET, "A python package for computing distance between 2d trajectories."

[77] P. Berkhin, "Survey of clustering data mining techniques. accrue software," *Inc. TR, San Jose, USA*, 2002.

[78] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Stanford, Tech. Rep., 2006.

[79] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

APPENDIX A PERFORMANCE MEASURE CORRELATIONS



(a) inD 1 (b) inD 2 (c) inD 3

(d) NGSIM 1 (e) NGSIM 2 (f) NGSIM 3

(g) NGSIM 4

Figure A.1 Correlations between different performance measures for each intersection